MITSUBISHI ELECTRIC

Programmable Automation Controllers

MELSEC MX Controller

MELSEC MX Controller MX-R Model Programming Manual

SAFETY PRECAUTIONS

(Read these precautions before using this product.)

Before using Mitsubishi Electric programmable controllers, please read the manuals for the products used and the relevant manuals carefully and pay full attention to safety to handle the products correctly.

Make sure that the end users read this manual and then keep the manual in a safe place for future reference.

INTRODUCTION

Thank you for purchasing the Mitsubishi Electric programmable controllers.
This manual describes the instructions and functions/function blocks required for programming.
Before using this product, please read this manual and the relevant manuals carefully and develop familiarity with the functions and performance of the controller to handle the product correctly.
When applying the program examples provided in this manual to an actual system, ensure the applicability and confirm that it will not cause system control problems.
Please make sure that the end users read this manual.

CONTENTS

SAFETY PRECAUTIONS 1
INTRODUCTION 2
RELEVANT MANUALS 20
TERMS 21
GENERIC TERMS AND ABBREVIATIONS 24

PART 1 PROGRAMMING OVERVIEW

CHAPTER 1 OVERVIEW 26

1.1 Instruction Configuration 26
1.2 Data Specification Method 28
Bit data 31
16-bit data (word data) 32
32-bit data (double word data) 35
Real number data (floating-point data) 38
String data 41
1.3 Execution Condition 43
1.4 Precautions on Programming 44
Errors common to instructions 44
Checking the ranges of instruction runtime devices and labels 44
Operation when a long timer or long retentive timer device is used 50
Precautions for using a link device assigned to a synchronized station as an instruction argument 52
Operations arising when the OUT, SET/RST, and PLS/PLF instructions of the same device are used 52
Restrictions on using file registers 58

CHAPTER 2 PROGRAM ORGANIZATION UNITS 59

2.1 Program Blocks 60
2.2 Functions (FUN) 61
2.3 Function Blocks (FB) 65
2.4 Precautions 76
When a function is used 76
When a function block is used 76
Operations of when the MC/MCR instructions are used to control EN 80

CHAPTER 3 LADDER DIAGRAM 85

3.1 Configuration 85
Ladder symbols 86
Program execution order 87
Precautions for using a function block in ladder diagram 87
3.2 Inline ST 88
3.3 Statements and Notes 90

CHAPTER 4 STRUCTURED TEXT LANGUAGE 91

4.1 Configuration 92
Delimiter 93
Operators 93
Instruction 94

Control statements . . . 95
Constants . . . 104
Labels and devices . . . 105
Comments . . . 107

CHAPTER 5 FBD/LD 108

5.1 Configuration . . . 108
Program elements . . . 109
Constant . . . 116
Labels and devices . . . 116
5.2 Inline ST . . . 118
5.3 Program Execution Order . . . 119
Execution order of program elements . . . 119

PART 2 LISTS OF INSTRUCTIONS AND FUN/FB

CHAPTER 6 INSTRUCTION LIST 122

6.1 Sequence Instructions . . . 122
6.2 Basic Instructions . . . 126
6.3 Application Instructions . . . 145
Program control . . . 145
Data processing . . . 147
Debugging and failure diagnostic . . . 155
String processing . . . 155
Real value processing . . . 158
Random number . . . 165
Device operation . . . 166
Timer, counter . . . 167
Clock . . . 168
Module access . . . 171
PID operation instruction . . . 173
PID control instructions . . . 173
Add-on instructions . . . 174

CHAPTER 7 NETWORK INSTRUCTION LIST 175

7.1 Ethernet Instructions . . . 175

CHAPTER 8 STANDARD FUNCTION/STANDARD FUNCTION BLOCK LISTS 177

8.1 Standard Functions . . . 177
8.2 Standard Function Blocks . . . 188

CHAPTER 9 MOTION CONTROL FUNCTION BLOCK LIST 190

9.1 Administrative Function Blocks . . . 190
9.2 Motion Function Blocks . . . 193
9.3 Standard Function Blocks . . . 197

CHAPTER 10 NETWORK FUNCTION BLOCK LIST 198

10.1 Function Blocks for Ethernet . . . 198
10.2 Function Blocks for CC-Link IE TSN . . . 199

PART 3 SEQUENCE INSTRUCTIONS/BASIC INSTRUCTIONS

CHAPTER 11 SEQUENCE INSTRUCTIONS 202

11.1 Contact Instructions 202
Operation start, series connection, parallel connection 202
Pulse operation start, pulse series connection, pulse parallel connection 204
Pulse NOT operation start, pulse NOT series connection, pulse NOT parallel connection 207
11.2 Association Instructions 210
Ladder block series/parallel connection 210
Inverting the operation result 211
Converting the operation result into a pulse 212
Converting the edge relay operation result into a pulse 213
11.3 Output Instructions 215
Out (excluding the timer, counter, and annunciator) 215
Timer 217
Long timer 220
Counter 223
Long counter 225
Annunciator 227
Setting devices (excluding annunciator) 228
Resetting devices (excluding annunciator) 230
Setting annunciator 232
Resetting annunciator 234
Rising edge output 236
Falling edge output 238
Inverting the bit device output 240
Converting the direct access output into a pulse 241
11.4 Shift Instructions 243
Shifting bit devices 243
11.5 Master Control Instructions 245
Setting/resetting a master control 245
11.6 Termination Instructions 249
Ending the main routine program 249
Ending the sequence program 250
11.7 Stop Instruction 252
Stopping the sequence program 252
11.8 No Operation Instruction 253
No operation (NOPLF) 253

CHAPTER 12 BASIC INSTRUCTIONS 254

12.1 Comparison Operation Instructions 254
Comparing 16-bit binary data 254
Comparing 32-bit binary data 256
Comparing 16-bit binary block data 258
Comparing 32-bit binary block data 261
12.2 Arithmetic Operation Instructions 264
Adding 16-bit binary data 264
Subtracting 16-bit binary data 268

CONTENTS

Adding 32-bit binary data. 272
Subtracting 32-bit binary data 276
Multiplying 16-bit binary data. 280
Dividing 16-bit binary data 282
Multiplying 32-bit binary data. 284
Dividing 32-bit binary data 286
Adding BCD 4-digit data 288
Subtracting BCD 4-digit data 291
Adding BCD 8-digit data 294
Subtracting BCD 8-digit data 297
Multiplying BCD 4-digit data 300
Dividing BCD 4-digit data 302
Multiplying BCD 8-digit data 304
Dividing BCD 8-digit data 306
Adding 16-bit binary block data 308
Subtracting 16-bit binary block data 310
Adding 32-bit binary block data 312
Subtracting 32-bit binary block data 315
Incrementing 16-bit binary data 318
Decrementing 16-bit binary data 319
Incrementing 32-bit binary data 320
Decrementing 32-bit binary data 321
12.3 Logical Operation Instructions 323
Performing an AND operation on 16-bit data. 323
Performing an AND operation on 32-bit data. 327
Performing an AND operation on 16-bit block data 331
Performing an OR operation on 16-bit data. 333
Performing an OR operation on 32-bit data. 336
Performing an OR operation on 16-bit block data 340
Performing an XOR operation on 16-bit data 342
Performing an XOR operation on 32-bit data 345
Performing an XOR operation on 16-bit block data. 349
Performing an XNOR operation on 16-bit data 351
Performing an XNOR operation on 32-bit data 354
Performing an XNOR operation on 16-bit block data 358
12.4 Bit Processing Instructions 360
Setting a bit in the word device 360
Resetting a bit in the word device 362
Performing a 16-bit test 364
Performing a 32-bit test 366
Batch-resetting bit devices 368
12.5 Shift Instructions 370
Shifting 16-bit binary data to the right by n bit(s) 370
Shifting 16-bit binary data to the left by n bit(s) 372
Shifting n-bit data to the right by one bit 374
Shifting n-bit data to the left by one bit 376
Shifting n-word data to the right by one word 378
Shifting n-word data to the left by one word 380
Shifting n-bit data to the right by n bit(s) 382
Shifting n-bit data to the left by n bit(s) 384
Shifting n-word data to the right by n word(s) 386

Shifting n-word data to the left by n word(s) . . . 388
12.6 Data Conversion Instructions . . . 390
Converting binary data to BCD 4-digit data . . . 390
Converting binary data to BCD 8-digit data . . . 392
Converting BCD 4-digit data to 16-bit binary data . . . 394
Converting BCD 8-digit data to 32-bit binary data . . . 396
Converting single-precision real number to 16-bit signed binary data . . . 398
Converting single-precision real number to 16-bit unsigned binary data . . . 400
Converting single-precision real number to 32-bit signed binary data . . . 402
Converting single-precision real number to 32-bit unsigned binary data . . . 404
Converting double-precision real number to 16-bit signed binary data . . . 406
Converting double-precision real number to 16-bit unsigned binary data . . . 408
Converting double-precision real number to 32-bit signed binary data . . . 410
Converting double-precision real number to 32-bit unsigned binary data . . . 412
Converting 16-bit signed binary data to 16-bit unsigned binary data . . . 414
Converting 16-bit signed binary data to 32-bit signed binary data . . . 416
Converting 16-bit signed binary data to 32-bit unsigned binary data . . . 418
Converting 16-bit unsigned binary data to 16-bit signed binary data . . . 420
Converting 16-bit unsigned binary data to 32-bit signed binary data . . . 422
Converting 16-bit unsigned binary data to 32-bit unsigned binary data . . . 424
Converting 32-bit signed binary data to 16-bit signed binary data . . . 426
Converting 32-bit signed binary data to 16-bit unsigned binary data . . . 428
Converting 32-bit signed binary data to 32-bit unsigned binary data . . . 430
Converting 32-bit unsigned binary data to 16-bit signed binary data . . . 432
Converting 32-bit unsigned binary data to 16-bit unsigned binary data . . . 434
Converting 32-bit unsigned binary data to 32-bit signed binary data . . . 436
Converting 16-bit binary data to Gray code data . . . 438
Converting 32-bit binary data to Gray code data . . . 440
Converting 16-bit binary Gray code data to 16-bit binary data . . . 442
Converting 32-bit binary Gray code data to 32-bit binary data . . . 444
Converting 16-bit binary data block to BCD 4-digit data block . . . 446
Converting BCD 4-digit block data to 16-bit binary block data . . . 448
Converting decimal ASCII data to 16-bit binary data . . . 450
Converting decimal ASCII data to 32-bit binary data . . . 453
Converting hexadecimal ASCII data to 16-bit binary data . . . 457
Converting hexadecimal ASCII data to 32-bit binary data . . . 459
Converting decimal ASCII data to BCD 4-digit data . . . 462
Converting decimal ASCII data to BCD 8-digit data . . . 465
Converting decimal string data to 16-bit binary data . . . 468
Converting decimal string data to 32-bit binary data . . . 471
Converting hexadecimal ASCII to hexadecimal binary data . . . 474
Converting single-precision real number to BCD format data . . . 476
Two's complement of 16-bit binary data (sign inversion) . . . 478
Two's complement of 32-bit binary data (sign inversion) . . . 480
Decoding 8-bit data to 256-bit data . . . 481
Encoding 256-bit data to 8-bit data . . . 483
Decoding data to seven-segment display data . . . 485
Separating data in units of 4 bits . . . 488
Combining data in units of 4 bits . . . 490
Separating data in units of bits . . . 492
Combining data in units of bits . . . 494

Separating data in units of bytes . . . 496
Combining data in units of bytes . . . 498
12.7 Data Transfer Instructions . . . 500
Transferring 16-bit binary data . . . 500
Transferring 32-bit binary data . . . 502
Inverting and transferring 16-bit binary data . . . 504
Inverting and transferring 32-bit binary data . . . 506
Inverting and transferring 1-bit data. . . . 508
Transferring 16-bit binary data block (16 bits) . . . 510
Transferring 16-bit binary data block (32 bits) . . . 512
Transferring the same 16-bit binary data block (16 bits) . . . 514
Transferring the same 16-bit binary data block (32 bits) . . . 516
Transferring the same 32-bit binary data block (16 bits) . . . 518
Transferring the same 32-bit binary data block (32 bits) . . . 520
Exchanging 16-bit binary data . . . 522
Exchanging 32-bit binary data . . . 523
Exchanging 16-bit binary block data . . . 524
Exchanging the upper and lower bytes of 16-bit binary data. . . . 526
Exchanging the upper and lower bytes of 32-bit binary data. . . . 527
Transferring 1-bit data . . . 528
Transferring n-bit data . . . 529

PART 4 APPLICATION INSTRUCTIONS

CHAPTER 13 PROGRAM CONTROL 532

13.1 Program Branch Instructions. . . . 532
Pointer branch . . . 532
Jumping to END . . . 535
13.2 Program Execution Control Instructions . . . 536
Disabling/enabling interrupt programs. . . . 536
Disabling interrupt programs with specified priority or lower . . . 539
Interrupt program mask . . . 544
Disabling/enabling the specified interrupt pointer . . . 546
Returning from the interrupt program . . . 548
Resetting the watchdog timer . . . 549
13.3 Structure Creation Instructions . . . 550
Performing the FOR to NEXT instruction loop. . . . 550
Forcibly terminating the FOR to NEXT instruction loop . . . 553
Calling a subroutine program . . . 555
Returning from the subroutine program called . . . 561
Calling a subroutine program and turning the output off . . . 562
Calling a subroutine program in the specified program file . . . 568
Calling a subroutine program in the specified program file and turning the output off . . . 575
Calling a subroutine program . . . 580

CHAPTER 14 DATA PROCESSING 585

14.1 Rotation Instructions . . . 585
Rotating 16-bit binary data to the right . . . 585
Rotating 16-bit binary data to the left. . . . 588
Rotating 32-bit binary data to the right . . . 591

Rotating 32-bit binary data to the left. . . . 593
14.2 Data Table Operation Instructions . . . 595
Reading the oldest data from the data table . . . 595
Reading the newest data from the data table . . . 597
Writing data to the data table. . . . 599
Inserting data to the data table . . . 601
Deleting data from data table . . . 603
14.3 Reading/Writing Data Instructions . . . 605
Reading data from the data memory . . . 606
Writing data to the data memory . . . 608
14.4 File Operation Instructions . . . 610
Reading data from the specified file . . . 610
Writing data to the specified file. . . . 624
Deleting the specified file. . . . 638
Copying the specified file. . . . 641
Moving the specified file . . . 646
Renaming the specified file . . . 651
Acquiring the status of the specified file . . . 655
Error codes generated for file operation instructions. . . . 660
14.5 Data Control Instructions . . . 661
Upper and lower limit control of 16-bit binary data . . . 661
Upper and lower limit control of 32-bit binary data . . . 663
Dead band control of 16-bit binary data . . . 665
Dead band control of 32-bit binary data . . . 667
Zone control of 16-bit binary data . . . 669
Zone control of 32-bit binary data . . . 671
Scaling 16-bit binary data (point coordinates) . . . 673
Scaling 32-bit binary data (point coordinates) . . . 680
Scaling double-precision real number data (point coordinates) . . . 687
Scaling 16-bit binary data (XY coordinates) . . . 691
Scaling 32-bit binary data (XY coordinates) . . . 696
14.6 Data Processing Instructions . . . 701
Searching 16-bit binary data . . . 701
Searching 32-bit binary data . . . 703
Searching 16-bit binary data (minimum, match, maximum) . . . 705
Searching 32-bit binary data (minimum, match, maximum) . . . 707
Checking 16-bit binary data. . . . 709
Checking 32-bit binary data. . . . 710
Searching the maximum value of 16-bit binary data . . . 711
Searching the maximum value of 32-bit binary data . . . 713
Searching the minimum value of 16-bit binary data . . . 715
Searching the minimum value of 32-bit binary data . . . 717
Sorting 16-bit binary data . . . 719
Sorting 32-bit binary data . . . 721
Adding 16-bit binary data. . . . 723
Adding 32-bit binary data. . . . 725
Calculating the mean value of 16-bit binary data . . . 727
Calculating the mean value of 32-bit binary data . . . 729
Calculating the square root of 16-bit binary data. . . . 731
Calculating the square root of 32-bit binary data. . . . 732
CRC operation. . . . 733

CHAPTER 15 DEBUGGING AND FAILURE DIAGNOSTIC 735

15.1 Debugging and Failure Diagnostic Instructions . . . 735

Resetting the error display and the annunciator display . . . 735

CHAPTER 16 STRING PROCESSING 737

16.1 String Processing Instructions . . . 737

Comparing string data . . . 737
Concatenating string data . . . 740
Transferring string data . . . 744
Transferring Unicode string data . . . 746
Converting 16-bit binary data to decimal ASCII . . . 748
Converting 32-bit binary data to decimal ASCII . . . 753
Converting 16-bit binary data to hexadecimal ASCII . . . 758
Converting 32-bit binary data to hexadecimal ASCII . . . 762
Converting 16-bit binary data to string data . . . 766
Converting 32-bit binary data to string data . . . 769
Converting BCD 4-digit data to decimal ASCII code . . . 772
Converting BCD 8-digit data to decimal ASCII code . . . 776
Converting single-precision real number to string data . . . 781
Converting hexadecimal binary data to hexadecimal ASCII code . . . 787
Converting and concatenating multiple data strings . . . 789
Converting Unicode string to Shift JIS string . . . 805
Converting Shift JIS string to Unicode string (without byte order mark) . . . 808
Converting shift JIS string to Unicode (with byte order mark) . . . 810
Detecting a string length . . . 812
Extracting string data from the right . . . 814
Extracting string data from the left . . . 816
Extracting the specified string data . . . 818
Replacing the specified string data . . . 820
Searching string data . . . 823
Inserting string data . . . 825
Deleting string data . . . 827

CHAPTER 17 REAL VALUE PROCESSING 829

17.1 Floating-point Instruction . . . 829

Comparing single-precision real numbers . . . 829
Comparing double-precision real numbers . . . 831
Adding single-precision real numbers . . . 834
Subtracting single-precision real numbers . . . 838
Adding double-precision real numbers . . . 842
Subtracting double-precision real numbers . . . 846
Multiplying single-precision real numbers . . . 850
Dividing single-precision real numbers . . . 852
Multiplying double-precision real numbers . . . 854
Dividing double-precision real numbers . . . 856
Converting 16-bit signed binary data to single-precision real number . . . 858
Converting 16-bit unsigned binary data to single-precision real number . . . 859
Converting 32-bit signed binary data to single-precision real number . . . 860
Converting 32-bit unsigned binary data to single-precision real number . . . 862
Converting double-precision real number to single-precision real number . . . 864

Converting 16-bit signed binary data to double-precision real number . . . 866
Converting 16-bit unsigned binary data to double-precision real number . . . 867
Converting 32-bit signed binary data to double-precision real number . . . 868
Converting 32-bit unsigned binary data to double-precision real number . . . 869
Converting single-precision real number to double-precision real number . . . 870
Converting string data to single-precision real number . . . 872
Converting BCD format data to single-precision real number . . . 876
Inverting the sign of single-precision real number . . . 878
Inverting the sign of double-precision real number . . . 879
Transferring single-precision real number . . . 880
Transferring double-precision real number . . . 881
Calculating the sine of single-precision real number . . . 882
Calculating the cosine of single-precision real number . . . 884
Calculating the tangent of single-precision real number . . . 886
Calculating the arc sine of single-precision real number . . . 888
Calculating the arc cosine of single-precision real number . . . 890
Calculating the arc tangent of single-precision real number . . . 892
Calculating the sine of double-precision real number . . . 894
Calculating the cosine of double-precision real number . . . 896
Calculating the tangent of double-precision real number . . . 898
Calculating the arc sine of double-precision real number . . . 900
Calculating the arc cosine of double-precision real number . . . 902
Calculating the arc tangent of double-precision real number . . . 904
Calculating the sine of BCD data . . . 906
Calculating the cosine of BCD data . . . 908
Calculating the tangent of BCD data . . . 910
Calculating the arc sine of BCD data . . . 912
Calculating the arc cosine of BCD data . . . 914
Calculating the arc tangent of BCD data . . . 916
Converting single-precision real number angle to radian . . . 918
Converting single-precision real number radian to angle . . . 920
Converting double-precision real number angle to radian . . . 922
Converting double-precision real number radian to angle . . . 924
Calculating the square root of single-precision real number . . . 926
Calculating the square root of double-precision real number . . . 928
Calculating the exponent of single-precision real number . . . 930
Calculating the exponent of double-precision real number . . . 932
Calculating the natural logarithm of single-precision real number . . . 934
Calculating the natural logarithm of double-precision real number . . . 936
Calculating the square root of BCD 4-digit data . . . 938
Calculating the square root of BCD 8-digit data . . . 940
Calculating the exponentiation of single-precision real number . . . 942
Calculating the exponentiation of double-precision real number . . . 944
Calculating the common logarithm of single-precision real number . . . 946
Calculating the common logarithm of double-precision real number . . . 948
Searching the maximum value of single-precision real number . . . 950
Searching the maximum value of double-precision real number . . . 952
Searching the minimum value of single-precision real number . . . 954
Searching the minimum value of double-precision real number . . . 956

CHAPTER 18 RANDOM NUMBER 958

18.1 Random Number Instructions . . . 958
Generating random number . . . 958
Changing random sequence . . . 959

CHAPTER 19 DEVICE OPERATION 960

19.1 Index Register Instructions . . . 960
Saving all data of the index register . . . 960
Returning all data of the index register . . . 963
Saving the selected data of the index register and long index register . . . 964
Returning the selected data of the index register and long index register . . . 968
19.2 File Register Operation Instructions . . . 970
Switching the file register block number . . . 970
19.3 File Register Read/Write Instructions . . . 972
Reading 1-byte data from the file register . . . 972
Writing 1-byte data to the file register . . . 974
19.4 Indirect Address Read Instructions . . . 977
Reading the indirect address . . . 977

CHAPTER 20 TIMER, COUNTER 979

20.1 Special Counter Instructions . . . 979
Counting up or down the current value (1-phase input) . . . 979
Counting up or down the current value (2-phase input) . . . 981
20.2 Special Timer Instructions . . . 983
Teaching timer . . . 983
Special function timer . . . 985

CHAPTER 21 CLOCK 988

21.1 Clock Instructions . . . 988
Reading clock data . . . 988
Writing clock data . . . 990
Adding clock data . . . 992
Subtracting clock data . . . 994
Converting time data from hour/minute/second to second . . . 996
Converting time data from second to hour/minute/second . . . 998
Converting date and time data from date and time to second . . . 1000
Converting date and time data from second to date and time . . . 1002
Comparing date data . . . 1004
Comparing time data . . . 1008
Outputting a comparison result of time data . . . 1011
Reading expansion clock data . . . 1013
Adding expansion clock data . . . 1015
Subtracting expansion clock data . . . 1017
21.2 Timing Check Instructions . . . 1019
Generating timing pulses . . . 1019
Measuring time of the specified data . . . 1021

CHAPTER 22 MODULE ACCESS 1023

22.1 Module Access Instructions . . . 1023
Performing I/O refresh . . . 1023

Selecting refresh to be performed . . . 1025
Performing module refresh . . . 1027
Reading 1-word/2-word data from another module (16-bit specification) . . . 1029
Writing 1-word/2-word data to a module (16-bit specification) . . . 1032
Reading 1-word/2-word data from another module (32-bit specification) . . . 1037
Writing 1-word/2-word data to a module (32-bit specification) . . . 1041
Reading the module model name . . . 1046
Reading module specific information . . . 1050

CHAPTER 23 PID OPERATION INSTRUCTION 1057

23.1 Overview . . . 1057
Control data . . . 1059
Auto tuning . . . 1067
23.2 PID Operation Instruction . . . 1069

CHAPTER 24 PID CONTROL INSTRUCTIONS 1073

24.1 Overview . . . 1073
Operation method . . . 1073
PID control procedure . . . 1074
Helpful functions . . . 1080
24.2 PID Control Instructions (Inexact Differential) . . . 1082
Registering the PID control data to the controller . . . 1084
Performing PID operation . . . 1087
Stopping the operation of specified loop number . . . 1090
Starting the operation of specified loop number . . . 1091
Changing the parameters of specified loop number . . . 1092
24.3 PID Control Instructions (Exact Differential) . . . 1094
Registering the PID control data to the controller . . . 1096
Performing PID operation . . . 1098
Stopping the operation of specified loop number . . . 1101
Starting the operation of specified loop number . . . 1102
Changing the parameters of specified loop number . . . 1103

CHAPTER 25 ADD-ON 1105

25.1 Add-on Instructions . . . 1105
Add-on execution instruction . . . 1105

PART 5 NETWORK INSTRUCTIONS

CHAPTER 26 Ethernet INSTRUCTIONS 1110

26.1 Open/Close Processing Instructions . . . 1110
Opening a connection . . . 1110
Closing a connection . . . 1115
26.2 Socket Communications Instructions . . . 1117
Reading receive data during the END processing . . . 1117
Reading receive data when the instruction is executed . . . 1121
Sending data . . . 1124
Reading connection information . . . 1128
Changing the communication target (UDP/IP) . . . 1131

Changing the receive mode. 1133
Reading socket communications receive data 1137
26.3 Predefined Protocol Support Function Instruction 1139
Executing the registered protocols 1139
26.4 SLMP Frame Send Instruction 1147
Sending an SLMP frame 1147
Sending an encrypted SLMP frame. 1156
26.5 File Transfer Function Instructions 1165
Sending FTP client files. 1165
Retrieving FTP client files 1171

PART 6 STANDARD FUNCTIONS

CHAPTER 27 TYPE CONVERSION FUNCTIONS 1178

27.1 Converting BOOL to WORD 1178
27.2 Converting BOOL to DWORD. 1180
27.3 Converting BOOL to INT 1181
27.4 Converting BOOL to DINT 1182
27.5 Converting BOOL to TIME 1183
27.6 Converting BOOL to STRING 1184
27.7 Converting WORD to BOOL 1185
27.8 Converting WORD to DWORD 1186
27.9 Converting WORD to INT 1187
27.10 Converting WORD to DINT 1188
27.11 Converting WORD to TIME. 1190
27.12 Converting WORD to STRING 1191
27.13 Converting DWORD to BOOL. 1192
27.14 Converting DWORD to WORD 1193
27.15 Converting DWORD to INT 1195
27.16 Converting DWORD to DINT. 1197
27.17 Converting DWORD to TIME. 1198
27.18 Converting DWORD to STRING 1199
27.19 Converting INT to BOOL. 1200
27.20 Converting INT to WORD 1201
27.21 Converting INT to DWORD 1202
27.22 Converting INT to DINT. 1204
27.23 Converting INT to BCD 1205
27.24 Converting INT to REAL 1207
27.25 Converting INT to LREAL 1208
27.26 Converting INT to TIME. 1209
27.27 Converting INT to STRING 1210
27.28 Converting DINT to BOOL 1212
27.29 Converting DINT to WORD 1213
27.30 Converting DINT to DWORD. 1215
27.31 Converting DINT to INT. 1216
27.32 Converting DINT to BCD. 1217
27.33 Converting DINT to REAL. 1219
27.34 Converting DINT to LREAL. 1220
27.35 Converting DINT to TIME 1221
27.36 Converting DINT to STRING. 1222

27.37 Converting BCD to INT 1224
27.38 Converting BCD to DINT 1226
27.39 Converting BCD to STRING 1229
27.40 Converting REAL to INT 1231
27.41 Converting REAL to DINT 1233
27.42 Converting REAL to LREAL 1235
27.43 Converting REAL to STRING 1236
27.44 Converting LREAL to INT 1239
27.45 Converting LREAL to DINT 1240
27.46 Converting LREAL to REAL 1241
27.47 Converting TIME to BOOL 1242
27.48 Converting TIME to WORD 1243
27.49 Converting TIME to DWORD 1244
27.50 Converting TIME to INT 1245
27.51 Converting TIME to DINT 1246
27.52 Converting TIME to STRING 1247
27.53 Converting STRING to BOOL 1249
27.54 Converting STRING to WORD 1250
27.55 Converting STRING to DWORD 1251
27.56 Converting STRING to INT 1252
27.57 Converting STRING to DINT 1254
27.58 Converting STRING to BCD 1256
27.59 Converting STRING to REAL 1258
27.60 Converting STRING to TIME 1261
27.61 Converting Bit Array to INT 1262
27.62 Converting Bit Array to DINT 1263
27.63 Converting INT to Bit Array 1264
27.64 Converting DINT to Bit Array 1265
27.65 Copying the Bit Array 1266
27.66 Reading the Specified Bit of the Word Label 1267
27.67 Writing the Specified Bit of the Word Label 1269
27.68 Copying the Specified Bit of the Word Label 1271
27.69 Getting the Start Data 1273

CHAPTER 28 SINGLE VARIABLE FUNCTIONS 1274

28.1 Calculating the Absolute Value 1274
28.2 Calculating the Square Root 1276
28.3 Calculating the Natural Logarithm 1277
28.4 Calculating the Common Logarithm 1278
28.5 Calculating the Exponent 1280
28.6 Calculating the Sine 1281
28.7 Calculating the Cosine 1282
28.8 Calculating the Tangent 1283
28.9 Calculating the Arc Sine 1284
28.10 Calculating the Arc Cosine 1285
28.11 Calculating the Arc Tangent 1286

CHAPTER 29 ARITHMETIC OPERATION FUNCTIONS 1287

29.1 Addition 1287
29.2 Multiplication 1290

29.3 Subtraction. 1292
29.4 Division. 1295
29.5 Remainder 1297
29.6 Exponentiation. 1299
29.7 Assignment 1300

CHAPTER 30 BIT SHIFT FUNCTIONS 1302

30.1 Shifting Data to the Left by n Bit(s) 1302
30.2 Shifting Data to the Right by n Bit(s). 1304
30.3 Rotating Data to the Left by n Bit(s) 1306
30.4 Rotating Data to the Right by n Bit(s) 1308

CHAPTER 31 BOOLEAN FUNCTIONS 1310

31.1 AND Operation, OR Operation, and XOR Operation 1310
31.2 NOT Operation. 1313

CHAPTER 32 SELECTION FUNCTIONS 1314

32.1 Selecting a Value. 1314
32.2 Selecting the Maximum/Minimum Value 1316
32.3 Controlling the Upper/Lower Limit 1318
32.4 Multiplexer 1321

CHAPTER 33 COMPARISON FUNCTIONS 1323

33.1 Comparing Data. 1323
33.2 Comparing Data. 1325

CHAPTER 34 STRING FUNCTIONS 1327

34.1 Detecting a String Length. 1327
34.2 Extracting String Data From the Left/Right. 1329
34.3 Extracting String Data. 1332
34.4 Concatenating String Data 1334
34.5 Inserting String Data 1336
34.6 Deleting String Data 1338
34.7 Replacing String Data 1340
34.8 Searching String Data. 1343

CHAPTER 35 TIME DATA TYPE FUNCTIONS 1345

35.1 Addition 1345
35.2 Subtraction. 1347
35.3 Multiplication 1349
35.4 Division. 1351

PART 7 STANDARD FUNCTION BLOCKS

CHAPTER 36 BISTABLE FUNCTION BLOCKS 1354

36.1 Bistable Function Block (Set-Dominant). 1354
36.2 Bistable Function Block (Reset-Dominant). 1356

CHAPTER 37 EDGE DETECTION FUNCTION BLOCKS 1358
37.1 Detecting a Rising Edge . . . 1358
37.2 Detecting a Falling Edge . . . 1360

CHAPTER 38 COUNTER/TIMER FUNCTION BLOCKS 1362
38.1 Up Counter . . . 1362
38.2 Down Counter . . . 1364
38.3 Up/Down Counter . . . 1366
38.4 Counter Function Block . . . 1369
38.5 Pulse Timer . . . 1371
38.6 On Delay Timer . . . 1374
38.7 Off Delay Timer . . . 1377
38.8 Timer Function Block . . . 1379

PART 8 MOTION CONTROL FUNCTION BLOCK

CHAPTER 39 MOTION CONTROL FB OVERVIEW 1384
39.1 Motion Control FBs . . . 1384
How to use motion control FBs . . . 1384
Operation of motion control FBs for each system status . . . 1384
Motion control FB types . . . 1385
Motion control types . . . 1391
Error processing . . . 1391
Units used in control . . . 1391
I/O variables used in motion control FBs . . . 1392
Precautions for the motion control FB . . . 1393
39.2 Motion Control FB Configuration . . . 1394
Ladder . . . 1394
FBD/LD (Function block diagram/ladder) . . . 1395
ST (Structured text) . . . 1395

CHAPTER 40 LIST OF MOTION CONTROL FUNCTION BLOCK VARIABLES 1396
40.1 List of Variables . . . 1396
Axis variables . . . 1398
Axes group variables . . . 1423
System variables . . . 1430
Input axis variable . . . 1434
Output axis variable . . . 1436
System variable (Advanced synchronous control) . . . 1441
Other structure variable . . . 1442
Other structure variable (Advanced synchronous control) . . . 1462
40.2 List of Enumerators . . . 1474
ENUM enumerators . . . 1474

CHAPTER 41 ADMINISTRATIVE FUNCTION BLOCKS 1482
41.1 Axes Group Enabled . . . 1482
41.2 Axes Group Disabled . . . 1486
41.3 Operable . . . 1490
41.4 Current Position Change . . . 1494

41.5 Torque Limit Value 1502
41.6 Override Value Setting 1510
41.7 Parameter Read 1514
41.8 Parameter Write 1521
41.9 Axis Error Reset 1527
41.10 Axes Group Error Reset 1530
41.11 Touch Probe Enabled 1534
41.12 Touch Probe Disabled 1550
41.13 Cam Table Selection 1555
41.14 Current Value Change per Cycle 1562
41.15 All Axes Operation Available 1570
41.16 Axes Group Override Value Setting 1573
41.17 Motion Error Reset 1577
41.18 Control Operation Cycle Synchronization 1581
41.19 Advanced Synchronous Control One Cycle Current Position Calculation 1584
41.20 Cam Set Position Calculation in Advanced Synchronous Control 1587
41.21 Digital Cam Switch Output 1590
41.22 Axis Status Read of Single Axis 1598

CHAPTER 42 MOTION FUNCTION BLOCK 1601

42.1 Homing 1601
42.2 Forced Stop 1611
42.3 Group Forced Stop 1616
42.4 Absolute Value Positioning 1622
42.5 Relative Value Positioning 1635
42.6 JOG Operation 1644
42.7 Speed Control 1653
42.8 Torque Control 1661
42.9 Pressure Control 1672
42.10 Speed Control (Including Position Loop) 1682
42.11 Absolute Value Linear Interpolation Control 1690
42.12 Relative Value Linear Interpolation Control 1705
42.13 Absolute Value Circular Interpolation Control 1719
42.14 Relative Value Circular Interpolation Control 1736
42.15 Cam Operation Start 1754
42.16 Gear Operation Start 1787
42.17 Addition/Subtraction Positioning 1797
42.18 Backlash Compensation Filter 1808
42.19 Smoothing Filter 1821
42.20 Moving Direction Restriction Filter 1831
42.21 Speed Limit Filter 1840
42.22 Axis Control Start Wait 1848
42.23 Axes Group Control Start Wait 1855
42.24 Advanced Synchronous Control 1862
42.25 Multiple Axes Positioning Data Operation 1868
42.26 Absolute Position Tracking Control 1878
42.27 Motion Cyclic Position Control 1891
42.28 Motion Cyclic Velocity Control 1899
42.29 Motion Cyclic Torque Control 1903

CHAPTER 43 STANDARD FUNCTION BLOCK 1909
43.1 Profile Read 1909
43.2 Profile Write 1922

PART 9 NETWORK FUNCTION BLOCKS

CHAPTER 44 FUNCTION BLOCKS FOR Ethernet 1946
44.1 Opening/Disconnecting a Connection 1946
44.2 Socket Communications. 1952
44.3 SLMP. 1958

CHAPTER 45 FUNCTION BLOCKS FOR CC-Link IE TSN 1986
45.1 Remote Reset. 1986

APPENDICES 1991
Appendix 1 Instruction Processing Time. 1991
Time added to instruction processing time 1994
Appendix 2 Module Dedicated Instruction Availability. 1995
Appendix 3 Determining Three PID Constants 2006

INDEX 2008

INSTRUCTION INDEX 2010

REVISIONS. 2019
CONDITIONS OF USE FOR THE PRODUCT 2020
WARRANTY 2021
INFORMATION AND SERVICES 2022
TRADEMARKS 2022

RELEVANT MANUALS

The following manuals and guidelines are relevant to this product.

Manuals

Manual name [manual number]	Description
MELSEC MX Controller MX-R Model Programming Manual [SH-082644ENG] (this manual)	Program language specifications, controller instructions, standard functions/ function blocks, and specifications of motion control function blocks
MELSEC MX Controller MX-R Model User's Manual [SH-082641ENG]	Procedures before operation, specifications, devices, memory, functions, parameters, and troubleshooting of the controller
MELSEC iQ-R Programming Manual (Module Dedicated Instructions) [SH-081976ENG]	Dedicated instructions for the intelligent function modules
GX Works3 Operating Manual [SH-081215ENG]	System configuration, parameter settings, and online operations of GX Works3

Guidelines

Guideline name [guideline number]	Description
FA SYSTEM SECURITY GUIDELINE -SEPARATE VOLUME [MELSEC]- [BCN-P5999-1474]	Mitsubishi Electric approach to security for the MELSEC series and information for using the MELSEC series safely and securely in an FA system

TERMS

Unless otherwise specified, this manual uses the following terms.

Term	Description
Actual position	A position obtained by rounding a position address returned by a device with a ring counter. It is affected by a current position change.
Actual pressure	The actual value of the pressure
Actual velocity	A value obtained by converting the actual value received from the device on the real axis to the unit of the axis
Axes group variable	An AXES_GROUP type variable instance including parameters and data related to the axes group
Axis	A target to perform the motion control
Axis variable	An AXIS_* type variable instance including parameters and data related to the axis
Axis warning	An error or a warning related to an axis
Basic core	A core that executes a program
Buffer memory	A memory to store data such as setting values and monitored values.
Buffering	Standby status of a motion control FB when multiple start is performed
Buffering FB	A motion control FB that is in a pending status (Busy is TRUE) when multiple start is performed
Cam table	An operation profile data used for cam control (Ex.) Operation profile data of cam data format or operation profile data of cam data format for rotary cutter
CC-Link IE TSN Class	A group of devices and switching hubs compatible with CC-Link IE TSN, ranked according to the functions and performance by the CC-Link Partner Association. For CC-Link IE TSN Class, refer to the CC-Link IE TSN Installation Manual (BAP-C3007ENG-001) published by the CC-Link Partner Association.
CC-Link IE TSN function part	A part that performs CC-Link IE TSN communications and Ethernet communications on the CC-Link IE TSN port (P1, P2).
CC-Link IE TSN Protocol version 1.0	This protocol is used to perform communications by using the time sharing method defined by IEEE 802.1AS or IEEE 1588 for time synchronization.
CC-Link IE TSN Protocol version 2.0	This protocol is used to perform communications by using the time sharing method defined by IEEE 802.1AS and time-managed polling method for time synchronization. Supporting this protocol version mitigates restrictions when "Mixture of CC-Link IE TSN Class B/A or CC-Link IE TSN Class A Only" is set in the system configuration, and increases the number of connectable devices.
Command limit velocity	Set value for the limit velocity
Command pressure	Set value for the pressure
Commanded limit velocity	A specified value of the limit velocity
Commanded position	A command position address that is specified in the motion FB
Commanded pressure	A specified value of the pressure
Commanded velocity	Velocity that is specified in the motion FB
Continuous update	Continous reflection of input values to the control while the ContinuousUpdate input of the motion control FB is TRUE
Core	An operation circuit of the controller. The controller has multiple cores.
CPU function part	A part that performs sequence control and Ethernet communications on the Ethernet port (P3)
CPU Module Logging Configuration Tool	Software to configure data logging settings and to manage collected data
Cumulative current position	A command position address that is not rounded by the ring counter. It is affected by a current position change.
Cyclic data transfer processing	Processing of a cyclic transmission from its start to finish, performed by all the stations on a single network. The processing is performed asynchronously with the sequence scan of the CPU function part. The cyclic data transfer processing time varies depending on data volume and the number of transient transmission requests.
Dedicated instruction	An instruction that simplifies programming for using functions of intelligent function modules
Device	A memory of the controller to store data. Devices such as X, Y, M, and D are provided depending on the intended use.
Device station	A station other than a master station: a local station, a remote station
Disconnection	A process of stopping data link for relevant stations if a data link error occurs
Extension core	A core to which functions to be executed can be assigned
External signal high-accuracy input	An input signal linked with the signal detection time of the device station
FB instance	A function block that is inserted to a sequence program
First point of cam table	The meaning is different depending on the interpolation method specification for the cam data • Section interpolation or spline interpolation: Point of Table No.1 • Linear interpolation: Point of No.1

Term	Description
Forward-side position	Data to be sent to the drive unit as object data "Forward-side stop function (2E56H)". The limit position to which an axis can move in the pressure increase direction at feed/dwell, or the last position at pressure release.
General-purpose hub	A CC-Link IE TSN Class A switching hub authorized by CC-Link Partner Association
Global label	A label that is valid for all the program data when multiple program data are created in the project. There are two types of global label: a module specific label (module label), which is generated automatically by GX Works3, and an optional label, which can be created for any specified device.
Grandmaster	A source device or station to synchronize clocks in the time synchronization via PTP (Precision Time Protocol)
GX LogViewer	Software to display data collected by data logging
GX Works3	The product name of the software package for the MELSEC programmable controllers
Input variable	An input argument of FB
Intelligent function module	A module that has functions other than input and output, such as an A/D converter module and D/A converter module
iQSS	iQ Sensor Solution. Cooperation of sensors, programmable controllers, HMI (Human Machine Interface), and engineering tool to facilitate the start-up or maintenance of sensors. (📖iQ Sensor Solution Reference Manual)
Jerk	Chronological change rate of acceleration or deceleration
Label	A variable consisting of a specified string used in I/O data or internal processing
LB	A link relay of a link device. Bit data sent from each station of the network.
Link device	A device (RX, RY, RWr, RWw, LB, LW) in the CC-Link IE TSN function part or a network module
Local station	A station that performs cyclic transmission and transient transmission with the master station and other local stations
LW	A link register of a link device. Word data (16-bit data) sent from each station of the network.
Machine feed value	A command position address that is not rounded by the ring counter. It is based on the home position, and it is not affected by a current position change.
Master axis (input) absolute coordinate	Another term for the Master axis absolute coordinate (MasterAbsolute) for input variables of MC_CamTableSelect (Cam Table Selection)
Master station	A station that controls the entire network. This station can perform cyclic transmission and transient transmission with all stations. Only one master station can be used in a network.
Module label	A label that represents one of memory areas (I/O signals and buffer memory areas) specific to each module in a given string. For the module used, GX Works3 automatically generates this label, which can be used as a global label.
Motion control FB	An FB that is related to motion control. Its name starts with MC_ or MCv_.
Motion FB	A motion control FB that takes an axis or an axes group for the argument and changes the axis status or the axes group status when executed
Motion function part	A part of the controller used for motion control
MR Configurator2	A product name of the servo setup software
Multicast filter	A filter function that selects whether to send cyclic data of multicast mode received by the own station to the subsequent stations. Setting parameters for this function are not required because the master station automatically sets the parameters according to the system configuration.
Multicast mode	A communication mode used to send cyclic data to multiple stations
Multiple axes interpolation control	Control in which multiple axes work together such as linear interpolation and circular interpolation. Axes to be cooperated are specified in an axes group.
Multiple start	Executing another motion control FB to the same axis to which the other motion control FB is being executed
Object	Data of a device compatible with CANopen
Output variable	An output argument of FB
POU	A unit that configures a program. Units are categorized and provided in accordance with functions. A program that is composed of POUs (program organization units) allows the lower-layer processing, when the program is multi-layered, to be divided into several units by processing and function, enabling the creation of programs based on each unit.
Predefined protocol support function	A function of GX Works3. This function sets protocols appropriate to each external device and reads/writes protocol setting data.
Program block	A group of POUs that configure a program
Real axis	An axis that is linked with a device on the network
Real drive axis	An axis that is linked with a CC-Link IE TSN compatible device that supports csp/csv/cst mode (sequential command) of the CiA402 drive profile
Real encoder axis	An axis that generates a commanded position from the current position of the encoder connected with a device. It is used as the master axis of the single axis synchronous control.
Reconnection	A process of restarting data link for relevant stations when the faulty station recovers from an error

Term	Description
Relay station	A station that relays data link to other station with mounting more than one network modules on one programmable controller.
Remote station	A station that exchanges I/O signals (bit data) and I/O data (word data) with another station by cyclic transmission. This station can perform transient transmission.
Reserved address	An IP address reserved for special purposes, defined by RFC 6890. This IP address cannot be used when the programmable controller is connected via the global IP network.
Routing	A process of selecting paths for communication with other networks. There are two types of routing: dynamic routing that auto-selects the communication routes, and static routing where communication routes are arbitrarily set.
RWr	A remote register of the link device. Word data (16-bit data) input from a device station to the master station. (For some areas in a local station, data are input in the opposite direction.)
RWw	A remote register of the link device. Word data (16-bit data) output from the master station to a device station. (For some areas in a local station, data are output in the opposite direction.)
RX	Remote input of the link device. Bit data input from a device station to the master station. (For some areas in a local station, data are input in the opposite direction.)
RY	Remote output of the link device. Bit data output from the master station to a device station. (For some areas in a local station, data are output in the opposite direction.)
SB	An abbreviation for a link special relay. Bit data that indicates the operating status and data link status of a module on CC-Link IE.
Set position	A commanded position address rounded by the ring counter. It is affected by a current position change.
Set velocity	The current control value that is generated by motion operation
Signal flow	The execution status that the last time an operation of a program or an FB is executed in each step.
Single axis synchronization	A control to output the position information (command) of one Slave (slave axis) that is synchronized with Master (master axis)
Slave axis (output) absolute coordinate	Another term for the Slave axis absolute coordinate (SlaveAbsolute) for input variables of MC_CamTableSelect (Cam Table Selection)
SLMP	A SeamLess Message Protocol. This protocol is used to access an SLMP-compatible device or a programmable controller connected to an SLMP-compatible device from an external device.
Soft LED	An LED that is not installed as hardware and is controlled by software This can be checked with module diagnostics.
SW	An abbreviation for a link special register. Word data that indicates the operating status and data link status of a module on CC-Link IE.
Switching speed	• The set velocity of the motion control FB being executed when Aborting is specified • The command velocity when the motion control FB being executed reached the target position when Blending is specified
System error	An error or a warning not related to an axis. When Axis information (Axis) and Axes Group Information (AxesGroup) of the motion control FBs are not correct, the system error will occur.
Target position	A command position address that is specified in the motion FB
Transient transmission group number	Number that is assigned for transient transmission to any given stations. By specifying a group of stations as transient transmission target, data can be sent to the stations of the same group No.
TSN hub	A CC-Link IE TSN Class B switching hub authorized by CC-Link Partner Association
Unicast mode	A communication mode used to send cyclic data to one station
Virtual axis	An axis that is not linked with a device on the network
Virtual drive axis	A virtual axis that can generate a command virtually
Virtual encoder axis	An axis that generates a commanded position from a variable. It is used as the master axis of the single axis synchronous control.
Virtual linked axis	An axis that connects FBs and transmits commands in the single axis synchronous control

GENERIC TERMS AND ABBREVIATIONS

Unless otherwise specified, this manual uses the following generic terms and abbreviations.

Generic term/abbreviation	Description
Base unit	A generic term for main base units and extension base units
CC-Link IE	A generic term for the following items: • CC-Link IE TSN • CC-Link IE Controller Network (MELSEC iQ-R CC-Link IE Controller Network User's Manual (Application)) • CC-Link IE Field Network (MELSEC iQ-R CC-Link IE Field Network User's Manual (Application))
CC-Link IE Controller Network-equipped module	An RJ71GP21-SX CC-Link IE Controller Network module, an RJ71GP21S-SX CC-Link IE Controller Network module, and the following modules when the CC-Link IE Controller Network function is used: • RJ71EN71
CC-Link IE TSN master/local module	RJ71GN11-T2, RJ71GN11-SX, RJ71GN11-EIP (CC-Link IE TSN part)
CC-Link IE TSN-equipped module	A generic term for the RJ71GN11-T2, RJ71GN11-SX, and RJ71GN11-EIP
Controller	A generic term for the MXR300-16, MXR300-32, MXR300-64, MXR500-128, and MXR500-256
csp	An abbreviation for cyclic synchronous position mode (one of the control modes on the drive unit side)
cspr	An abbreviation for cyclic synchronous pressure mode (one of the control modes on the drive unit side)
cst	An abbreviation for cyclic synchronous torque mode (one of the control modes on the drive unit side)
csv	An abbreviation for cyclic synchronous velocity mode (one of the control modes on the drive unit side)
ct	An abbreviation for continuous operation to torque control mode (one of the control modes on the drive unit side)
Data link	A generic term for cyclic transmission and transient transmission
Device supporting iQSS	A device which supports iQ Sensor Solution. iQ Sensor Solution Reference Manual
Drive unit	A generic term for motor drive devices such as a servo amplifier
Engineering tool	A tool used for setting up controllers or servo amplifiers, programming, debugging, and maintenance
Ethernet device	A generic term for the devices supporting IP communication (such as a personal computer, a vision sensor, and a bar code reader)
Ethernet-equipped module	A generic term for the following modules: • Controller (when the Ethernet function is used) • RJ71EN71 (when the Ethernet function is used)
GOT	A generic term for Mitsubishi Electric Graphic Operation Terminal GOT1000 and GOT2000 series
hm	An abbreviation for homing mode (one of the control modes on the drive unit side)
I/O module	An input module, an output module, an I/O combined module, and an interrupt module
ICMP	An abbreviation for Internet Control Message Protocol. This protocol is used to exchange messages of errors in an IP network or other information related to an Ethernet network.
MCFB	An abbreviation for motion control FB
MR-J5(W)-G	A generic term for the MR-J5-_G/MR-J5-_G_-RJ/MR-J5-_G_-LL/MR-J5W_-_G/MR-J5D_-_G_ model servo amplifiers
Operation profile data	A generic term for waveform data used for each control
PDO	An abbreviation for Process Data Object. Aggregation of application objects transferred periodically between multiple CANopen nodes.
Power supply module	A MELSEC iQ-R series power supply module
PTP	An abbreviation for Precision Time Protocol. A predefined protocol for time synchronization between devices on a network.
RJ71GN11	A generic term for the RJ71GN11-T2 and RJ71GN11-SX
SDO	An abbreviation for Service Data Object. A message to access an object entry in the object dictionary of a CANopen node. Data is sent/received between the stations aperiodically.
SLMP	An abbreviation for Seamless Message Protocol. This protocol is used to access an SLMP-compatible device from an external device (such as a personal computer or HMI (Human Machine Interface)) or an SLMP-compatible module (such as the Ethernet-equipped module or modules on CC-Link IE TSN).
Station sub ID number	An ID number of a sensor connected to a CC-Link-compatible communication module

PART 1 PROGRAMMING OVERVIEW

This part consists of the following chapters.

1 OVERVIEW

2 PROGRAM ORGANIZATION UNITS

3 LADDER DIAGRAM

4 STRUCTURED TEXT LANGUAGE

5 FBD/LD

1 OVERVIEW

1.1 Instruction Configuration

Many instructions available for controllers are each divided into the instruction part and operand part.

The instruction part and operand part are used as follows.

- Instruction part: Indicates the function of the relevant instruction.
- Operand part: Indicates the data used for the instruction.

The operand part is further classified to source data, destination data, and numerical data.

Source (s)

Source is the data used in the operation.

Depending on the label or device specified in each instruction, the source becomes as follows.

Type	Description
Constant	The constant specifies a numerical value used in the operation. It is set during program creation and cannot be changed during program execution. When using constants in variable data, perform index modification.*1
Device Label	The user specifies the device or label where the data to be used in the operation is stored. Necessary data must be thus stored in the specified device or label before operation execution. By changing the data to be stored in the specified device or label during program execution, the data to be used by the instruction can be changed.

*1 For the index modification, refer to the following.

User's manual for the controller used

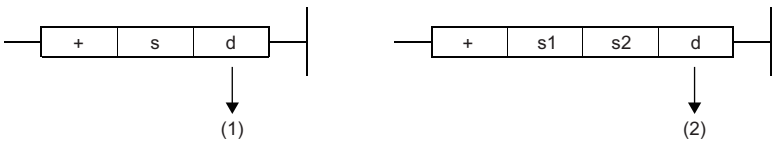
Destination (d)

Data after operation is stored in the destination area.

However, some instructions require the data to be used in the operation to be stored before the operation.

Ex.

Binary 16-bit data addition instruction



(1) The data required for operation is stored before the operation.

(2) Only the operation result is stored.

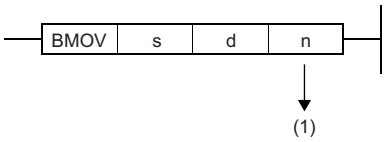
A label or device to store data must be set for the destination.

Numerical value (n)

For the numerical values of the numbers of devices, transfers, data, and strings, specify those used by an instruction which uses multiple devices or an instruction which specifies the numbers of repetitions, data to be processed, and strings.

Ex.

Block transfer instruction



(1) The number of transfers executed by the BMOV instruction is specified.

A numerical value from 0 to 65535 or 0 to 4294967295 can be set for the size such as the number of devices, transfers, or characters. *1

Note, however, that when the size specification such as the number of devices, transfers, or characters is 0, the relevant instruction results in non-processing.

The upper limit of the numerical value may be less than the values mentioned above, depending on the capacity of the device memory or file storage area.*2

*1 The setting range varies depending on the instruction. For details, refer to the description of each instruction.

*2 The capacities of the device memory and file storage area vary depending on models. For details, refer to the following.
User's manual for the controller used

Point

Be careful when a large numerical value is used such as for the number of transfers. It delays the scan time.

1.2 Data Specification Method

The following table lists the types of data that can be used for instructions in controllers.

Data	Classification
Bit data	Bit data
16-bit data (word data)	16-bit signed binary data
	16-bit unsigned binary data
32-bit data (double word data)	32-bit signed binary data
	32-bit unsigned binary data
Real number data (floating-point data)	Single-precision real number data
	Double-precision real number data
BCD data	BCD 4-digit data
	BCD 8-digit data
	BCD 16-digit data
String data	String
	Unicode string

Device data

The following table lists devices and constants that can be used to specify the setting data of instructions.

Data type	Description	Specifiable device/constant*1
Bit	Bit data can be handled. ☞ Page 31 Bit data	• Bit device • Bit specification of word device
Word	Word data can be handled. ☞ Page 32 16-bit data (word data)	• Word device • Digit-specified bit device (K1 to K4)*2 • Decimal constant • Hexadecimal constant
16-bit signed binary	16-bit data can be handled. The value range varies depending on whether the value is signed or unsigned. ☞ Page 32 16-bit data (word data)	
16-bit unsigned binary		
Double word	Double-word data can be handled. ☞ Page 35 32-bit data (double word data)	• Word device • Double-word device • Digit-specified bit device (K1 to K8)*2 • Decimal constant • Hexadecimal constant
32-bit signed binary	Two consecutive sets of 32-bit data or 16-bit data can be handled. The value range varies depending on whether the value is signed or unsigned. ☞ Page 35 32-bit data (double word data)	
32-bit unsigned binary		
BCD 4-digit	BCD 4-digit data can be handled. 16-bit data is divided by 4 digits and each digit is specified in 0 to 9.	• Word device • Digit-specified bit device (K1 to K4)*2 • Decimal constant • Hexadecimal constant
BCD 8-digit	BCD 8-digit data can be handled. 32-bit data is divided by 8 digits and each digit is specified in 0 to 9.	• Word device • Double-word device • Digit-specified bit device (K1 to K8)*2 • Decimal constant • Hexadecimal constant
Single-precision real number	Single-precision real number data (single-precision floating-point data) can be handled. ☞ Page 38 Configuration of single-precision real number data	• Word device • Double-word device • Real constant
Double-precision real number	Double-precision real number data (double-precision floating-point data) can be handled. ☞ Page 39 Configuration of double-precision real number data	• Word device • Double-word device • Real constant
String	ASCII code and Shift JIS code string data can be handled. ☞ Page 41 String data	• Word device • String constant
Unicode string	Unicode string data can be handled. ☞ Page 41 String data	• Word device • String constant
Device name	A device can be specified directly.	• Device name corresponding to applicable device

*1 A constant can be used in the data specified for the source (s) or numerical data (n) by an instruction.
*2 For the specification method, refer to the detail page of each data type.

Label data

The following table lists labels that can be used to specify the setting data of instructions.

■Primitive data type

Data type (label)	Specifiable label
Bit (BOOL)	• Bit type label • Bit-specified word [unsigned]/bit string [16 bits] type label • Bit-specified word [signed] type label • Timer/retentive timer/long timer/long retentive timer type label contact/coil • Counter/ long counter type label contact/coil
Word [unsigned]/bit string [16 bits] (WORD)	• Word [unsigned]/bit string [16 bits] type label • Digit-specified bit type label (K1 to K4) • Current value of timer/retentive timer type label • Current value of counter type label
Double word [unsigned]/bit string [32 bits] (DWORD)	• Double word [unsigned]/bit string [32 bits] type label • Digit-specified bit type label (K1 to K8) • Current value of long timer/long retentive timer type label • Current value of long counter type label
Word [signed] (INT)	• Word [signed] type label • Digit-specified bit type label (K1 to K4) • Current value of timer/retentive timer type label • Current value of counter type label
Double word [signed] (DINT)	• Double word [signed] type label • Digit-specified bit type label (K1 to K8) • Current value of long timer/long retentive timer type label • Current value of long counter type label
Single-precision real number (REAL)	• Single-precision real data type label
Double-precision real number (LREAL)	• Double-precision real data type label
Time (TIME)	• Time type label
String (STRING)	• String type label
String [Unicode] (WSTRING)	• String [Unicode] type label
Pointer (POINTER)	• Pointer type label

Point

For details on individual labels, refer to the following.

User's manual for the controller used

■Generic data type

The generic data type is the data type of the labels which summarize several primitive data types.

Generic data types are used when multiple data types are allowed for arguments and return values of functions or function blocks.

Labels defined in generic data types can be used in any sub-level data type.

Data type (label)							Specifiable data type
ANY*1	ANY_ELEMENTARY	ANY_BIT				ANY_BOOL	Bit
						ANY_BITADDR*1	Bit
						ANY16_U	Word [unsigned]/bit string [16 bits]
						ANY32_U	Double word [unsigned]/bit string [32 bits]
		ANY_WORDADDR	ANY_NUM	ANY_INT	ANY16	ANY16_S	Word [signed]
						ANY16_U	Word [unsigned]/bit string [16 bits]
					ANY32	ANY32_S	Double word [signed], time
						ANY32_U	Double word [unsigned]/bit string [32 bits]
				ANY_REAL		ANYREAL_32	Single-precision real number
						ANYREAL_64	Double-precision real number
			ANY_STRING			ANYSTRING_SINGLE	String
						ANYSTRING_DOUBLE	String [Unicode]
			ANY16_OR_STRING_SINGLE			ANY16_S	Word [signed]
						ANY16_U	Word [unsigned]/bit string [16 bits]
						ANYSTRING_SINGLE	String
			ANY_DT				Word [signed], word [unsigned]/bit string [16 bits]
			ANY_TM				Word [signed], word [unsigned]/bit string [16 bits]
	ANY_STRUCT*1						Structures
	STRUCT						Structures

*1 Can also be used as an array.

■Generic data type (array)

For the following generic data type, define the number of array elements.

Data type (label)			Specifiable data type
ANYBIT_ARRAY			Bit array
ANYWORD_ARRAY	ANY16_ARRAY	ANY16_S_ARRAY	Word [signed] array
		ANY16_U_ARRAY	Word [unsigned]/bit string [16 bits] array
	ANY32_ARRAY	ANY32_S_ARRAY	Double word [signed] array, time array
		ANY32_U_ARRAY	Double word [unsigned]/bit string [32 bits] array
	ANY_REAL_ARRAY	ANY_REAL_32_ARRAY	Single-precision real number array
		ANY_REAL_64_ARRAY	Double-precision real number array
	ANY_STRING_ARRAY	ANY_STRING_SINGLE_ARRAY	String array
		ANY_STRING_DOUBLE_ARRAY	String [Unicode] array
STRUCT_ARRAY			Structure array

Bit data

Data size and data range

Bit data is handled in increments of bits such as contacts and coils.

Data name	Data size	Value range
Bit data	1 bit	0, 1

Handling bit data with bit devices and labels

One point of bit device/label can handle 1-bit data.

Handling bit data with bit word devices

By specifying a bit number for a word device, bit data of the specified bit number can be handled.

A bit in a word device can be specified by "Word device number.Bit number".

A bit number can be specified in hexadecimal in the range from 0 to F.

For example, bit 5 (b5) of D0 is specified as D0.5, and bit 10 (b10) of D0 is specified as D0.A.

The following word devices support bit specification.

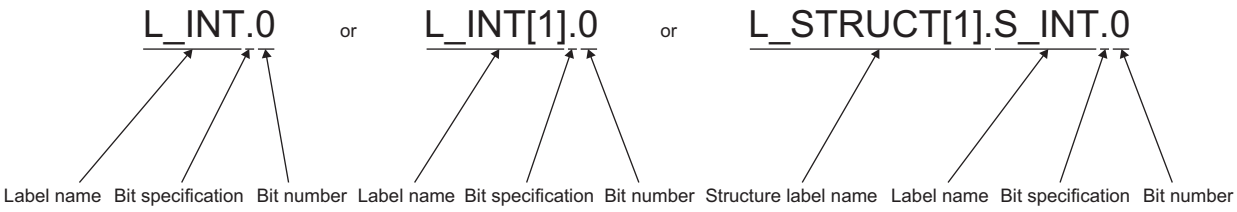
Item	Device
Word devices which support bit specification	• Data register (D) • Link register (W, J□\W) • Link special register (SW, J□\SW) • Function register (FD) • Special register (SD) • Module access device (U□\G) • CPU Buffer memory access device (U3E□\G) • File register (R, ZR) • Module refresh register (RD)

Handling bit data with word type labels

By specifying a bit number for a word type label, bit data of the specified bit number can be handled.

A bit in a word type label can be specified by "Label name.Bit number".

Ex.



The following data types of labels support bit specification.

Item	Data type
Data types of labels which support bit specification.	• Word [signed] (INT type) • Word [unsigned]/bit string [16 bits] (WORD type) • Current value (N) of timer (TIMER type)*1 • Current value (N) of retentive timer (RETENTIVETIMER type)*1 • Current value (N) of counter (COUNTER type)*1

*1 Cannot be specified in ladder programs.

16-bit data (word data)

Data size and data range

16-bit data includes signed and unsigned 16-bit data.

In signed 16-bit data, a negative number is represented in two's complement.

Data name	Data size	Value range	
		Decimal notation	Hexadecimal notation
Signed 16-bit data	16 bits (1 word)	-32768 to 32767	0000H to FFFFH
Unsigned 16-bit data		0 to 65535	

Handling 16-bit data with bit devices

A bit device can be handled as 16-bit data by performing digit specification.

Item		Notation	Example
Bit device	Other than link direct device	K□Bit device start number □: Number of digits (Specify the number within the range of 1 to 4.)	K4X40 K2M113
	Link direct device	J□\K□Bit device start number □ (on the left): Network number □ (on the right): Number of digits (Specify the number within the range of 1 to 4.)	J1\K3B10 J10\K2Y40

Handling 16-bit data with bit type array labels

A bit type array label can be handled as 16-bit data by performing digit specification.

The following table shows the notation for handling a bit type array label as 16-bit data by digit specification.

Item	Notation	Example
Bit type array label	K□Label name □: Number of digits (Specify the number within the range of 1 to 4.) Specify a label name without an array element.	K1L_BOOL

Digit specification range

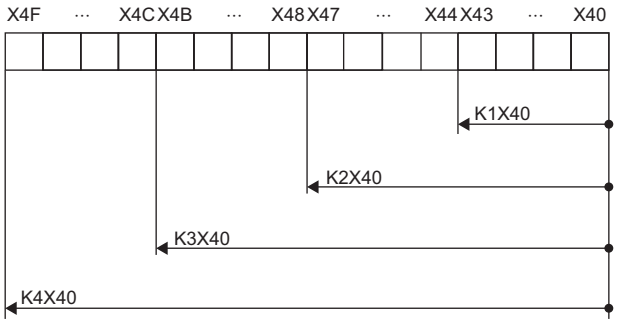
The following table lists the range of 16-bit data for each digit specification.

Digit specification	Decimal notation	Hexadecimal notation
K1	0 to 15	0H to FH
K2	0 to 255	00H to FFH
K3	0 to 4095	000H to FFFH
K4	Signed 16-bit data: -32768 to 32767 Unsigned 16-bit data: 0 to 65535	0000H to FFFFH

Ex.

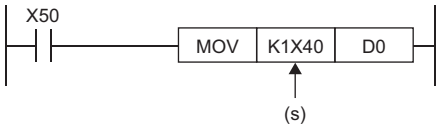
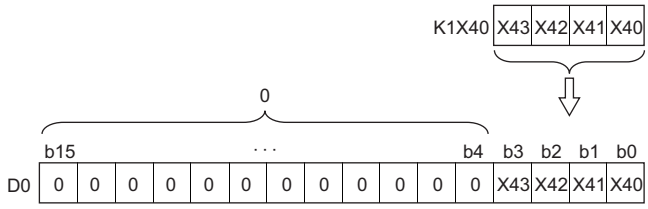
When digit specification is made for X40, the applicable number of points is as follows.

- K1X40→4 points from X40 to X43
- K2X40→8 points from X40 to X47
- K3X40→12 points from X40 to X4B
- K4X40→16 points from X40 to X4F



■Specifying a bit device with digit specification in the source (s)

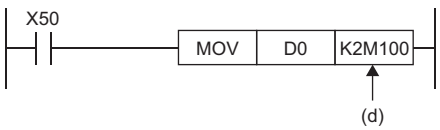
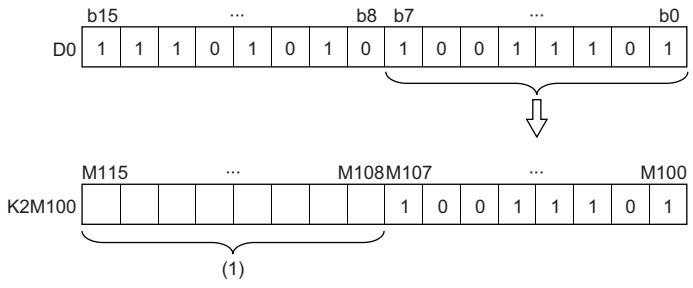
When a bit device is specified with digit specification in the source of an instruction, 0 is stored in the word device of the destination, in the upper bits than those specified in the source of the instruction.

Ladder example	Processing
• 16-bit data instruction  	

■Specifying a bit device with digit specification in the destination (d)

When a digit specification is made in the destination of an instruction, the number of points by the digit specification is applicable in the destination.

The upper bit devices than the number of points specified by digits remain unchanged.

Ladder example	Processing
• When the source data is a word device  	 (1) The data remain the same.

Handling 16-bit data with word devices/labels

■Word device

One point of word device can handle 16-bit data.

■Word type label

One point of word type label can handle 16-bit data.

32-bit data (double word data)

Data size and data range

32-bit data includes signed and unsigned 32-bit data.

In signed 32-bit data, a negative number is represented in two's complement.

Data name	Data size	Value range	
		Decimal notation	Hexadecimal notation
Signed 32-bit data	32 bits (2 word)	-2147483648 to 2147483647	00000000H to FFFFFFFFH
Unsigned 32-bit data		0 to 4294967295	

Handling 32-bit data with bit devices

A bit device can be handled as 32-bit data by performing digit specification.

Item		Notation	Example
Bit device	Other than link direct device	K□ Bit device start number Number of digits: Specify the number within the range of 1 to 8.	K8X80 K6B018
	Link direct device	J□\K□ Bit device start number Number of digits: Specify the number within the range of 1 to 8. Network number	J1\K7B30 J10\K5X128

Handling 32-bit data with bit type array labels

A bit type array label can be handled as 32-bit data by performing digit specification.

The following table shows the notation for handling a bit type array label as 32-bit data by digit specification.

Item	Notation	Example
Bit type array label	K□ Label name Number of digits: Specify the number within the range of 1 to 8. Specify a label name without an array element. For languages other than ladder, specify the number within the range of K5 to K8. (For languages other than ladder, the label with digit specification within the range of K1 to K4 is handled as ANY16.)	K8L_BOOL

Digit specification range

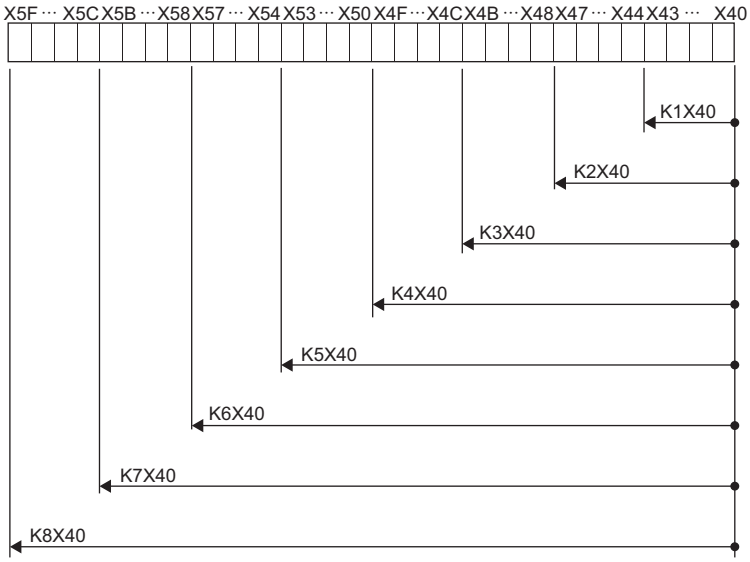
The following table lists the range of 32-bit data for each digit specification.

Digit specification	Decimal notation	Hexadecimal notation
K1	0 to 15	0H to FH
K2	0 to 255	00H to FFH
K3	0 to 4095	000H to FFFH
K4	0 to 65535	0000H to FFFFH
K5	0 to 1048575	00000H to FFFFFH
K6	0 to 16777215	000000H to FFFFFFH
K7	0 to 268435455	0000000H to FFFFFFFH
K8	Signed 32-bit data: -2147483648 to 2147483647 Unsigned 32-bit data: 0 to 4294967295	00000000H to FFFFFFFFH

Ex.

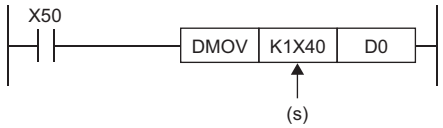
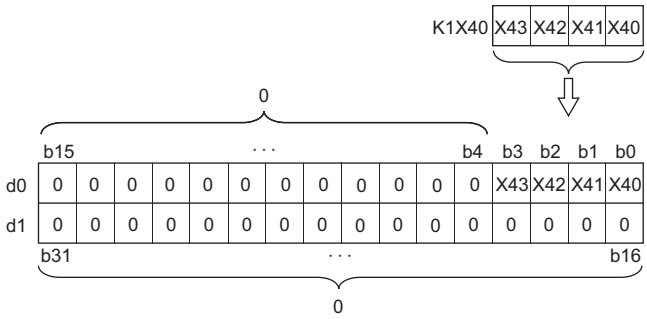
When digit specification is made for X40, the applicable number of points is as follows.

- K1X40→4 points from X40 to X43
- K2X40→8 points from X40 to X47
- K3X40→12 points from X40 to X4B
- K4X40→16 points from X40 to X4F
- K5X40→20 points from X40 to X53
- K6X40→24 points from X40 to X57
- K7X40→28 points from X40 to X5B
- K8X40→32 points from X40 to X5F



■Specifying a bit device with digit specification in the source (s)

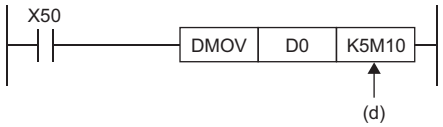
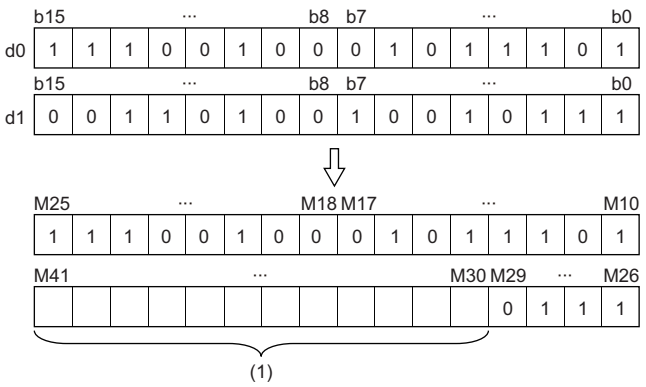
When a bit device is specified with digit specification in the source of an instruction, 0 is stored in the word device of the destination, in the upper bits than those specified in the source of the instruction.

Ladder example	Processing
• 32-bit data instruction 	

■Specifying a bit device with digit specification in the destination (d)

When a digit specification is made in the destination of an instruction, the number of points by the digit specification is applicable in the destination.

The upper bit devices than the number of points specified by digits remain unchanged.

Ladder example	Processing
• When the source data is a word device 	 (1) The data remain the same.

Handling 32-bit data with word devices/labels

■Word device

Two points of word device can handle 32-bit data.

Note, however, that one point of the following devices can handle 32-bit data.

- Long timer (LT)
- Long retentive timer (LST)
- Long counter (LC)
- Long index register (LZ)

■Double word type label

One point of double word device can handle 32-bit data.

Real number data (floating-point data)

Data size and data range

Real number data includes single-precision 32-bit real number data and double-precision 64-bit real number data.
Real number data can be stored only in devices other than bit devices or in single-precision or double-precision real data type labels.

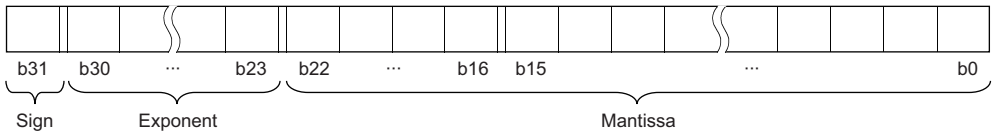
Data name		Data size	Value range
Single-precision real number data (single-precision floating-point data)	Positive number	32 bits (2 word)	$2^{-126} \leq$ real number $< 2^{128}$
	Zero		0
	Negative number		$-2^{128} <$ real number $\leq -2^{-126}$
Double-precision real number data (double-precision floating-point data)	Positive number	64 bits (4 word)	$2^{-1022} \leq$ real number $< 2^{1024}$
	Zero		0
	Negative number		$-2^{1024} <$ real number $\leq -2^{-1022}$

Configuration of single-precision real number data

Single-precision real number data consists of a sign, mantissa, and exponent, and is expressed as shown below.



The following figure shows the bit configuration of the internal expression of single-precision real number data and the meaning of each part.



■Sign (1 bit)

This bit represents the positive or negative sign of a numerical value. "0" indicates a positive number or 0. "1" Indicates a negative number.

■Mantissa (23 bits)

A mantissa means XXXXX··· of $1.XXXXX\cdots \times 2^N$ representing a single-precision real number in binary.

■Exponent (8 bits)

An exponent means N of $1.XXXXX\cdots \times 2^N$ representing a single-precision real number in binary. The following table shows the relationships between the exponent value and N of a single-precision real number.

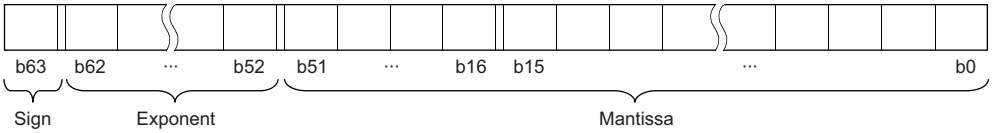
Exponent (b24 to b30)	FFH	FEH	FDH	···	81H	80H	7FH	7EH	···	02H	01H	00H
N	Not used	127	126	···	2	1	0	-1	···	-125	-126	Not used

Configuration of double-precision real number data

Double-precision real number data consists of a sign, mantissa, and exponent, and is expressed as shown below.



The following figure shows the bit configuration of the internal expression of double-precision real number data and the meaning of each part.



■Sign (1 bit)

This bit represents the positive or negative sign of a numerical value. "0" indicates a positive number or 0. "1" Indicates a negative number.

■Mantissa (52 bits)

A mantissa means XXXXX··· of 1.XXXXX···$\times 2^N$ representing a single-precision real number in binary.

■Exponent (11 bits)

An exponent means N of 1.XXXXX···$\times 2^N$ representing a single-precision real number in binary. The following table shows the relationships between the exponent value and N of a single-precision real number.

Exponent (b52 to b62)	7FFH	7FEH	7FDH	···	401H	400H	3FFH	3FEH	···	02H	01H	00H
N	Not used	1023	1022	···	2	1	0	-1	···	-1021	-1022	Not used

Precautions

■When setting an input value of single-precision real number from the engineering tool

The number of significant digits is about 7 because the engineering tool processes single precision real number data in 32-bit single precision.

When the input value of single-precision real number data exceeds 7 digits, the 8th digit is rounded off.

Therefore, if the rounded-off value goes out of the range from -2147483648 to 2147483647, it will not be an intended value.

Ex.

When "2147483647" is set as an input value, it is handled as "2147484000" because 8th digit "6" is rounded off.

Ex.

When "E1.1754943562" is set as an input value, it is handled as "E1.175494" because 8th digit "3" is rounded off.

Set an input value within the following range. If the set value is out of the following range, a conversion error occurs.

Decimal point expression: 0.0000000001 ≤ Absolute value of real number data ≤ 999999900000.0

Exponential notation: 1.175494351E-38 ≤ Absolute value of real number data ≤ 3.402823466E+38

■When setting an input value of double-precision real number from the engineering tool

The number of significant digits is about 15 because the engineering tool processes double precision real number data in 64-bit double precision.

When the input value of double-precision real number data exceeds 15 digits, the 16th digit is rounded off.

Therefore, if the rounded-off value goes out of the range from -2147483648 to 2147483647, it will not be an intended value.

Ex.

When "2147483646.12345678" is set as an input value, it is handled as "2147483646.12346" because 16th digit "6" is rounded off.

Ex.

When "E1.7976931348623157+307" is set as an input value, it is handled as "E1.79769313486232+307" because 16th digit "5" is rounded off.

Set an input value within the following range. If the set value is out of the following range, a conversion error occurs.

Decimal point expression: 0.00000000000000000001 ≤ Absolute value of real number data ≤ 9999999999999999000.0

Exponential notation: 2.22507385850721E-308 ≤ Absolute value of real number data ≤ 1.79769313486231E+308

Point

The monitor function of the engineering tool can monitor real number data of controllers.

To represent "0" in real number data, set all numbers in each of the following range to 0.

- Single-precision real number data: b0 to b31
- Double-precision real number data: b0 to b63

The setting range of real number data is as follows. *1

- Single-precision real number data: -2^{128}<[single-precision real number data]≤-2^{-126}, 0, 2^{-126}≤[single-precision real number data]<2^{128}
- Double-precision real number data: -2^{1024}<[double-precision real number data]≤-2^{-1022}, 0, 2^{-1022}≤[double-precision real number data]<2^{1024}

Do not specify "-0" (only the most significant bit is 1) in real number data. Performing a real number operation using -0 results in an operation error.

*1 For the operations to be performed when an overflow or underflow occurs or when a special value is input, refer to the following.
User's manual for the controller used

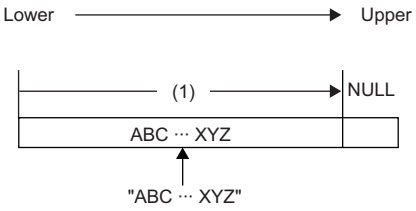
String data

Format of string data

The following table lists the types of string data, each of which ends with a NULL code to be handled as a string.

Type	Character code	Last character
String	ASCII code, Shift JIS code	NULL(00H)
Unicode string	Unicode (UTF-16 (little endian))	NULL(0000H)

String data is stored in devices or an array in ascending order of device numbers or array element numbers.



(1) Character code string

Notation of string

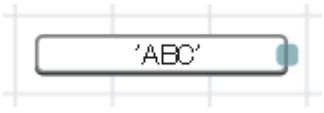
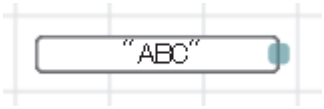
The following shows the notation of strings in ladder programs.

Data type		Notation	Example
String	STRING	Enclose a string (ASCII code, Shift JIS code) and Unicode string in double quotation marks (").	"ABC"
String [Unicode]	WSTRING		

The following shows the notation of strings in ST programs.

Data type		Notation	Example
String	STRING	Enclose a string (ASCII code, Shift JIS code) in single quotation marks (').	'ABC'
String [Unicode]	WSTRING	Enclose a Unicode string in double quotation marks (").	"ABC"

The following shows the notation of strings in FBD/LD programs.

Data type		Notation	Example
String	STRING	Enclose a string (ASCII code, Shift JIS code) in single quotation marks (').	
String [Unicode]	WSTRING	Enclose a Unicode string in double quotation marks (").	

Data range

The following table summarizes the ranges of string data.

Type	Maximum number of characters that can be set in a label	Maximum number of characters that can be used for string constant
String	255 single-byte characters (excluding the last NULL character)	255 single-byte characters (excluding the last NULL character)
Unicode string*1	255 characters (excluding the last NULL character)	255 characters (excluding the last NULL character)

*1 For the Unicode string, characters up to the basic multilingual plane can be used.

Number of words required for storing data

String data can be stored in word devices.

The following table lists the numbers of words required for storing string data.

Number of string bytes	Number of words required for storing strings	Number of words required for storing Unicode strings
0 byte	1 [word]	1 [word]
Odd number of bytes	(Number of string bytes+1)÷ 2 [words]	— (because one character is an even number of bytes)
Even number of bytes	(Number of string bytes÷2) +1 [words]	Number of characters+1 [words]

String data storage location

An image of the string data storage location is shown below.

■Strings

In each string storage image, "NULL" indicates a NULL code (00H).

String to be stored	Image of storing string data from D0	Image of storing string data from word type label array arrayA[0]
Null string (("") or (''))	D0: NULL | NULL	arrayA[0]: NULL | NULL
ABC	D0: B | A D1: NULL | C	arrayA[0]: B | A arrayA[1]: NULL | C
ABCD	D0: B | A D1: D | C D2: NULL | NULL	arrayA[0]: B | A arrayA[1]: D | C arrayA[2]: NULL | NULL

■Unicode strings

In each Unicode string storage image, "NULL" indicates a NULL code (0000H).

String to be stored	Image of storing string data from D0	Image of storing string data from word type label array arrayA[0]
Null string ("")	D0: NULL	arrayA[0]: NULL
ABCD	D0: A D1: B D2: C D3: D D4: NULL	arrayA[0]: A arrayA[1]: B arrayA[2]: C arrayA[3]: D arrayA[4]: NULL

1.3 Execution Condition

Types of execution conditions

The following table lists the execution conditions of instructions.

Execution condition		Description*1
On		An instruction is executed during on. It is executed only while the precondition of the instruction is on. When the precondition is off, the instruction is not executed.
Rising edge		An instruction is executed one time when turned on. It is executed only once on the rising edge (off to on) of the precondition of the instruction and is no longer executed later even when the condition turns on.
Off		An instruction is executed during off. It is executed only while the precondition of the instruction is off. When the precondition is on, the instruction is not executed.
Falling edge		An instruction is executed one time when turned off. It is executed only once on the falling edge (on to off) of the precondition of the instruction and is no longer executed later even when the condition turns off.
Every scan	—	An instruction is always executed regardless of whether the precondition of the instruction is on or off. When the precondition is off, the instruction performs off processing.

*1 When the program is described in structured text language (ST) or function block diagram/ladder diagram (FBD/LD), EN will be the precondition of the instruction.

Execution condition of each instruction

The execution condition varies depending on the instruction. For execution condition, refer to the details of each instruction in this manual.

When the program is described in structured text language (ST) or function block diagram/ladder diagram (FBD/LD), EN will be the execution condition. The instruction is executed only when EN is TRUE. The status of ENO will be the same as that of EN.

Note that the execution condition of standard functions and function blocks differs depending on the existence of EN. If there is no EN, the standard function or function block is executed at every scan. For the execution condition of the standard function or function block with EN, refer to the details of each standard function or function block in this manual.

For the execution conditions on the motion control function blocks, refer to the following.

☞ Page 1387 Execution command (Execute) type/Enable (Enable) type

1.4 Precautions on Programming

Errors common to instructions

The following table lists the conditions under which an error occurs when the instruction is executed.

Error description*1	Error code
An I/O number which is out of range (other than 000H to FFFH and 3E0H to 3E3H) is specified.	2800H
An I/O number which corresponds to no module is specified.	2801H
An I/O number of the module that cannot be specified by using the instruction is specified.	2803H
A network number which is out of range (1 to 239) is specified.	2804H
A network number which does not exist is specified.	2805H
• The device or label specified by the instruction exceeds the available range. • The file register is accessed while the file register is not set in the file setting of a CPU parameter or the file register to be used in the program is not set.	2820H
• The range of the buffer memory of the module specified by the instruction is exceeded. • The module specified by the instruction does not have buffer memory.	2823H

*1 For a contact instruction, an error is not detected but the operation result becomes no continuity.

Checking the ranges of instruction runtime devices and labels

Device range check

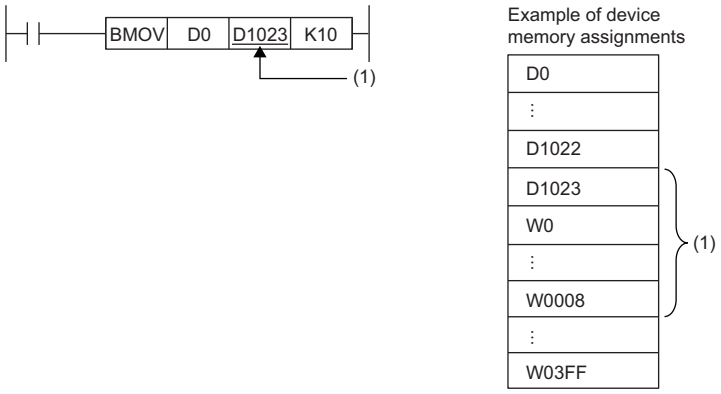
When a device is specified in an instruction, range check is performed, so a program needs to be created so that the operation result falls within the range of the relevant device.

If a range exceeding that of the relevant device is specified, an error (error code: 2820H) occurs.

The same applies if the label assigned to a device is specified in an instruction in the program.

Ex.

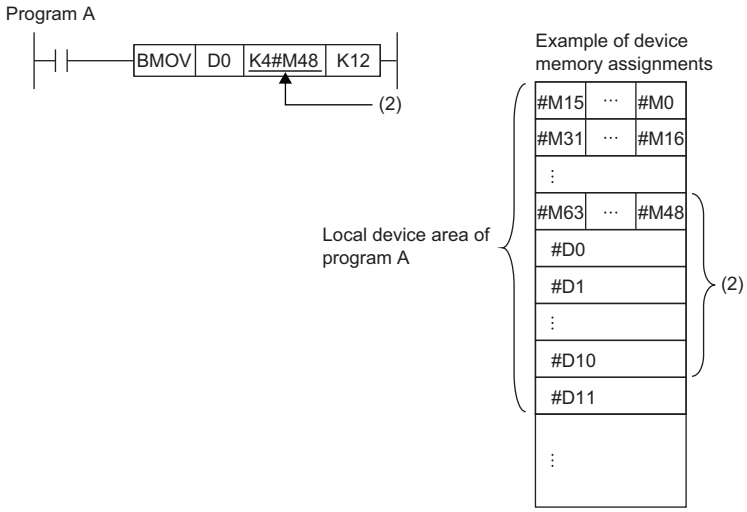
When W0 is assigned after global device D1023 in the device memory



(1) The transfer destination is in the range corresponding to D1023 to D1032. An error occurs because the range from D1024 to D1032 does not exist.

Ex.

When local devices are set in the range from M0 to M63 or D0 to D11



(2) The transfer destination is in the range corresponding to local devices #M48 to #M239 of program A. An error occurs because the range from #M64 to #M239 does not exist.

Checking the range of file register

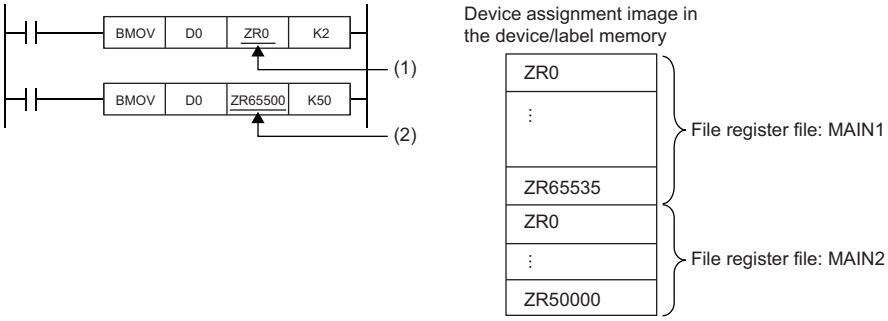
When a file register is specified in an instruction, a range check is performed, so a program needs to be created so that the operation result falls within the range of the relevant file register.

If a range exceeding that of the file register (ZR) is specified, an error (error code: 2820H) occurs.

If a range exceeding that of the file register of the block number used by the file register (R) is specified, an error (error code: 2820H) occurs.

Ex.

When a file register (ZR) is specified

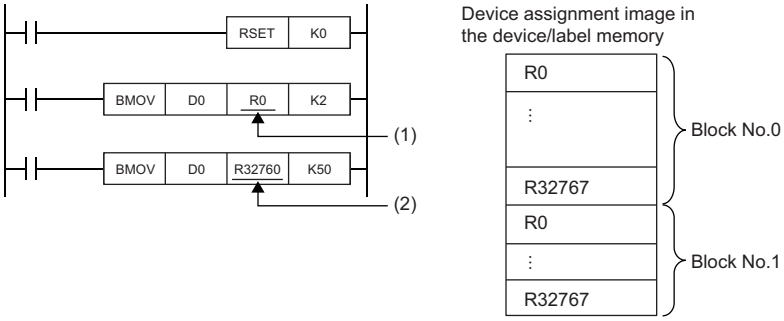


(1) The transfer destination is in the range of file register MAIN1. Data is written to ZR0 and ZR1.

(2) The transfer destination is out of the range of file register MAIN1. An error occurs because the area range of file register MAIN1 is exceeded.

Ex.

When a file register (R) is specified



(1) The transfer destination is in the range of the R device of block number 0. Data is written to R0 and R1.

(2) The transfer destination is out of the range of the R device of block number 0. An error occurs because the area range of the R device of block number 0.

Global (latch) label and local (latch) label range check

When a global (latch) label or local (latch) label is specified in an instruction, range check is performed, so a program needs to be created so that the operation result falls within the range of the relevant label. For digit specification or bit specification as well, specify a value within the range of the relevant label.

If a range exceeding that of the relevant device is specified, an error (error code: 2820H) occurs.

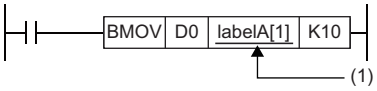
■Array label

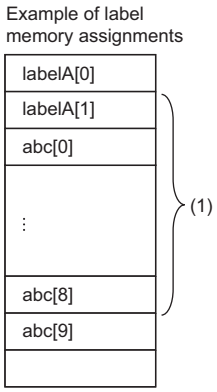
When an array label is specified for an argument, the entire array label is checked. When elements of an array label are specified as well, values within the range of the array label are checked.

Ex.

When labels are arrayed as follows

Label name	Data type	Number of array elements
labelA	Bit string [16 bits]	2
abc	Bit string [16 bits]	10





(1) The transfer destination is in the range corresponding to 10 points from labelA[1]. An error occurs because the range from labelA[2] to labelA[9] does not exist.

■Structure type label

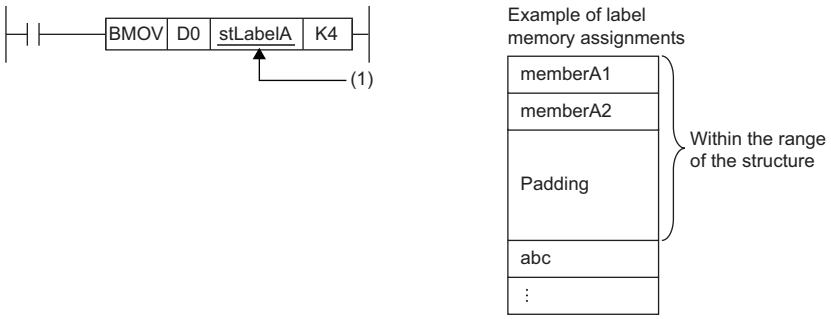
When a structure type label is specified for an argument, the entire structure type label is checked. When a structure member is specified, values within the range that corresponds to the data type of the relevant member are checked.

Ex.

When labels are arrayed as follows

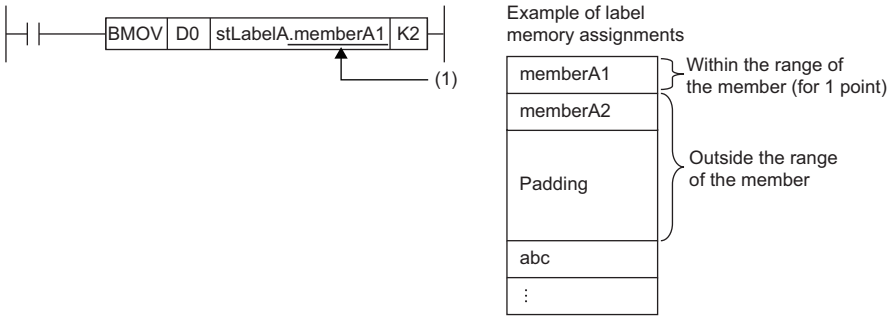
Label name	Data type		Member
stLabelA	Structure A	Word/bit string [16 bits]	memberA1
		Word/bit string [16 bits]	memberA2
abc	Word/bit string [16 bits]		—

- When a structure type label is specified



(1) The transfer destination is in the range corresponding to 4 points from the start of the structure type label "stLabelA". Because access is in the range that includes two points for padding in the structure, data is written.

- When the member "memberA1" is specified



(1) The transfer destination is in the range corresponding to 2 points from the start of memberA1. Because the range of memberA1 corresponds to one point, an error occurs.

Module label and network label range check

When a module label or network label is specified in an instruction, range check is performed in the network label/module label assignment area. Therefore, create a program so that the operation result falls within the range of the network label/module label assignment area.

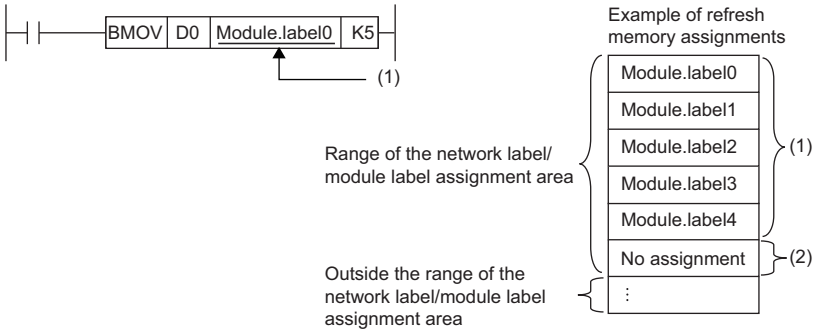
If a range exceeding that of the relevant network label/module label assignment area is specified, an error (error code: 2820H) occurs.

For details on the setting method of the network label/module label assignment area, check the user's manual for the controller used.

Ex.

When module labels are set as follows

Instance name	Label name	Data type	Assignment device
Module	label0	Word [unsigned]	UV: 0
	label1	Word [unsigned]	UV: 1
	label2	Word [unsigned]	UV: 2
	label3	Word [unsigned]	UV: 3
	label4	Word [unsigned]	UV: 4

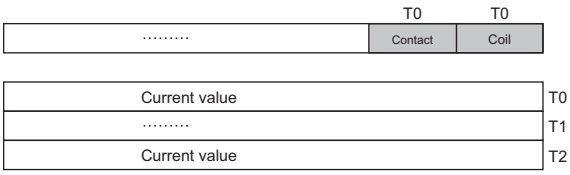
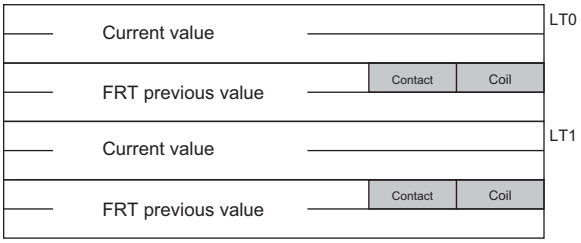
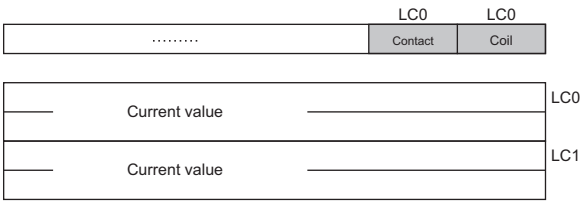


(1) The transfer destination is within the range of the network label/module label assignment area. It is written to Module.label0 to Module.label4. Data is written even if the data type is crossed.

(2) Although the area is outside the transfer destination range, it is within the network label/module label assignment area, so this area can be accessed.

Operation when a long timer or long retentive timer device is used

When the data to be handled exceeds the width (32 bits) of the current value, the long timer or long retentive timer operates by using not only the area of the current value but also the areas of the previous value, contact, and coil.

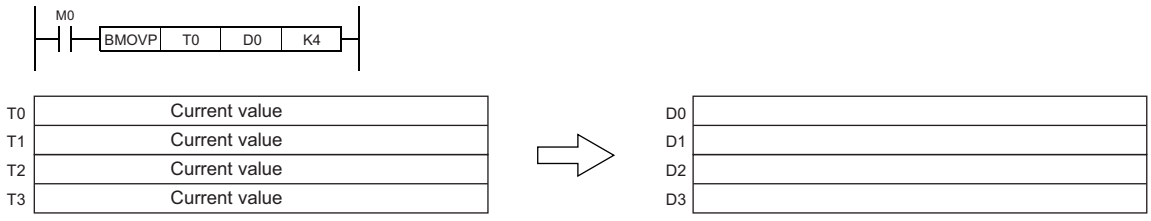
Device	Configuration
Timer (T) Retentive timer (ST) Counter (C)	
Long timer (LT) Long retentive timer (LST)	
Long counter (LC)	

When the BMOV instruction is used to batch-transfer current values, current values alone cannot be batch-transferred. Batch-transfer the current values, contacts, and coils altogether and, after the batch transfer is finished, use only the current values. When the DMOV instruction is used to batch-transfer current values, repeat the transfer of the current values alone using the FOR to NEXT instruction.

Ex.

To batch-transfer the current values of the timer device

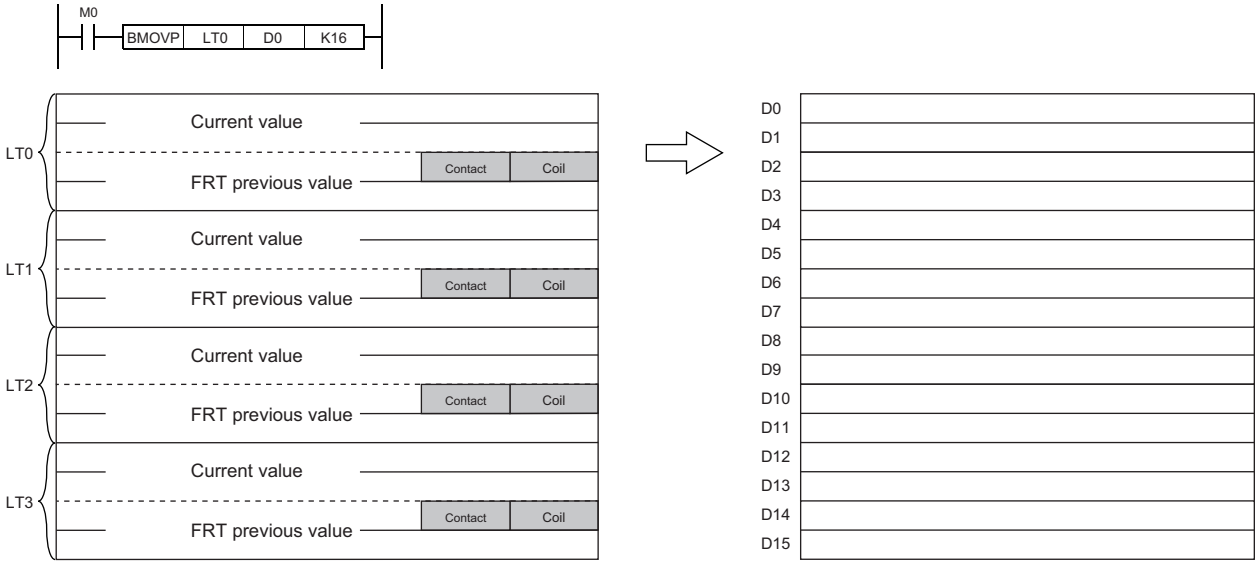
When the BMOV instruction is used, only current values are batch-transferred.



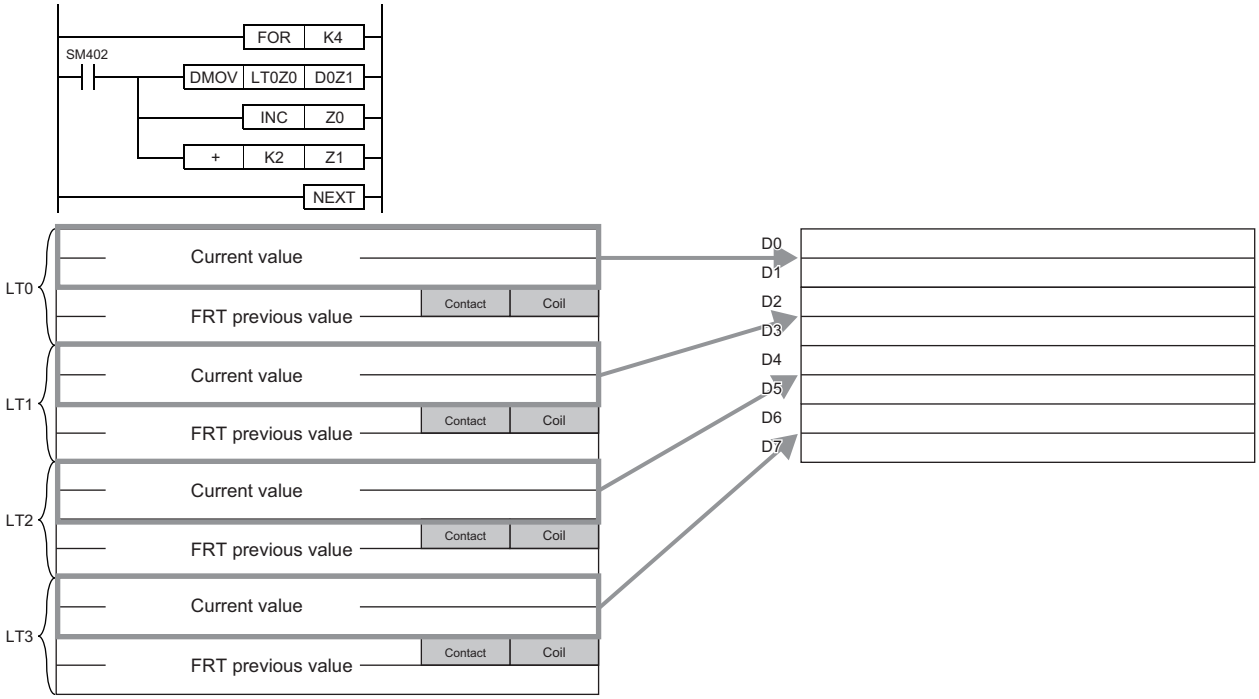
Ex.

To batch-transfer the current values of the long timer device

When the BMOV instruction is used, all current values, contacts, and coils are batch-transferred.



When the DMOV instruction is used, only current values are batch-transferred.



Precautions for using a link device assigned to a synchronized station as an instruction argument

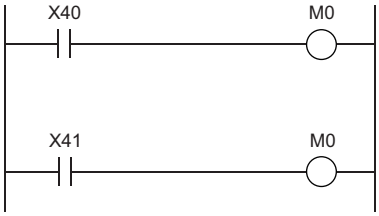
When using a link device assigned to a synchronized station in CC-Link IE TSN as an instruction argument, set link refresh and then specify the refresh target device or label. If direct access device (U□\G□, J□\□) is specified to an instruction argument, the link device does not operate correctly.
For details on CC-Link IE TSN, check the user's manual for the controller used.

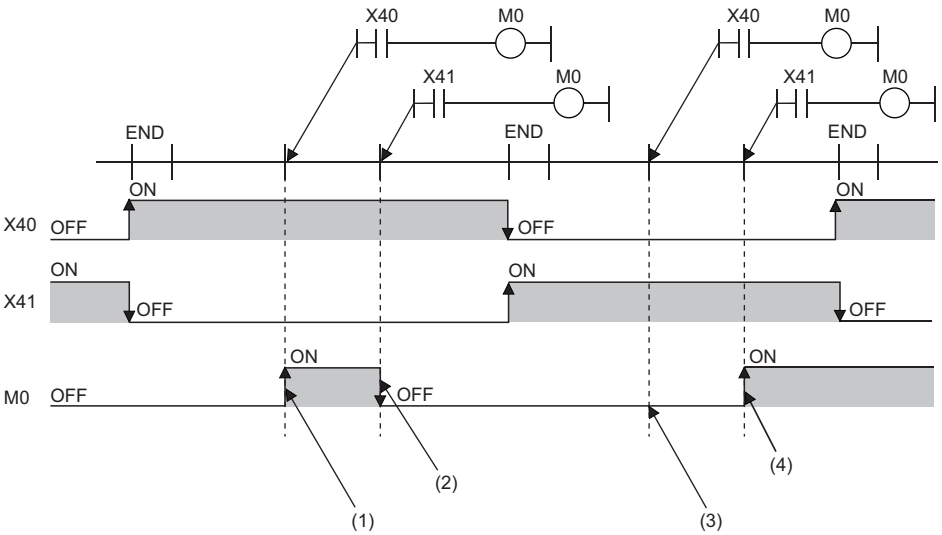
Operations arising when the OUT, SET/RST, and PLS/PLF instructions of the same device are used

This section describes the operation when two or more OUT, SET/RST, and PLS/PLF instructions that use the same device are executed within one scan.

For OUT instructions of the same device

More than one OUT instruction of the same device must not be issued during one scan.
Otherwise, the specified device turns on or off, depending on the operation result up to each OUT instruction while it is in execution.
In this case, the device may turn on/off during one scan because the on/off state of the specified device is determined during execution of each OUT instruction.
The following figure shows the behavior arising when a circuit turning on/off the same internal relay (M0) is created with input X40 and X41.





(1) Since X40 is on, M0 turns on.
(2) Since X41 is off, M0 turns off.
(3) Since X40 is off, M0 remains off.
(4) Since X41 is on, M0 turns on.

If output (Y) is specified using an OUT instruction, the on/off state of the last OUT instruction executed during the one scan will be output.

If SET/RST instructions of the same device are used

■For SET instructions

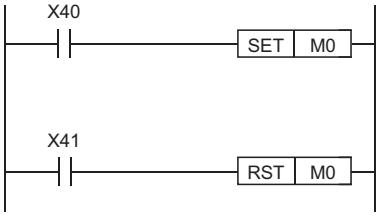
The SET instruction turns on the specified device if the execution command is on, and causes no operation if it is off.
Thus, if two or more SET instructions of the same device are executed during one scan, the specified device turns on even if one execution command is on.

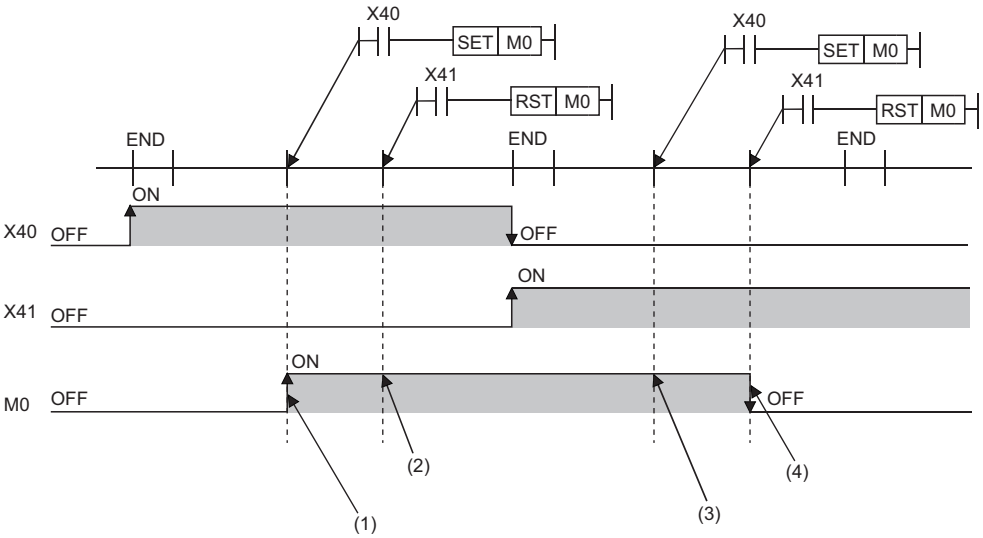
■For RST instructions

The RST instruction turns on the specified device if the execution command is off, and causes no operation if it is off.
Thus, if two or more RST instructions of the same device are executed during one scan, the specified device turns on even if one execution command is off.

■If the SET and RST instructions of the same device exist in one scan

If the SET and RST instructions of the same device exist in one scan, the SET instruction turns on the specified device if the execution command is on, and turns off the specified device if it is on.
If both the SET and RST instructions are off, the on/off state of the specified device will be unchanged.



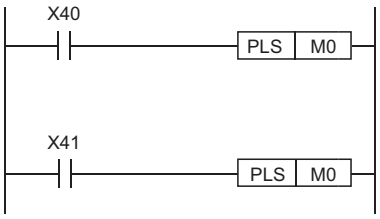


(1) Since X40 is on, M0 turns on.
(2) Since X41 is off, M0 remains on. (The RST instruction results in non-processing.)
(3) Since X40 is off, M0 remains on. (The SET instruction results in non-processing.)
(4) Since X41 is on, M0 turns off.

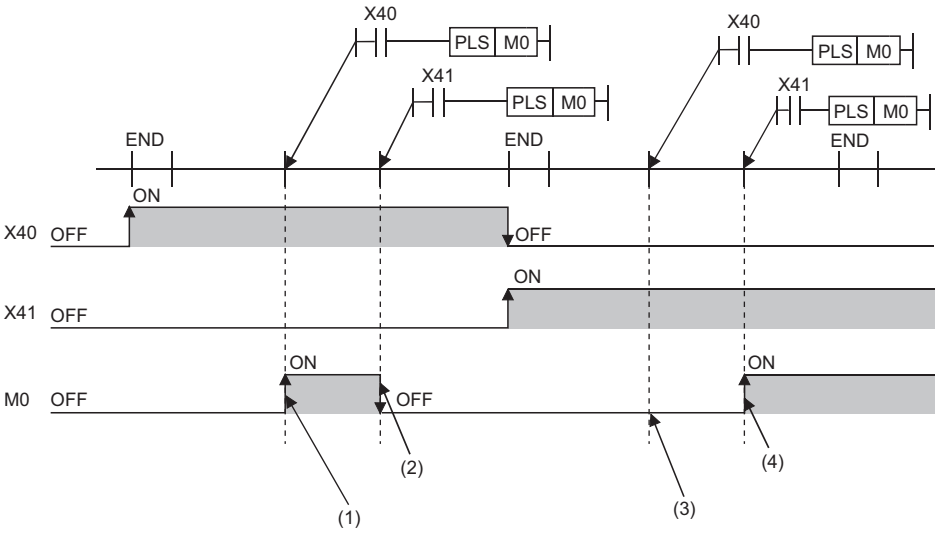
If output (Y) is specified using a SET/RST instruction, the on/off state of the last SET/RST instruction executed during the one scan will be output.

If PLS instructions of the same device are used

The PLS instruction turns on the specified device when the execution command specifies an off-to-on change. The specified device is turned off unless the execution command specifies an off-to-on change (i.e. off to off, on to on, on to off).
Thus, if two or more PLS instructions of the same device are issued during one scan, the specified device is turned on when the execution command of each PLS instruction specifies an off-to-on change. The specified device is turned off unless the execution command specifies an off-to-on change.
Thus, if two or more PLS instructions are issued during one scan, the device turned on by a PLS instruction may not turn on for one scan.

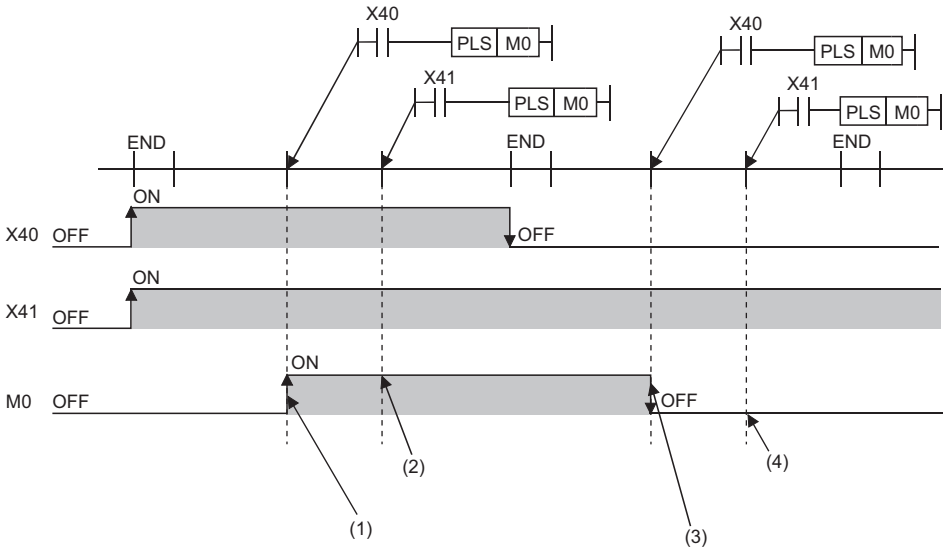


• If X40 and X41 differ in the on/off timing (i.e. the specified device does not turn on for one scan)



(1) Since X40 turns on, M0 turns on.
(2) Since X41 is other than turning on, M0 turns off.
(3) Since X40 is other than turning on, M0 remains off.
(4) Since X41 turns on, M0 turns on.

• If the off-to-on changes of X40 and X41 are at the same timing

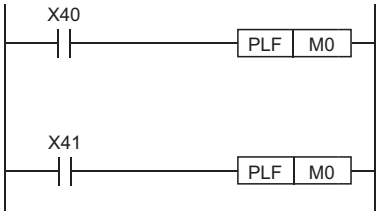


(1) Since X40 turns on, M0 turns on.
(2) Since X41 turns on, M0 remains off.
(3) Since X40 is other than turning on, M0 turns off.
(4) Since X41 is other than turning on, M0 remains off.

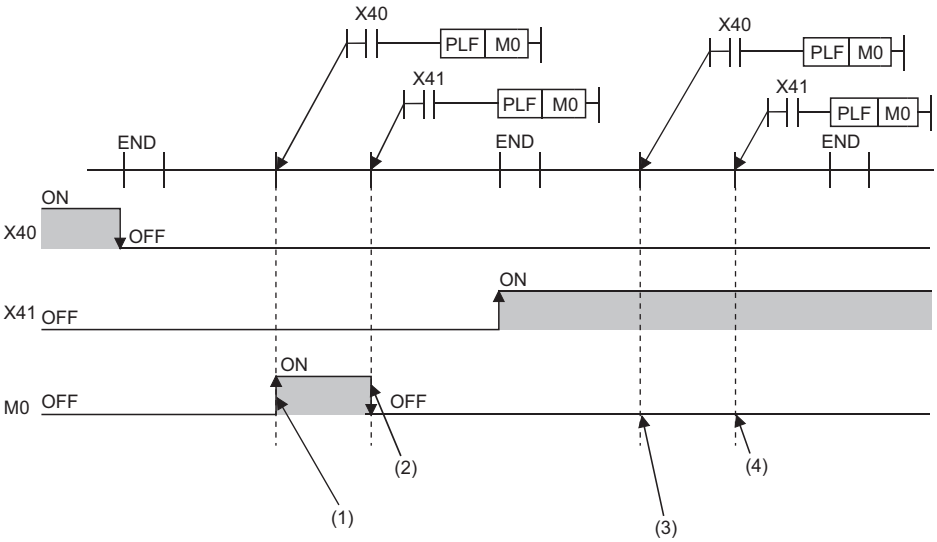
If output (Y) is specified using a PLS instruction, the on/off state of the last PLS instruction executed during the one scan will be output.

If PLF instructions of the same device are used

The PLF instruction turns on the specified device when the execution command specifies an off-to-on change. The specified device is turned off unless the execution command specifies an on-to-off change (i.e. off to off, off to on, on to on).
Thus, if two or more PLF instructions of the same device are issued during one scan, the specified device is turned on when the execution command of each PLF instruction specifies an on-to-off change. The specified device is turned off unless the execution command specifies an on-to-off change.
Thus, if two or more PLF instructions are issued during one scan, the device turned on by a PLF instruction may not turn on for one scan.

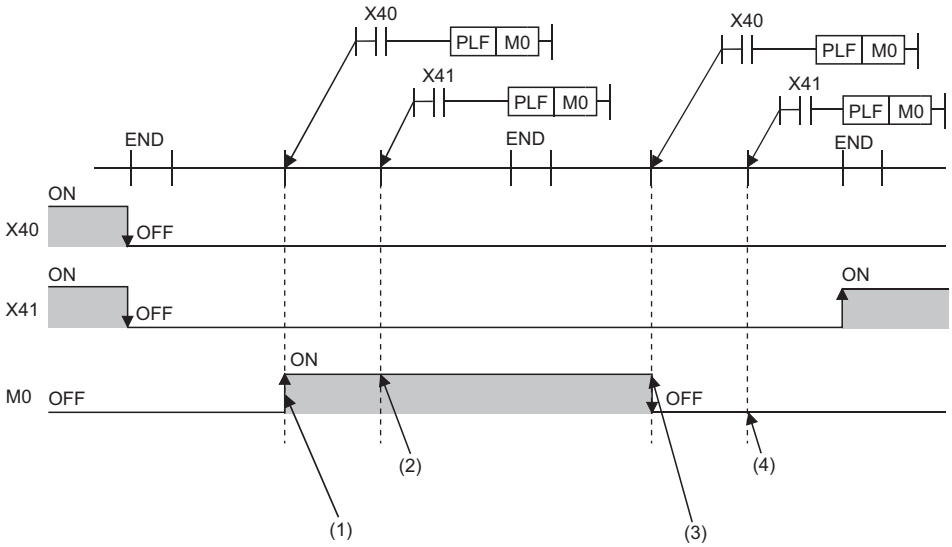


• If X40 and X41 differ in the on/off timing (i.e. the specified device does not turn on for one scan)



(1) Since X40 turns off, M0 turns on.
(2) Since X41 is other than turning off, M0 turns off.
(3) Since X40 is other than turning off, M0 remains off.
(4) Since X41 is other than turning off, M0 remains off.

• If the on-to-off changes of X40 and X41 are at the same timing



(1) Since X40 turns off, M0 turns on.
(2) Since X41 turns off, M0 remains on.
(3) Since X40 is other than turning off, M0 turns off.
(4) Since X41 is other than turning off, M0 remains off.

If output (Y) is specified using a PLF instruction, the on/off state of the last PLF instruction executed during the one scan will be output.

Restrictions on using file registers

When a file register is specified for the refresh device, note the following restrictions.

When the block number is changed by the RSET instruction

When the block number is changed by the RSET instruction, note the following.

- Data is refreshed by the file register (R) of the new block number.
- Data is refreshed by the file register (R) of the block number immediately before refresh.

To read or write refresh data, specify the block number immediately before refresh.

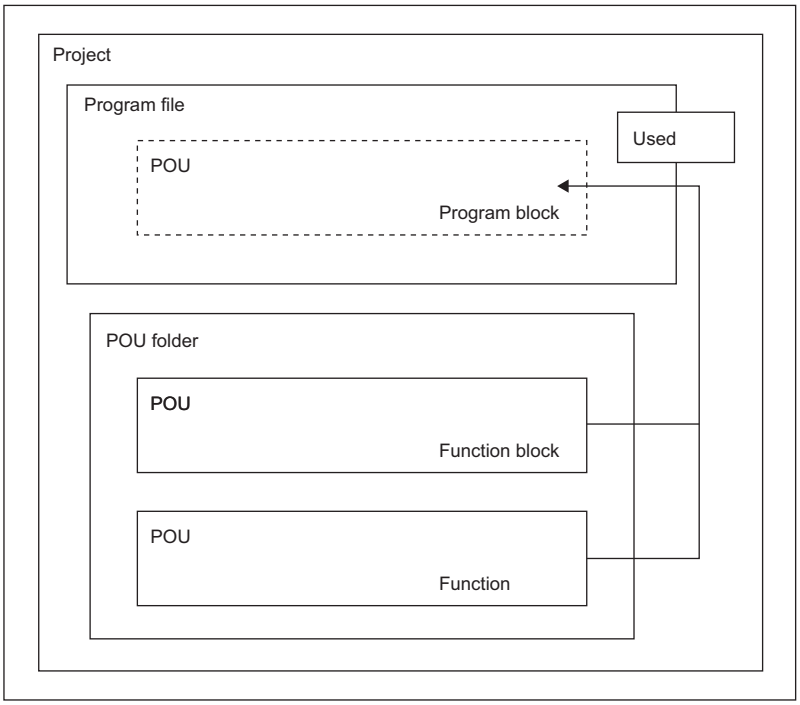
2 PROGRAM ORGANIZATION UNITS

There are three types of program organization units (POUs).

- Program block
- Function
- Function block

Processing can be described in the programming language that suits the control performed in each POU. Processing can be described in the ladder diagram, structured text language, or FBD/LD in a function or a function block.

Functions and function blocks are called and executed by program blocks.



Point

A structured program is a program created by components. Processes in lower levels of hierarchical sequence program are divided into several components according to their processing information and functions.

Each component of a program is specified to have a high degree of independence for easy addition and replacement.

The following are the examples of processing that would be ideal to be structured.

- Processing which is used repeatedly in a program
- Processing which can be separated as one function

This chapter describes two types of POUs using labels.

Devices can also be used in the program (worksheet) of each POU. For details on devices, refer to the following.

User's manual for the controller used

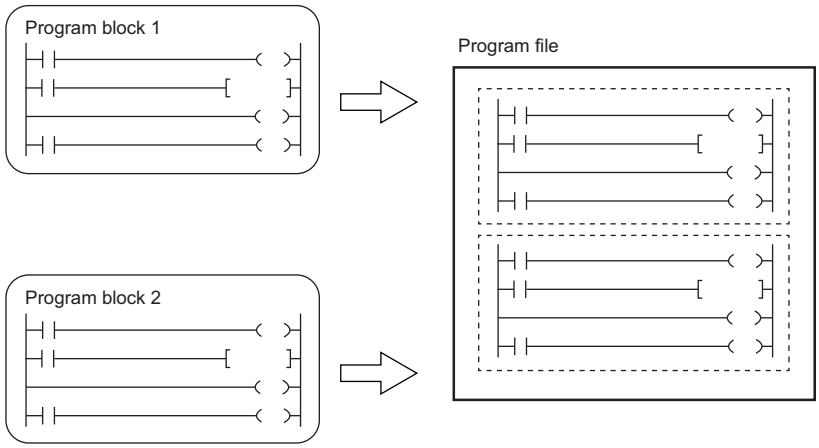
Point

Up to 32 worksheets can be created in one POU in the structured text language and FBD/LD.

Set the execution order of multiple worksheets on the "Worksheet Execution Order Setting" window of the engineering tool. (GX Works3 Operating Manual)

2.1 Program Blocks

A program block is a unit for making up a program.
Multiple program blocks can be created in a program file and executed in the order specified in the program file setting. If the order is not specified in the program file setting, the program blocks are executed in ascending order of their names.
By separating program blocks for individual functions and processing, the order of programs can be changed easily and programs can be exchanged easily.
The program of a program block is stored by each registration destination program in a program file.



Dividing into program blocks

A main routine program, subroutine program, and interrupt program can be created separately in individual program blocks.

Program type	Description
Main routine program	A program beginning with step 0 and ending with the FEND instruction
Subroutine program	A program beginning with a pointer (P) and ending with the RET instruction. This program is executed only when it is called by a subroutine call instruction (such as the CALL and ECALL instructions).
Interrupt program	A program beginning with an interrupt pointer (I) and ending with the IRET instruction. When an interrupt factor occurs, the interrupt program corresponding to the interrupt pointer number is executed.

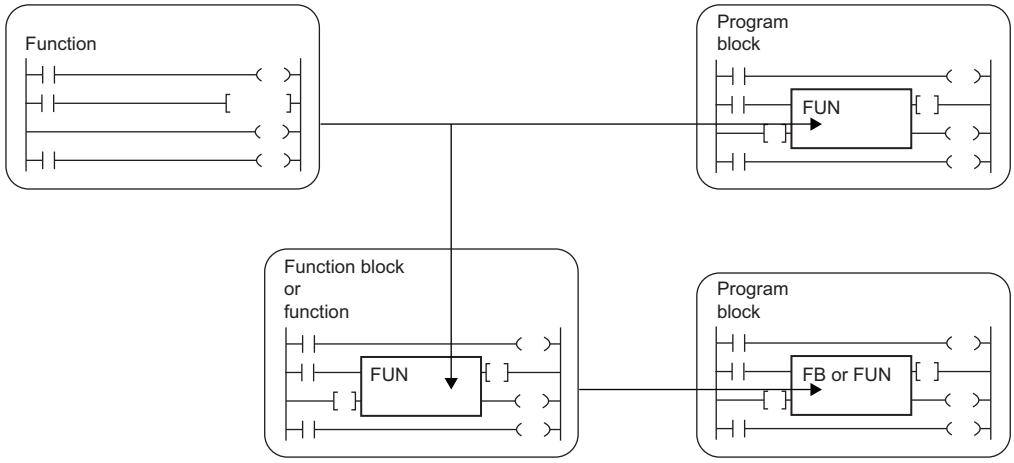
Point

- Create a subroutine program and interrupt program after the FEND instruction of the main routine program. Any program after the FEND instruction is not executed as a main routine program. For example, when the FEND instruction is used at the end of the second program block, the third program block or later runs as a subroutine program or interrupt program.
- To create an easy-to-understand program, use a pair of instructions, such as the FOR and NEXT instructions or the MC and MCR instructions, within a single program block.
- A simple program can be executed in the controller simply by writing the main routine in one program block.

For details on the subroutine program and interrupt program, refer to the following.
User's manual for the controller used

2.2 Functions (FUN)

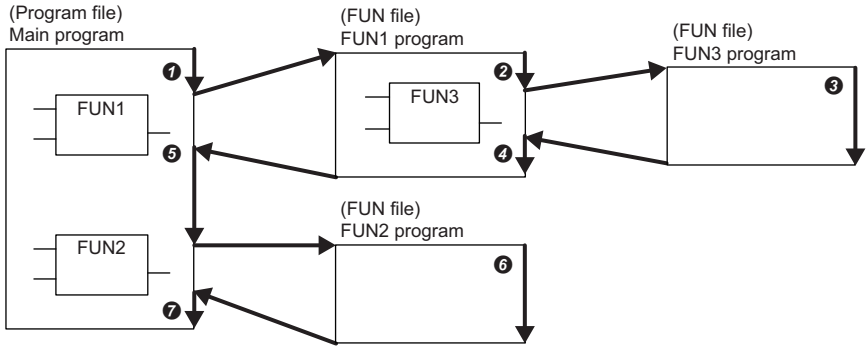
A function is a POU called and executed by program blocks, function blocks, and other functions.
After the processing completes, a function passes a value to the calling source. This value is called a return value.
A function always outputs the same return value, as the processing result, for the same input.
By defining simple, independent algorithms that are frequently used, functions can be reused efficiently.



Operation overview

The program of a function is stored in the FUN file and called by the calling source program when executed.

Ex.
When calling FUN1 and FUN2 from the main program, and calling FUN3 by FUN1 (Nested three times)
❶ to ❼ indicate the execution flow (order).

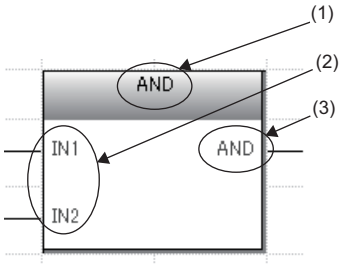


Up to 32 subroutine type function blocks and functions in total can be nested.

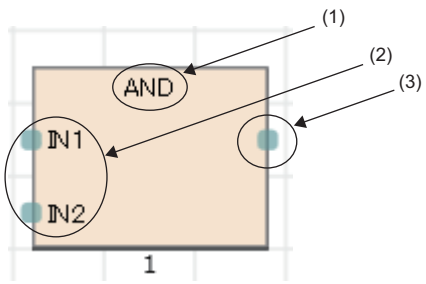
Input variables and output variables

Input and output variables can be defined in functions. Output data which is different from the return value can be assigned to the output variable.

Ladder program



FBD/LD program



(1) Function name
(2) Input variable
(3) Output variable

The return value of the function is not displayed.

Input variables are set in the VAR_INPUT class and output variables are set in the VAR_OUTPUT class.

Point

Variables defined in the function are overwritten every time the function is called.
To hold the data in the variables, create a program by using function blocks or so that the data in the output variable is saved in another variable.

EN and ENO

EN (enable input) and ENO (enable output) can be appended to a function to control execution processing.

- Set a boolean variable used as an execution condition of a function to EN.
- A function with EN is executed only when the execution condition of EN is TRUE.
- Set a boolean variable used to output a function execution result to ENO.

The following table lists the ENO states and operation results according to the EN states.

EN	ENO	Operation result
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE	Undefined value

Point

- Setting an output label to ENO is not always required for the program written in ladder or FBD/LD.
- When EN/ENO is used in a standard function, the function with EN is represented by "function-name_E".

Creating programs

The program of a function can be created by using the engineering tool.

[Navigation window] ⇨ [FB/FUN] ⇨ Right-click ⇨ [Add New Data]
Select "Function" for "Data Type" in "Basic Setting".

The created program is stored in the FUN file.

[CPU Parameter] ⇨ [Program Setting] ⇨ [FB/FUN File Setting]

Up to 64 created programs can be stored in one FUN file.

The rising edge execution instruction or falling edge execution instruction cannot be used in the function.

For details on program creation, refer to the following.

Item	Reference
How to create function programs	GX Works3 Operating Manual
Number of FB/FUN files that can be written to a controller	User's manual for the controller used

■Applicable devices and labels

The following table lists the devices and labels that can be used in function programs.

○: Applicable, △: Applicable only in instructions (Cannot be used to indicate the program step.), ×: Not applicable

Type of device/label		Availability
Label (other than the pointer type)	Global label	×
	Local label	○*1
Label (pointer type)	Pointer type global label	×
	Pointer type local label	○
Device	Global device	○
	Local device	×
Pointer	Global pointer	△
	Local pointer	×

*1 The following data types cannot be used.
Timer, retentive timer, counter, long timer, long retentive timer, and long counter

Point

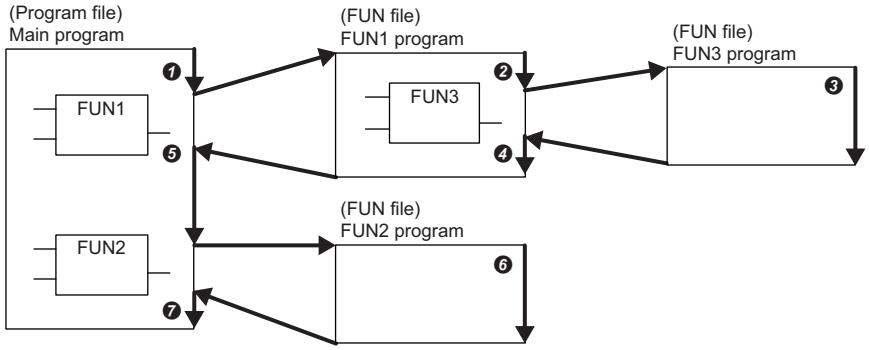
- Program a function name as a label in a function to set a return value of the function. Setting function names as labels is not necessary. The data type set in "Result Type" in the properties of the function can be used.

Labels defined by a function

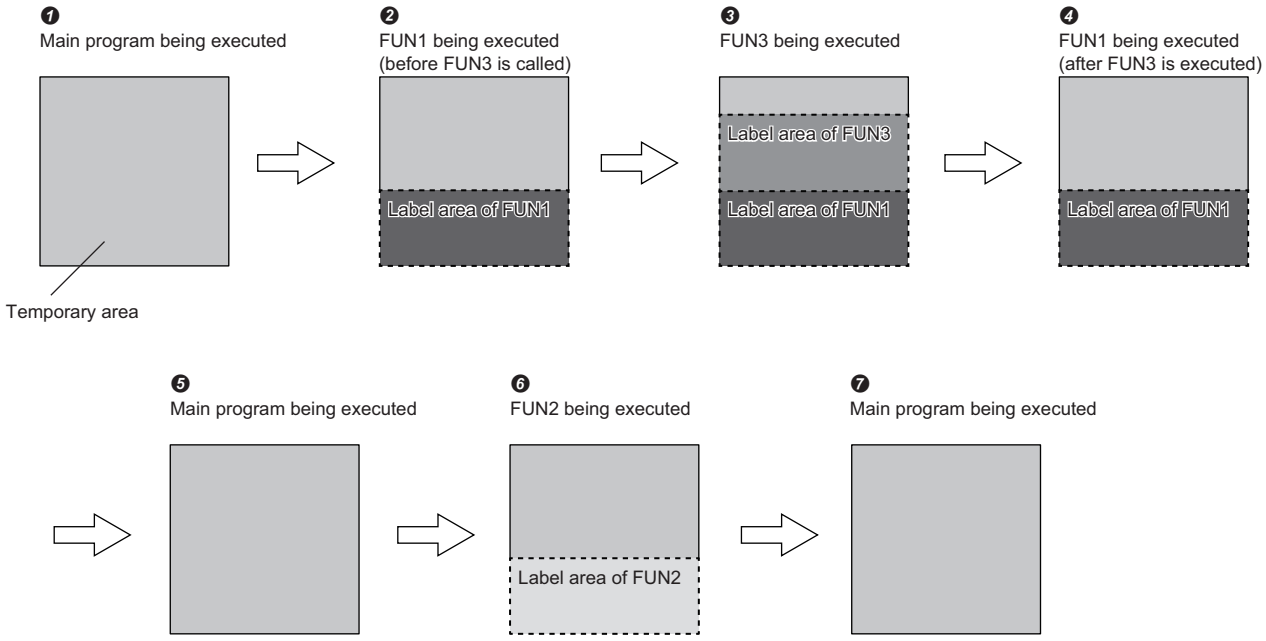
The labels defined by a function are assigned in the temporary areas of the storage-target memory during execution of the function, and the areas are freed after the processing completes.

Ex.

When calling FUN1 and FUN2 from the main program, and calling FUN3 by FUN1
(❶ to ❼ indicate the execution flow (order).)



The following figure shows the label assignments while the above functions are being executed.



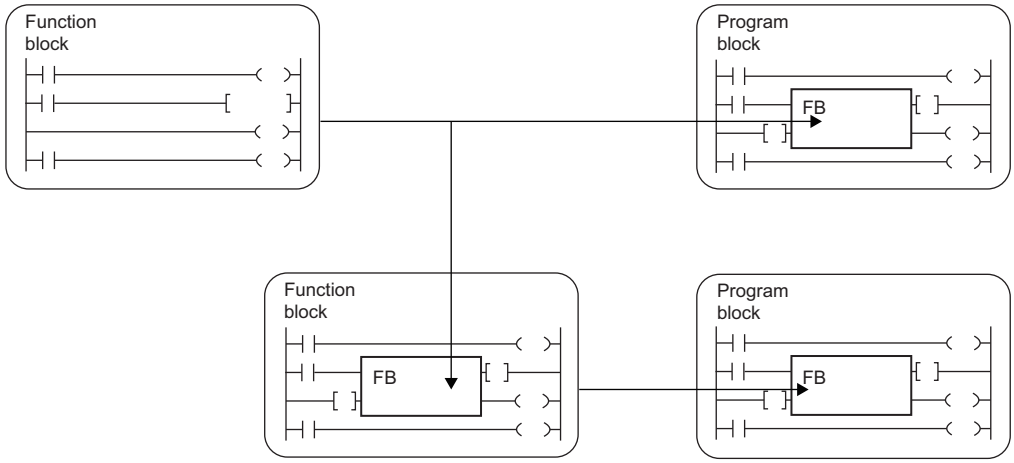
The class of labels that can be defined in the function are VAR, VAR_CONSTANT, VAR_INPUT, and VAR_OUTPUT.

Point

The label to be defined by a function must be initialized by a program before the first access because the label value will be undefined.

2.3 Function Blocks (FB)

A function block is a POU called and executed by program blocks and other function blocks.

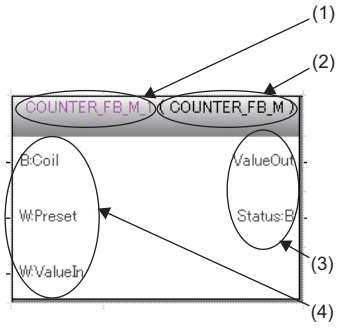


Unlike a function, a function block does not have a return value.

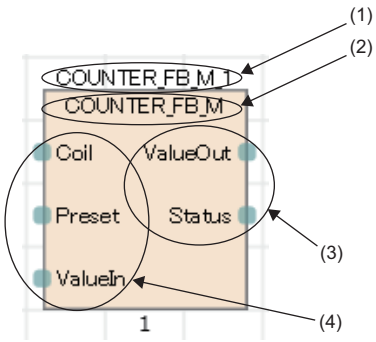
A function block can hold values in variables and thus can hold input states and processing results.

A function block uses the value it holds for the next processing and therefore it does not always output the same result even with the same input value.

Ladder program



FBD/LD program



(1) Instance name
(2) Function block name
(3) Output variable
(4) Input variable

A function block needs to be instantiated to be used in programs.

☞ Page 68 Instances

Point

- For details on module function blocks, refer to the following.

📖 Function Block Reference for the module used

Operation overview

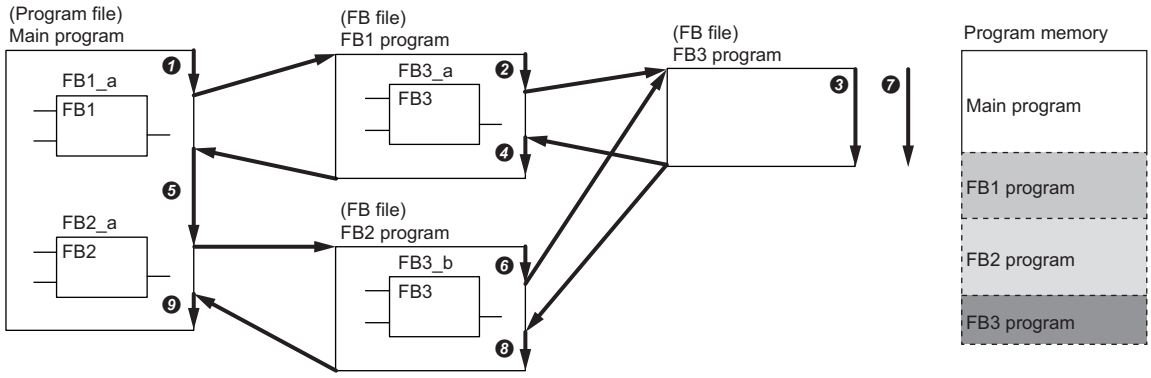
■Subroutine type function blocks

The program of a subroutine type function block is stored in the FB file and called by the calling source program when executed.

Ex.

When calling FB1_a and FB2_a from the main program, calling FB3_a by FB1_a, and calling FB3_b by FB2_a (Nested three times)

❶ to ❾ indicate the execution flow (order).

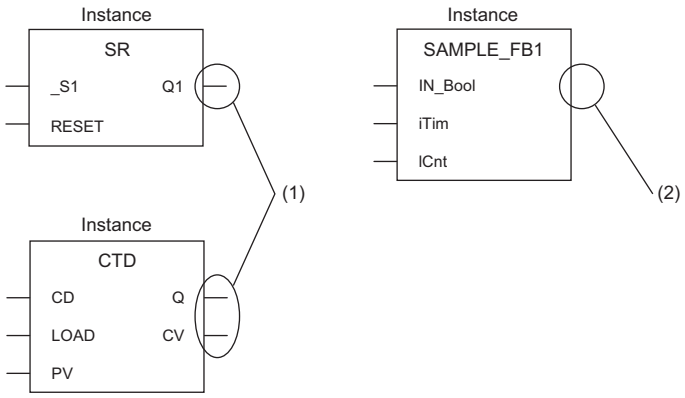


Up to 32 subroutine type function blocks and functions in total can be nested.

Input variables, output variables, and input/output variables

Input variables, output variables, and input/output variables need to be defined in function blocks.

A function block can output multiple operation results. It can also be set not to output operation results.



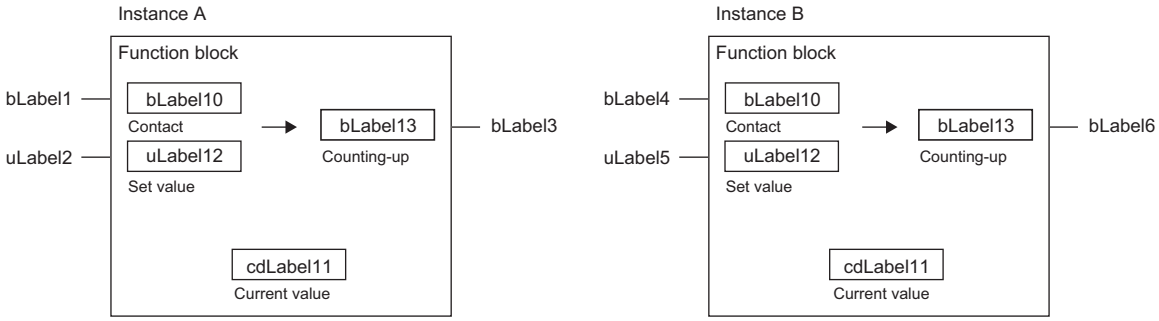
(1) The operation result(s) is output.
(2) No operation result is output.

Input variables are set in the VAR_INPUT class, output variables are set in the VAR_OUTPUT class and VAR_OUTPUT_RETAIN class, and input/output variables are set in the VAR_IN_OUT class.

Internal variables

Function blocks use internal variables. For each instance of a function block, labels are assigned to the different areas. Even though the same label names are used, different states are held for each instance.

Ex.



The above function block starts counting when the input variables turn on and turns on the output variable when the current value held in the internal variable reaches the set value. Even though the same function block is used, the output timings differ because the instances A and B hold different states.

Internal variables are set in the VAR, VAR_CONSTANT and VAR_RETAIN class.

External variables and public variables

Function blocks can use external variables (global label) and public variables.

Public variables are set in the VAR_PUBLIC and VAR_PUBLIC_RETAIN class.

Instances

■Instances

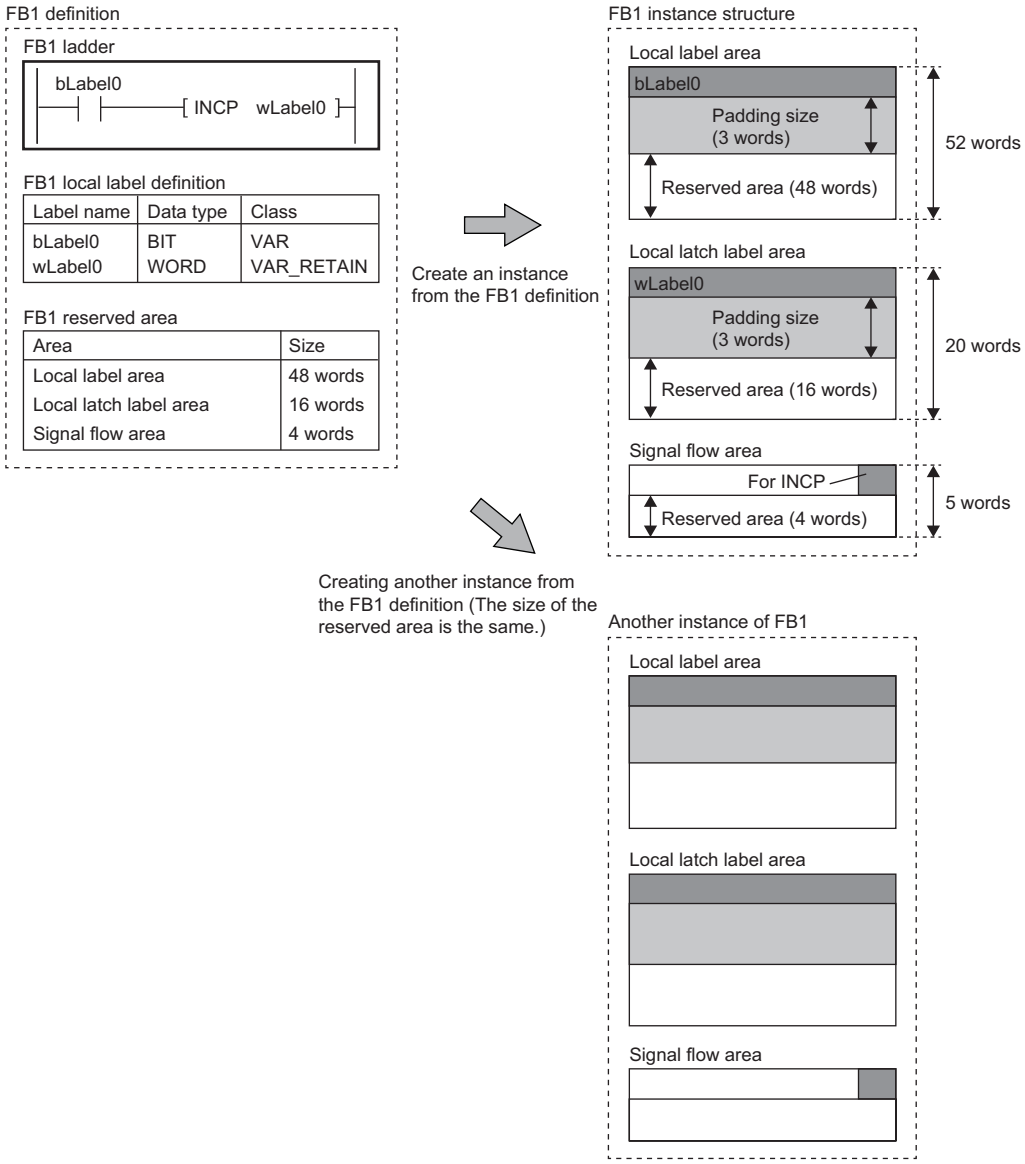
An instance is a label assigned to realize a function block definition. Multiple instances can be created from one function block definition.

An instance consists of the following items.

Item	Description
Local label area	Used to assign the local label of the function block.
Local latch label area	Used to assign the latch attribute local label of the function block.
Signal flow area	Used to assign the signal flow for the instruction in the function block definition.

Ex.

Structure of instance



For the local label area and local latch label area, since the label area is secured in units of four words, three-words (padding size) are secured in the above example.

Each area has a reserved area. The reserved area is used to add or change instances of the function block or instructions that refer to local labels or the signal flow memory with the label assignment intact even when the program is converted or changed online.

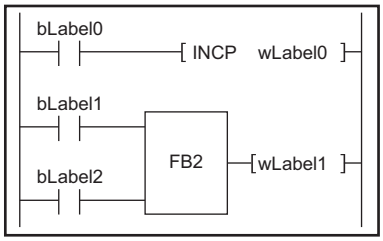
If the area of the target data type to be added or changed cannot be secured, all programs are required to be converted, or the program is required to be converted and changed online.

Ex.

Structure of instance for nested function block (when reserved area of FB2 is changed)

FB1 definition

FB1 ladder



FB1 local label definition

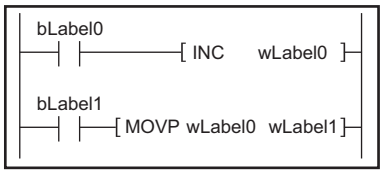
Label name	Data type	Class
bLabel0	BIT	VAR
bLabel1	BIT	VAR
bLabel2	BIT	VAR
wLabel0	WORD	VAR_RETAIN
wLabel1	WORD	VAR
FB2_a	FB2	VAR

FB1 reserved area

Area	Size
Local label area	48 words
Local latch label area	16 words
Signal flow area	4 words

FB2 definition

FB2 ladder



FB2 local label definition

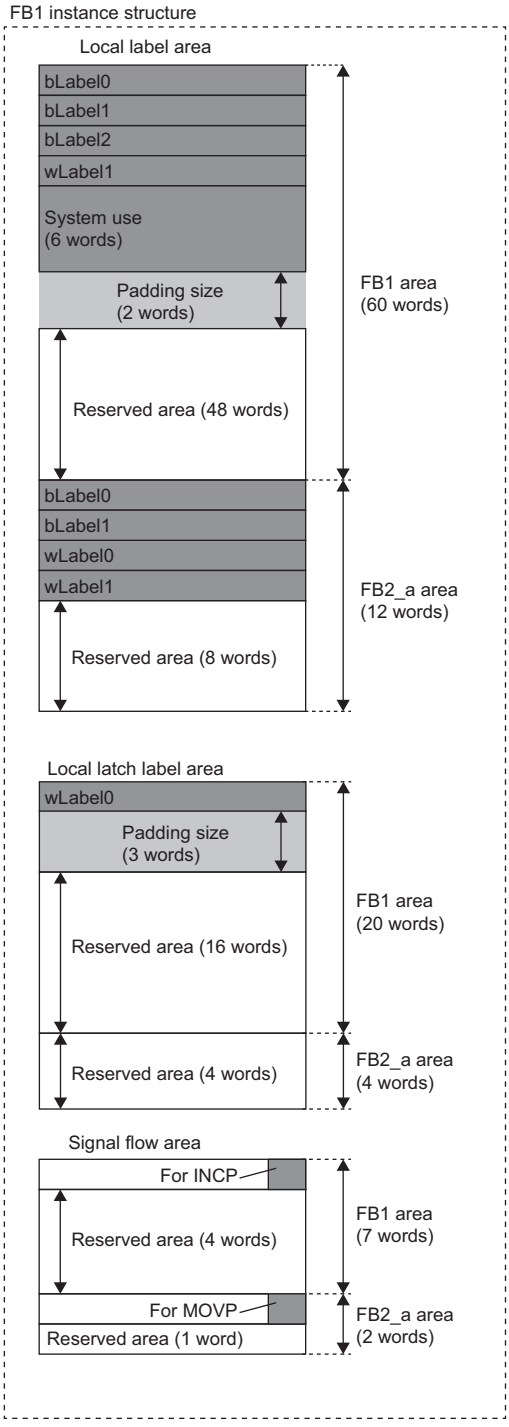
Label name	Data type	Class
bLabel0	BIT	VAR_INPUT
bLabel1	BIT	VAR_INPUT
wLabel0	WORD	VAR
wLabel1	WORD	VAR_OUTPUT

FB2 reserved area

Area	Size
Local label area	8 words
Local latch label area	4 words
Signal flow area	1 word

Creating an instance from the FB1 definition

FB1 instance structure



When an FB nests another FB as shown above, the FB instance declared as a local label is secured outside the instance area of the declared source. Also, an area (6 words) for use by the system is secured in the local label area of the declared source.
When an FB type local label is added or changed in FB1 of the above figure, the capacity for system use (6 words) is required for the FB1 local label area, and the capacity for the FB2 instance area (capacity for each of the local label area, local latch area, and signal flow area) is required for the instance area of the program that becomes the declared source of FB1.
To add or change local labels or FB instances with the assignment intact (without converting all programs), secure the capacity required to add or change them in the reserved area, or convert and change the program online. For the reserved area setting method, refer to the following.

GX Works3 Operating Manual

■Creating instances

A function block needs to be instantiated to be used in programs.

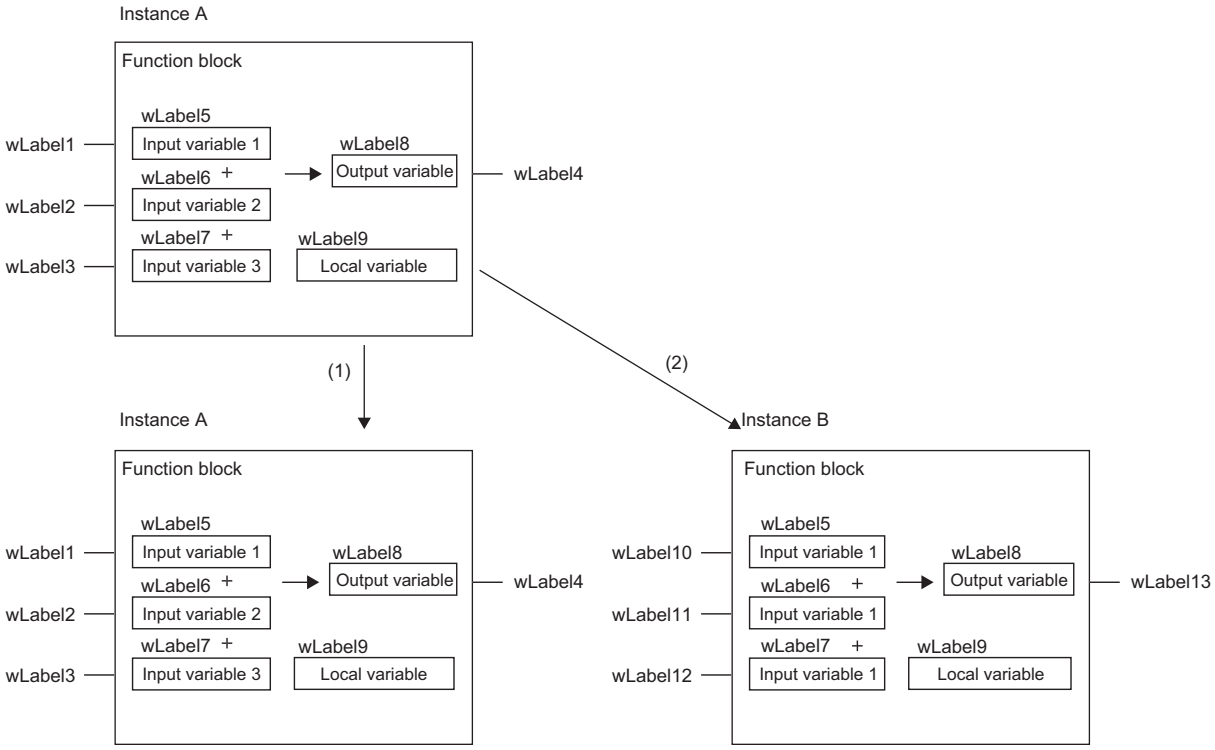
By creating instances, a function block can be called and executed by a program block or another function block.

Declare instances with global labels or local labels.

Label type	Instance type	Class
Global label	Global FB	VAR_GLOBAL
Local label*1	Local FB	VAR

*1 Local labels can be declared as the local labels of a program block or function block. Local labels cannot be declared in a function.

Same function blocks can be instantiated with different names in a single POU.



(1) Same instances use the same internal variables.

(2) Different instances use different internal variables.

■Capacity of instance

The capacity of each data area of an instance should be calculated as follows.

• Capacity of local label area

Capacity of local label area of instance = Total capacity of data of local labels (except the ones with latch attribute) + Capacity of reserved area

Item	Description
Capacity of local labels (except the ones with latch attribute)	Total capacity of the data used for local labels. The capacity of areas to be used differs depending on the memory assignment of labels. For details on memory assignment of labels, refer to the following. GX Works3 Operating Manual
Capacity of reserved area	48 words in default. Set the capacity in units of 4 words.

• Capacity of local latch label area

Capacity of local latch label area of instances = Total capacity of data of local labels with latch attribute + Capacity of reserved area

Item	Description
Capacity of latch attribute local labels	Total capacity of the data used for latch attribute local labels. The capacity of areas to be used differs depending on the memory assignment of labels. For details on memory assignment of labels, refer to the following. GX Works3 Operating Manual
Capacity of reserved area	16 words in default. Set the capacity in units of 4 words.

• Capacity of signal flow area

Capacity of signal flow area of instances = Total capacity of the signal flow area for the instruction in the function block definition + Capacity of reserved area

Item	Description
Capacity of signal flow area	Total capacity of the signal flow area for the instruction in the function block definition
Capacity of reserved area	4 words in default. Set the capacity in units of 1 words.

Point

Set enough reserved area capacity for the data (such as local labels or the signal flow memory) that is expected to be added or changed when the program is changed online. For the setting method, refer to the following.

GX Works3 Operating Manual

If some structure labels are included in the local labels of the function block definition, addition or change exceeding the estimated area capacity cannot be made when the program is converted and changed online, and all programs are required to be converted. For more details, check the user's manual for the controller used.

In the following function blocks, the controller memory can be efficiently used by setting smaller value than default to the reserved area.

- Debugged subroutine type function block where no local label is added or changed and no program is changed
- Subroutine type function block that declares many instances

Setting initial values

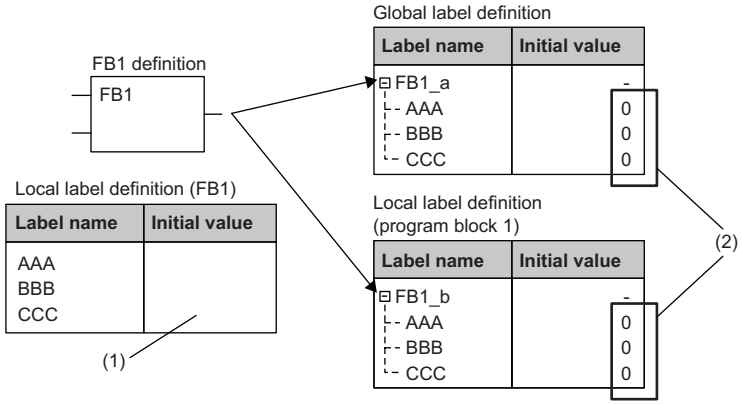
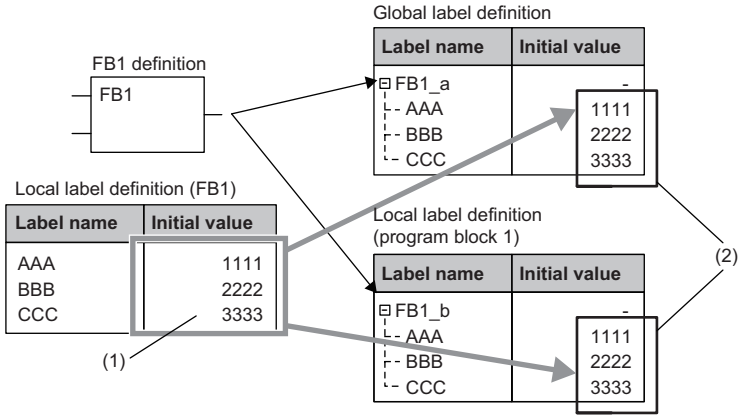
■Initial values of local label

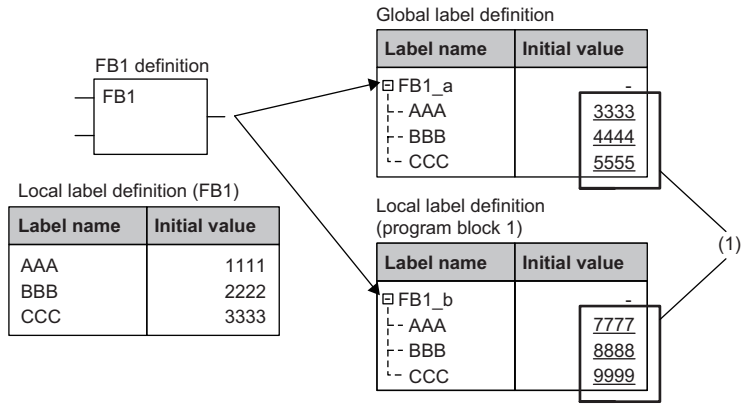
For the local label of a function block, an initial value can be set for each function block definition or instance.

The local label classes where initial values can be set are VAR, VAR_RETAIN, VAR_INPUT, VAR_OUTPUT, VAR_OUTPUT_RETAIN, VAR_PUBLIC, and VAR_PUBLIC_RETAIN.

■Initial values of instance

The following table summarizes the types of the initial values of instances.

Type	Description
Default initial value	An initial value predefined for each data type. If an initial value is not set for the local label of a function block, the default initial value will be used.  (1) The initial values have not been set for the local labels in the FB1 definition. (2) The default initial values are used.
FB definition initial value	An initial value that is set when the local label of a function block is defined. If this initial value has been set, the same definition initial value will be used for all the instances.  (1) The initial values have been set for the local labels in the FB1 definition. (2) All the instances of FB1 will be initialized by the same definition initial value.

Type	Description
Instance initial value	An initial value that is set for an instance included in the global label and program block local label definition.  (1) The initial value can be set for each instance in the FB1 definition.

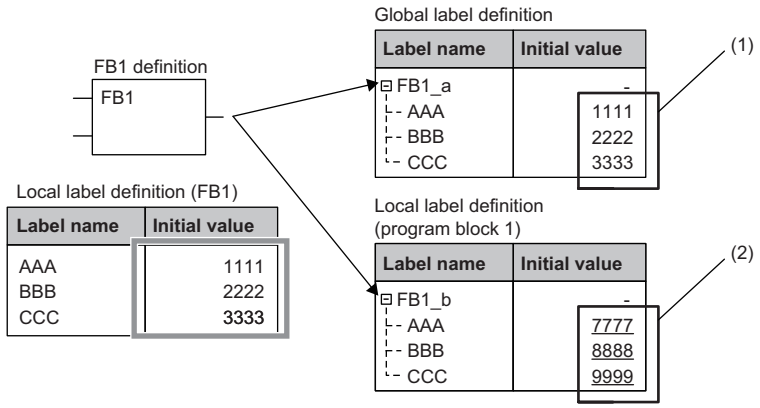
For function blocks, both the FB definition and instance initial values can be set.

If both initial values are set, the initial values used will take the following priority.

Priority	Type
High ↑ ↓ Low	Instance initial value
	FB definition initial value
	Default initial value

Point

If two instances of a function block for which the FB definition initial value has been set are created and the instance initial value is set for only one of them, the FB definition initial value will be used for the instance for which the instance initial value has not been set and the instance initial value will be used for the other.

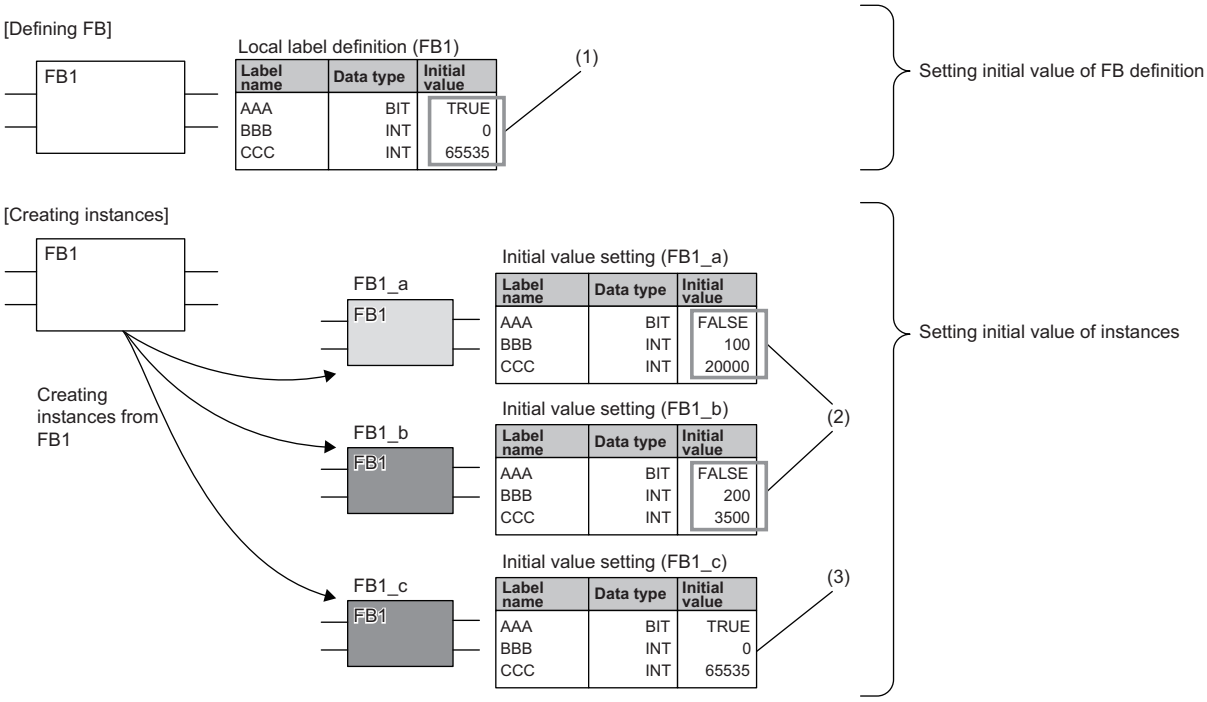


(1) When the instance initial value is not set, the FB definition initial value will be used.

(2) When the instance initial value is set, it will be used.

■Example

The following figure shows an example where the function block initial values are used.



(1) The common initial values are set for all the instances.
(2) The individual initial values can be set for each instance.
(3) If the individual initial values are not set, the common ones will be used.

EN and ENO

In the same way as a function, EN (enable input) and ENO (enable output) can also be appended to a function block to control execution processing. (☞ Page 62 EN and ENO)

When the instance of a function to which EN/ENO has been appended is called, an actual argument must be assigned to EN.

Point

For the execution conditions on the motion control function blocks, refer to the following.

☞ Page 1387 Execution command (Execute) type/Enable (Enable) type

Creating programs

The program of a function block can be created by using the engineering tool.

[Navigation window] ⇨ [FB/FUN] ⇨ Right-click ⇨ [Add New Data]
Select "Function Block" for "Data Type" in "Basic Setting".

The created program is stored in the FB file.

[CPU Parameter] ⇨ [Program Setting] ⇨ [FB/FUN File Setting]

Up to 64 created programs can be stored in one FB file.

For details on program creation, refer to the following.

Item	Reference
How to create function blocks	📖GX Works3 Operating Manual
Number of FB/FUN files that can be written to a controller	📖User's manual for the controller used

■Types of program

There are two types of function blocks and the program of each function block type is stored in different ways.

- Subroutine type function block

For details, refer to the following.

☞ Page 66 Operation overview

The above cannot be selected for module function blocks, standard functions, standard function blocks, and motion control function blocks.

■Inherent property setting

The following items can be set when a program of a function block is created. (📖GX Works3 Operating Manual)

Item	Description
Use MC/MCR to Control EN	For "Yes", the MC/MCR instructions are used to control EN. For "No", the CJ instruction is used to control EN. Select "Yes" when instructions executed at the rising edge or falling edge are used in an FB. The operations of a timer/counter and the OUT instruction used in an FB differ depending on the selected item. For details, refer to the following. ☞ Page 84 Operations of when the MC/MCR instructions are used to control EN
Use EN/ENO	For "Yes", a function block with EN/ENO is created, and EN/ENO labels can be used in a program without registering as local labels. For "No", a function block without EN/ENO is created. For details on EN/ENO, refer to the following. ☞ Page 74 EN and ENO

■Applicable devices and labels

The following table lists the devices and labels that can be used by function block programs.

○: Applicable, △: Applicable only in instructions (Cannot be used to indicate the program step.), ×: Not applicable

Type of device/label		Availability
Label (other than the pointer type)	Global label	○
	Local label	○
Label (pointer type)	Pointer type global label	△
	Pointer type local label	○
Device	Global device	○
	Local device	×
Pointer	Global pointer	△
	Local pointer	×

2.4 Precautions

When a function is used

Pointer

The local pointer and pointer type global labels cannot be used.

When a function block is used

Pointer

The local pointer cannot be used.

When the index register is used

When the index register is used in the function block program, ladder programs for saving and returning the index register values are required to protect the values.
Setting the index register data to 0 when saving can prevent an error that could be caused by an index modification validity check. (Whether the device number exceeds the device range or not is checked.)

Ex.

A program that saves the values in the index register Z1 and Z2 before the program execution and returns the saved values after the program execution



Specifying the start I/O number of intelligent function module

When accessing the buffer memory or I/O signals of the intelligent function module, specify the start I/O number using the index register.

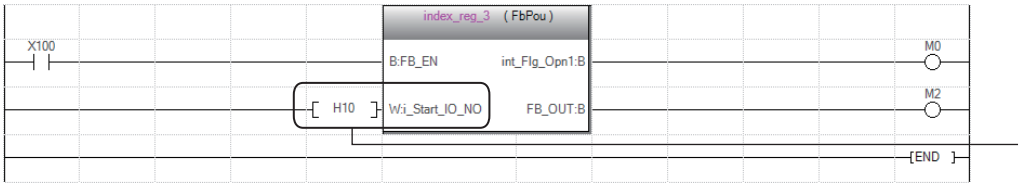
By receiving the start I/O number as an input variable, the same function block can be shared in multiple intelligent function modules without changing the start I/O number.

Ex.

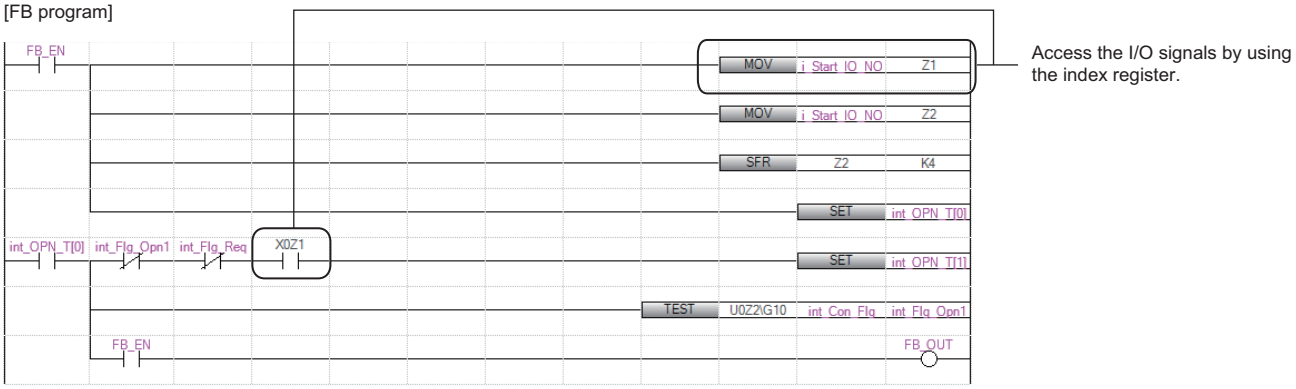
To access the I/O signals of the intelligent function module

Use the index register.

[Sequence program]



[FB program]

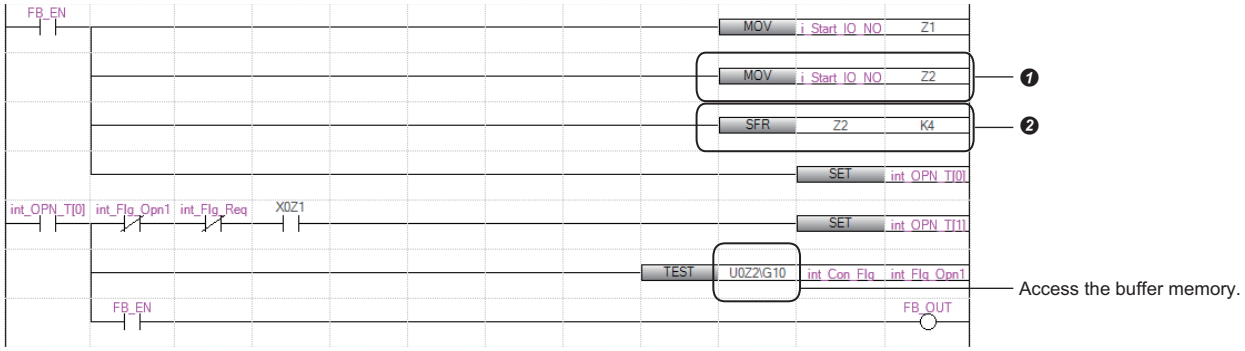


Ex.

To access the buffer memory of the intelligent function module

❶ Input the start I/O number of the target intelligent function module in the index register.

❷ Shift the four bits in the value to the right by using the SFR instruction, or use the quotient obtained by dividing the value by 16.



Copying a macro type function block

When using multiple macro type function blocks, drag and drop them from the Element Selection window.
When the FB compilation method copies a macro type function block on the program editor, the copied function block is changed to a subroutine type function block.
When copying a module FB or FB library, check the reference manual of the function block to be used, and check whether the FB compilation method indicates subroutine type before using it.

Point

A macro type function block is a type of function block used in a module FB or FB library. It cannot be defined by users.
The program is loaded by a calling source program along the execution flow. At the time of program execution, the loaded program is executed in the same way as the main program.

Calling a macro type function block

Call each instance of a macro type function block as a pointer destination by using a pointer type label only once in a project. If the same instance of a macro type function block as a pointer destination using a pointer type label is called multiple times, a conversion error occurs.

Restrictions for module function blocks

The following describes the restrictions for the use of module function blocks.

- Do not turn off the contact of the MC instruction when calling a module function block between the MC instruction and MCR instruction.
- Do not perform the jump processing that prevents module function blocks from being called by the CJ instruction, SCJ instruction, and JMP instruction.
- Execute a subroutine program every scan when calling a module function block in the subroutine program. Do not perform the non-execution processing of a subroutine program by using the FCALL(P) instruction, EFCALL(P) instruction, or XCALL instruction.
- Do not call a module function block in an interrupt program, an initial execution type program, a fixed scan execution type program, or an event execution type program.
- Do not call a module function block between the FOR and NEXT instructions, in the inline ST, or the control syntax of the structured text language (IF statement, FOR statement, and CASE statement.)
- Specifying modified devices for input and output arguments of some module FBs may cause an error. To specify modified devices for input and output arguments of a module FB, describe a processing so that the value is transferred after the module FB is called.

Changing operating parameters of module function blocks

Operating parameters (external variables) other than input and output labels of module function blocks can be changed on the label setting window by using the engineering tool.

- When the instance of a module function block is set to local label, change the parameters on the "Local Label Setting" window.

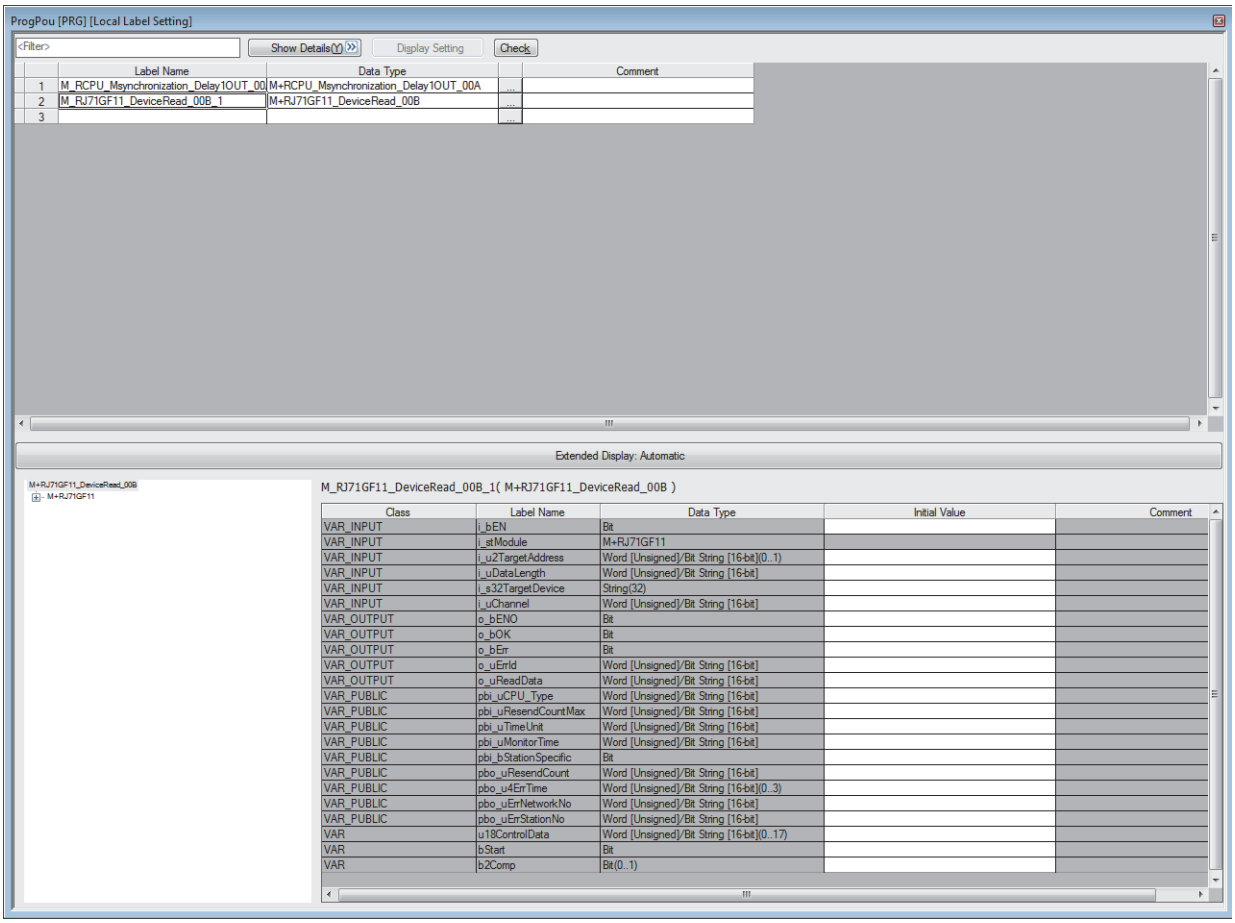
[Navigation Window] ⇨ [Program] ⇨ execution type ⇨ program file ⇨ program block ⇨ [Local Label]

- When the instance of a module function block is set to global label, change the parameters on the "Global Label Setting" window.

[Navigation Window] ⇨ [Label] ⇨ [Global Label]

Ex.

Local Label Setting window



Set operating parameters in the "Initial Value" field.

Operations of when the MC/MCR instructions are used to control EN

The following table lists operations of instructions, devices, and labels used in a function block when "Use MC/MCR to Control EN" is enabled in the inherent property setting of the function block.

Instruction/device/label used in a function block	Operation of Instruction/device/label used in a function block	
	When "Yes" is selected for "Use MC/MCR to Control EN"	When "No" is selected for "Use MC/MCR to Control EN"
Instructions executed at the rising edge or falling edge (PLS, instructions for conversion to pulses (□P)*1)	The next time EN is turned on, the instruction is executed if the condition contact is TRUE.	The next time EN is turned on, the instruction may not be executed even though the condition contact is TRUE.
Timer (low-speed/high-speed), long timer	The count value becomes 0 and the both coil and contact turn off.	The state of devices remains unchanged.
Retentive timer (low-speed/high-speed), long retentive timer, counter, long counter	The coil turns off, but the current count value and the current state of the contact remain unchanged.	The state of devices remains unchanged.
Devices specified as the device part of the OUT instruction	The devices are forcibly turned off.	The state of devices remains unchanged.

*1 Instructions specified in the coil side apply.

Restriction

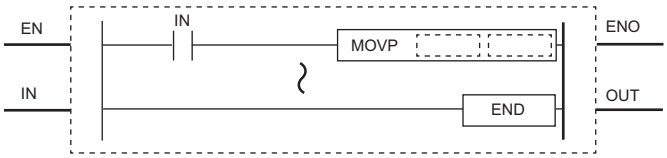
When "Yes" is selected for "Use MC/MCR to Control EN", do not use the MC/MCR instructions while the function block is being executed. If the MC/MCR instructions are used, the EN control may not operate properly.

Instructions executed at the rising/falling edge

The following describes operations of instructions executed at the rising/falling edge.

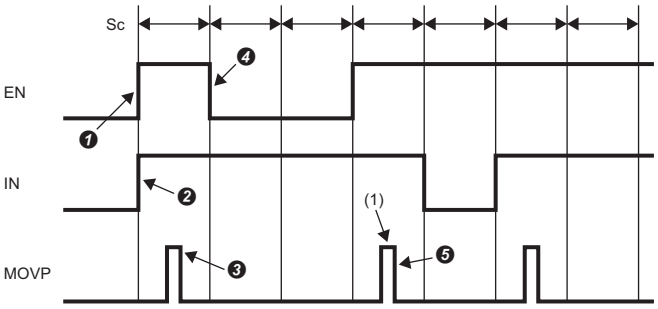
Ex.

A subroutine-type FB using an instruction executed at the rising edge



■When "Yes" is selected for "Use MC/MCR to Control EN"

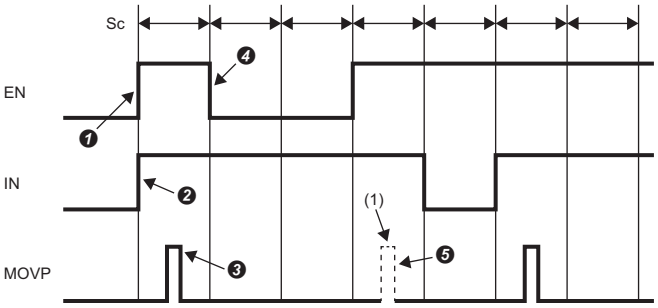
When EN is turned on, the instruction is executed if the condition contact is TRUE ((1) in the following figure).



Sc: Scan
❶ EN is turned on. (User operation)
❷ IN is turned on. (User operation)
❸ The MOVP instruction is executed. (Controller operation)
❹ EN is turned off. (User operation)
❺ The MOVP instruction is executed. (Controller operation)

■When "No" is selected for "Use MC/MCR to Control EN"

When EN is turned off, operations of the instruction differ depending on the condition contact status ((1) in the following figure).



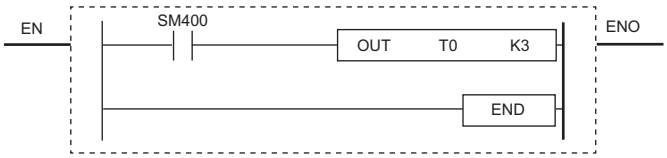
Sc: Scan
❶ EN is turned on. (User operation)
❷ IN is turned on. (User operation)
❸ The MOVP instruction is executed. (Controller operation)
❹ EN is turned off. (User operation)
❺ The MOVP instruction is executed when the condition contact is off immediately before EN is turned off at ❹. (Controller operation) (The MOVP instruction is not executed when the condition contact is on immediately before EN is turned off at ❹.)

Timer (low-speed/high-speed) and long timer

The following describes operations of the timer (low-speed/high-speed) and long timer.

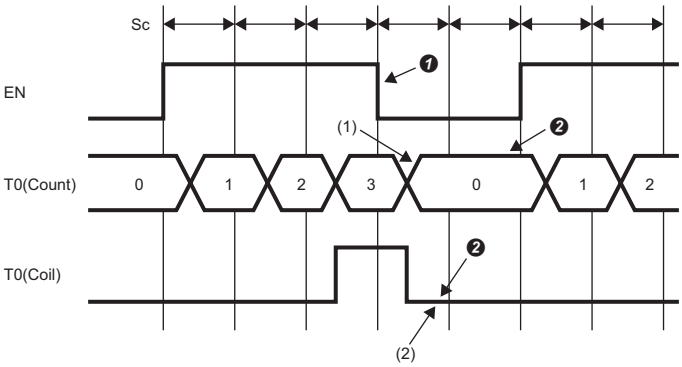
Ex.

A subroutine-type FB using a low-speed timer



■When "Yes" is selected for "Use MC/MCR to Control EN"

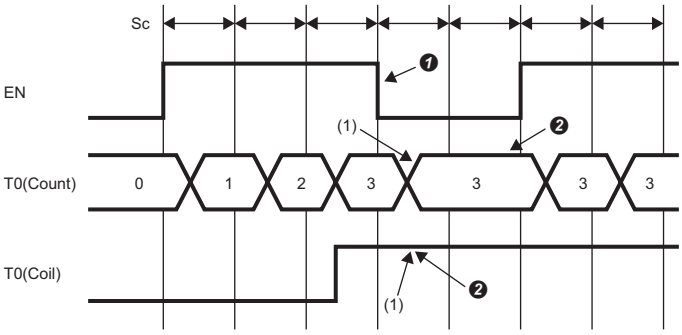
The count value becomes 0 ((1) in the following figure). The coil turns off ((2) in the following figure).



Sc: Scan
T0(Count): T0 (count value)
T0(Coil): T0 (coil)
❶ EN is turned off. (User operation)
❷ The coil turns off and the timer value and the count value are cleared. (Controller operation)

■When "No" is selected for "Use MC/MCR to Control EN"

The current count value and the current state of the coil remain unchanged. ((1) in the following figure).



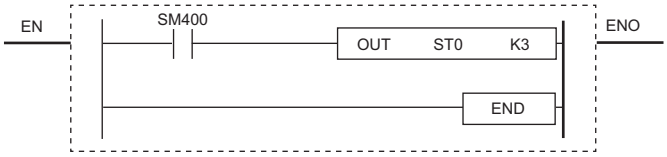
Sc: Scan
T0(Count): T0 (count value)
T0(Coil): T0 (coil)
❶ EN is turned off. (User operation)
❷ The values remain unchanged. (Controller operation)

Retentive timer (low-/high-speed), long retentive timer, counter, and long counter

The following describes operations of the retentive timer (low-speed/high-speed), long retentive timer, counter, and long counter.

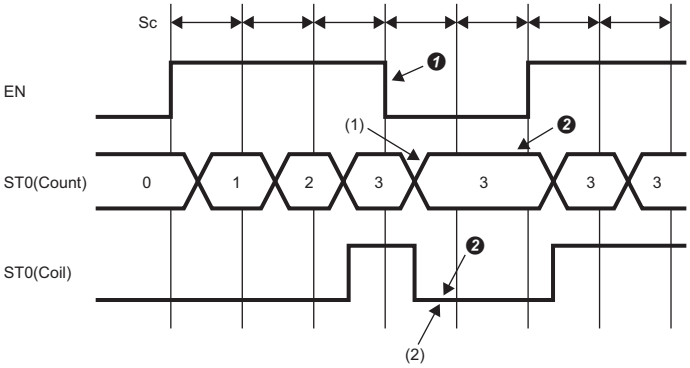
Ex.

A subroutine-type FB using a low-speed retentive timer



■When "Yes" is selected for "Use MC/MCR to Control EN"

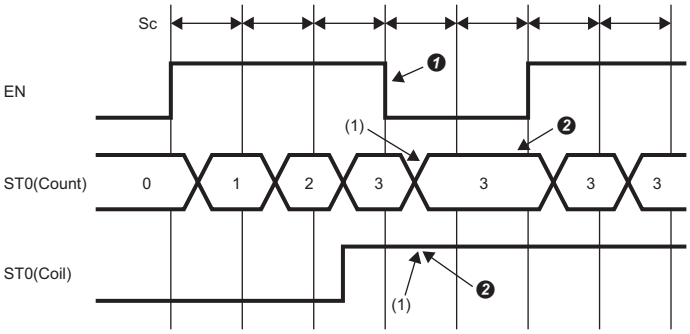
The current count value remain unchanged. ((1) in the following figure). The coil turns off ((2) in the following figure).



Sc: Scan
ST0(Count): T0 (count value)
ST0(Coil): T0 (coil)
❶ EN is turned off. (User operation)
❷ The coil turns off, but the current count value and the current state of the contact remain unchanged. (Controller operation)

■When "No" is selected for "Use MC/MCR to Control EN"

The current count value and the current state of the coil remain unchanged. ((1) in the following figure).



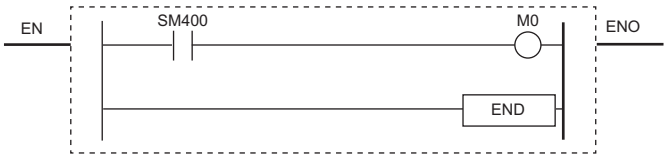
Sc: Scan
ST0(Count): T0 (count value)
ST0(Coil): T0 (coil)
❶ EN is turned off. (User operation)
❷ The values remain unchanged. (Controller operation)

Devices specified as the device part of the OUT instruction

The following describes operations of devices specified as the device part of the OUT instruction.

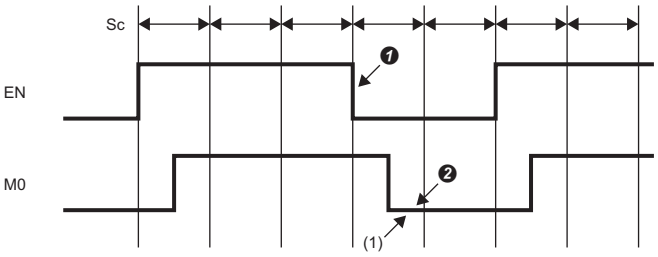
Ex.

A subroutine-type FB using M0 for the device part of the OUT instruction.



■When "Yes" is selected for "Use MC/MCR to Control EN"

M0 is forcibly turned off ((1) in the following figure).



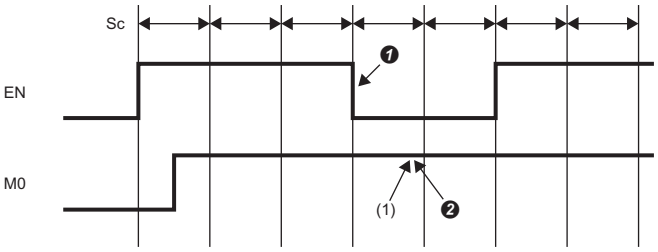
Sc: Scan

❶ EN is turned off. (User operation)

❷ The coil turns off. (Controller operation)

■When "No" is selected for "Use MC/MCR to Control EN"

The current state of M0 remains unchanged ((1) in the following figure).



Sc: Scan

❶ EN is turned off. (User operation)

❷ The state of coil remains unchanged. (Controller operation)

3 LADDER DIAGRAM

Ladder diagram is a programming language used to describe sequence control. Each ladder consists of contacts and coils and represents logical operations consisting of AND/OR in combinations of series and parallel.

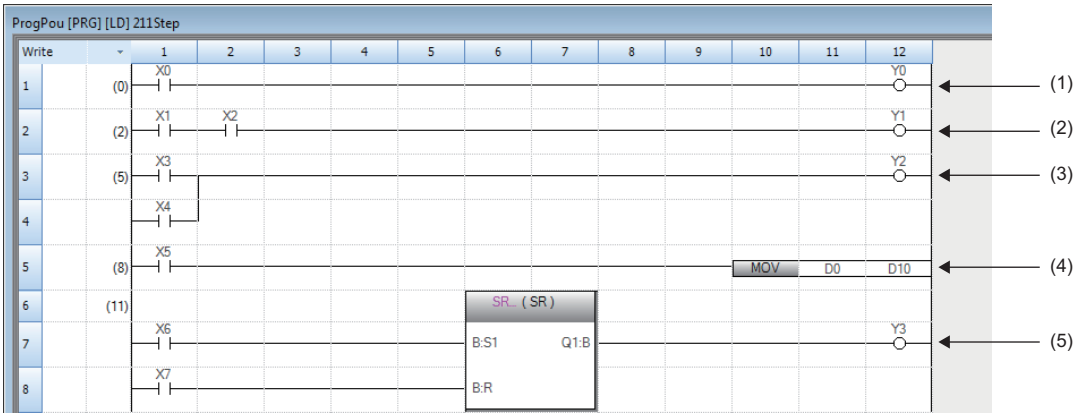
Point

This chapter describes the operation and specifications of the ladder diagram. For the operation method of the engineering tool for creating a ladder program, refer to the following.

GX Works3 Operating Manual

3.1 Configuration

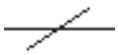
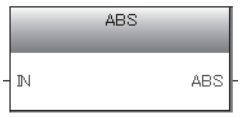
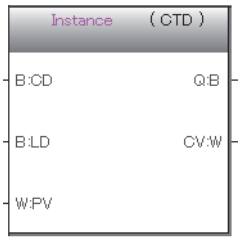
The following are the programs written in ladder diagram.



(1) Ladder program consisting of a contact and a coil
(2) Ladder program configured in series
(3) Ladder program configured in parallel
(4) Ladder program using an instruction
(5) Ladder program using a standard function/function block

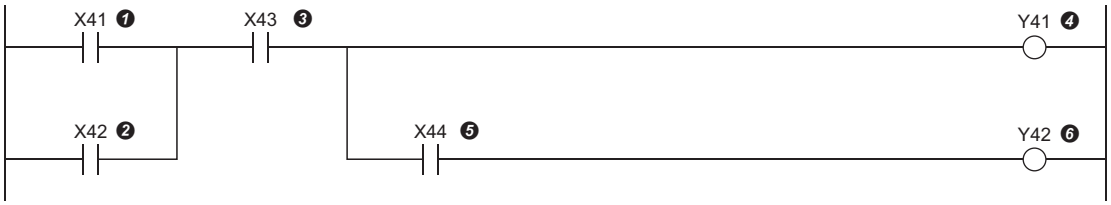
Ladder symbols

The following table lists the ladder symbols that can be used for programming in ladder diagram.

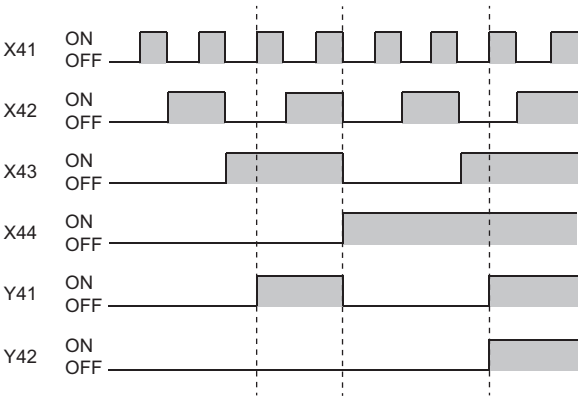
Item		Description
Normally open contact		Energized when the specified device or label is on.
Normally closed contact		Energized when the specified device or label is off.
Rising edge pulse		Energized on the rising edge (off to on) of the specified device or label.
Falling edge pulse		Energized on the falling edge (on to off) of the specified device or label.
Negated rising edge pulse		Energized when the specified device or label is off, on, or on the falling edge (on to off).
Negated falling edge pulse		Energized when the specified device or label is off, on, or on the rising edge (off to on).
Operation result rising edge pulse conversion		Energized on the rising edge (off to on) of the operation result. De-energized while the operation result is not on the rising edge.
Operation result falling edge pulse conversion		Energized on the falling edge (on to off) of the operation result. De-energized while the operation result is not on the falling edge.
Operation result inversion		Inverts the previous operation results.
Coil		Outputs the operation result to the specified device or label.
Instruction		Executes the instruction specified in "[ ]".
Loopback		When the number of contacts that can be created on a single ladder line is exceeded, the ladder is looped back with loopback source and destination symbols created on it.
Function		Executes a function. • How to create functions (GX Works3 Operating Manual) • Standard function (☞ Page 177 Standard Functions)
Function block		Executes a function block. • How to create function blocks (GX Works3 Operating Manual) • Standard function block (☞ Page 188 Standard Function Blocks) • Motion control function block (☞ Page 190 MOTION CONTROL FUNCTION BLOCK LIST) • Module function blocks (Function Block Reference for the module used)

Program execution order

The program is executed in order of the following numbers.



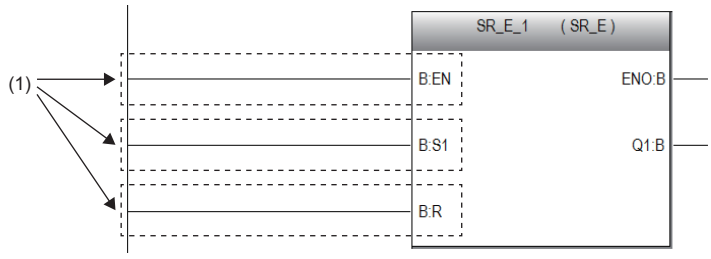
When the above program is executed, Y41 and Y42 turn on while X41 to X44 turn on or off as shown below.



Precautions for using a function block in ladder diagram

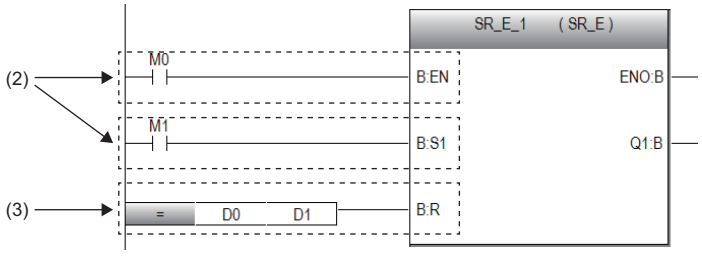
Precautions for directly connecting to an FB instance from the left rail

When EN and input variables (bit type) are directly connected to the left rail in the input circuit of the FB instance, the on/off state does not change.



(1): The on/off state does not change.

To change the on/off states of EN and the input variables (bit type), use a contact or an instruction equivalent to the contact.

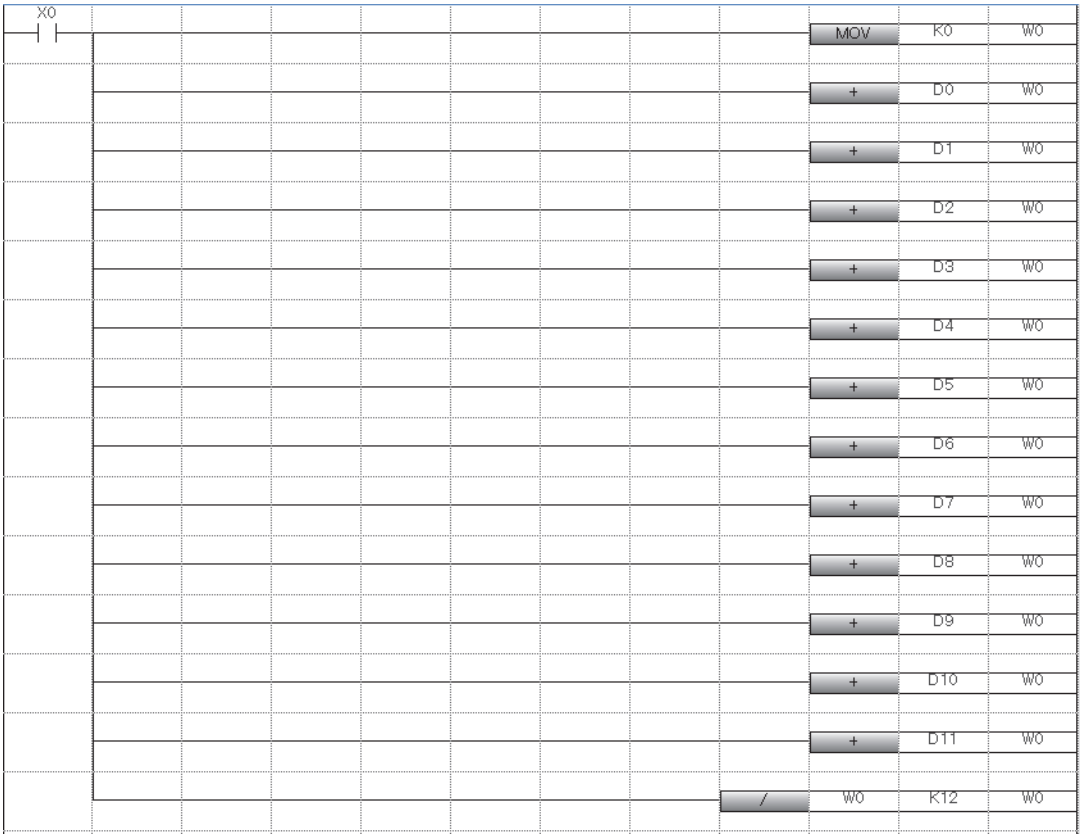


(2): Contact
(3): Instruction equivalent to a contact

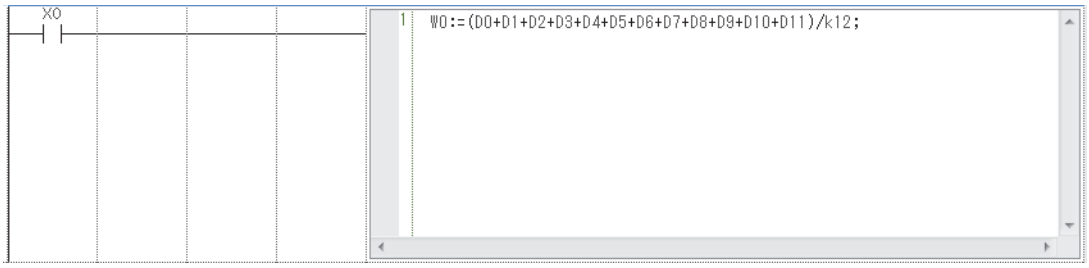
3.2 Inline ST

The inline ST is the function used to create an inline ST box that displays an ST program in the cell of the instruction corresponding to a coil in the ladder editor, and edit and monitor it.
This function enables to create numerical operations and string processing easily in ladder programs.

• Program that does not use the inline ST



• Program that uses the inline ST



Specifications

For the specifications of the ST program described to the inline ST, refer to the specifications of the ST language.
☞ Page 91 STRUCTURED TEXT LANGUAGE

Precautions

- Only one inline ST can be created on a single line of a ladder program.
- Both a function/function block and inline ST box cannot be used on a single line of a ladder program.
- Creating an inline ST box at the position of an instruction corresponding to a contact creates an inline ST box at the position of an instruction corresponding to a coil.
- Up to 20000 characters can be input in the inline ST. (A newline is counted as two characters.)
- Using a RETURN statement in the inline ST causes the processing in the inline ST box to end, instead of the processing of the program block.
- When the inline ST in the program of a function is used, no function block can be called from the inline ST.
- The inline ST uses the CJ instruction in conversion to control the operation of a program. When the contact of the inline ST is off, the processing in the inline ST is not executed by the CJ instruction. Thus, for example, the device that turned on by the assignment statement in the inline ST holds its output status even when the inline ST is not executed. For details on the CJ instruction, refer to the following.

☞ Page 532 CJ, SCJ, JMP

- In the inline ST, OUT, OUTH, or other instructions which refer to the previous execution status of the signal flow memory cannot be used.

📖User's manual for the controller used

3.3 Statements and Notes

Statements and notes can be used in ladder programs.

Statements

A statement is used to add a comment to a ladder block. Adding a comment makes it easy to understand the flow of processing.

There are three types of statements: line statement, P statement, and I statement.

The line statement can be displayed in the tree of the navigation window.

■Line statement

This type of statement adds a comment to the entire ladder block.

■P statement

This type of statement adds a comment to a pointer device.

■I statement

This type of statement adds a comment to an interrupt pointer device.

Notes

A note is used to add a comment to a coil and instruction in a program.

Adding a comment makes it easy to understand the contents of the coil and instruction.

Categories of statements and notes

Statements and notes are classified into two categories: "In PLC" and "In Peripheral".

Category	Type	Description
In PLC	• Line statement • P statement • I statement • Note	Strings of the description are not embedded in the program during conversion, but they are saved as attached information of the program. One line uses one step.
In Peripheral	• Line statement • P statement • I statement • Note	Strings of the description are not embedded in the program, but they are saved as attached information of the program. One line uses one step. A text that has been entered is automatically preceded by an asterisk "*".

4 STRUCTURED TEXT LANGUAGE

ST language is defined by International Standard IEC 61131-3 that defines the logic description system. ST language is a text language with a similar grammatical structure to C. This language is suitable for programming complicated processing that cannot be easily described by ladder diagram.

Point

This chapter describes the operation and specifications of the structured text language. For the operation method of the engineering tool for creating an ST program, refer to the following.

GX Works3 Operating Manual

The ST language supports control syntax, operational expressions, function blocks (FB), and functions (FUN), and can describe them as shown below.

Ex.

Control syntax such as selective branches by conditional statements and iteration statements

```
(*Control conveyors, Line A to C.*)
CASE  Line  OF
    1:
    Start_switch := TRUE; (*The conveyor starts.*)
    2:
    Start_switch := FALSE; (*The conveyor stops.*)
    3:
    Start_switch := TRUE; (*The conveyor stops with an alarm.*)
    ELSE
    Alarm_lamp := TRUE;
END_CASE;
IF Start_switch = TRUE THEN (*The conveyor starts and performs processing 100 times.*)
    FOR Count := 0 TO 100 BY 1 DO
      Count_No. := Count_No +1;
    END_FOR;
END_IF;
```

Ex.

Expression using operators (*, /, +, -, <, >, =)

```
D0 := D1 * D2 + D3 / D4 - D5 ;
IF  D0 > D10  THEN
     D0 := D10;
END_IF;
```

Ex.

Calling function blocks that have been defined

```
//FB data name: LINE1_FB
//Input variable: I_Test
//Output variable: O_Test
//Input/output variable: IO_Test
//FB label name: FB1
FB1(I_Test:= D0 , O_Test => D1 ,IO_Test:= D100 );
```

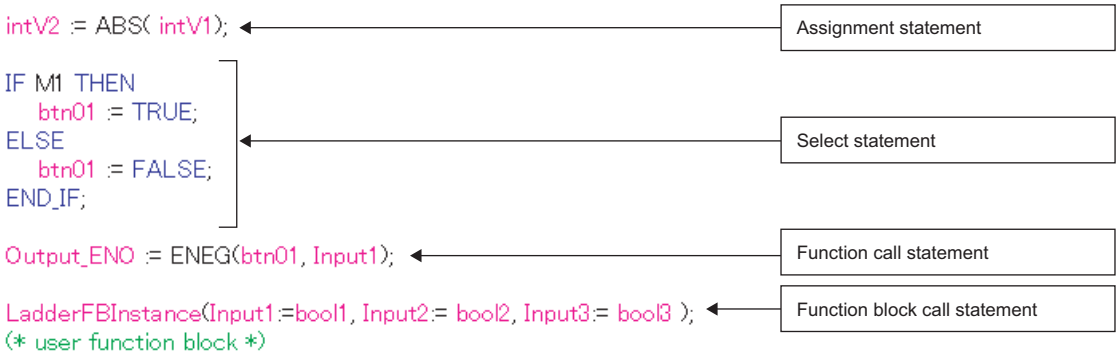
Ex.

Calling standard functions

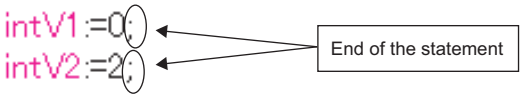
```
(*Convert BOOL data type to INT/DINT data type.*)
wLabel2 := BOOL_TO_INT (bLabel1);
```

4.1 Configuration

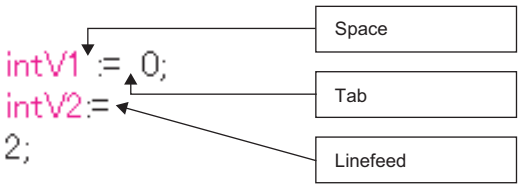
Programs written in ST language consist of operators and control statements.



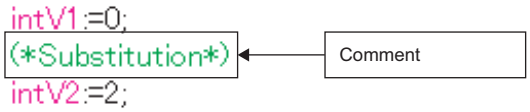
Each statement must end with a semicolon ";".



Spaces, tabs, and line feeds can be inserted between an operator and data.



Comments can be inserted into a program. Enclose a comment statement with "(*" and "*)".



Program components

An ST program consists of the following components.

Item		Example	Reference
Delimiter		;, (,)	☞ Page 93 Delimiter
Operator		+, -, <, >, =	☞ Page 93 Operators
Reserved word	Control statement	IF, CASE, WHILE, RETURN	☞ Page 95 Control statements
	Device	X40, Y50, M100, ZR0	🕮User's manual for the controller used
	Data type	BOOL, DWORD	🕮User's manual for the controller used
	Standard function	ADD, REAL_TO_STRING_E	☞ Page 177 Standard Functions
Constant		123, "abc"	☞ Page 104 Constants
Label		Switch_A	☞ Page 105 Labels and devices
Comment		(*Turn on.*)	☞ Page 107 Comments
Other symbols		One-byte space, line feed code, TAB code	—

- Write delimiters, operators, and reserved words in one-byte characters.
- For details on the reserved words, refer to the following.

🕮GX Works3 Operating Manual

Delimiter

ST language supports the following delimiters to make the program structure clear.

Symbol	Description
()	Parenthesized expression
[]	Specification of array element number
. (period)	Specification of structure or function block members
, (comma)	Argument separation
: (colon)	Device type specifier or CASE statement delimiter
; (semicolon)	Termination of statement
" (double quotation mark)	Notation of Unicode string
' (single quotation mark)	Notation of string (ASCII, Shift JIS)
.. (two periods)	Specification of integer range

Operators

The following table lists the operators, applicable data types, and operation result data types used in ST programs.

Operator	Applicable data type	Operation result type
*, /, +, -	ANY_NUM*1	ANY_NUM
<, >, <=, >=, =, <>	ANY_ELEMENTARY*2	Bit
MOD	ANY_INT	ANY_INT
AND, &, XOR, OR, NOT	ANY_BIT	ANY_BIT
**	ANY_REAL (base) ANY_NUM (exponent)*1	ANY_REAL

*1 Notation of constants with "WORD#" or "DWORD#" cannot be specified. (☞ Page 104 Notation of constants)

*2 WSTRING data type Unicode string cannot be specified.

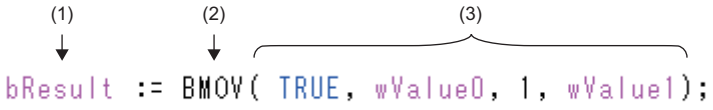
The following table lists the operators in descending order of priority.

Operator	Description	Example	Priority
( )	Parenthesized expression	(2+3)*(4+5)	1
Function ( )	Function call expression	CONCAT('AB','CD')	2
-	Sign inversion	-10	3
NOT	Bit type complement	NOT TRUE	
**	Exponentiation	3.0**4	4
*	Multiplication	10*20	5
/	Division	20/10	
MOD	Remainder	17 MOD 10	
+	Addition	1.4+2.5	6
-	Subtraction	3-2	
<, >, <=, >=	Comparison	10>20	7
=	Equality	T#26h=T#1d2h	8
<>	Inequality	8#15<>13	
&, AND	AND operation	TRUE AND FALSE	9
XOR	XOR operation	TRUE XOR FALSE	10
OR	OR operation	TRUE OR FALSE	11

- If one expression includes multiple operators with the same priority, operation is performed in order from the leftmost operator.
- Up to1024 operators can be used in a single expression.

Instruction

This section describes instructions used in the ST program.



(1) Specify a return value.
(The example shown above indicates that ENO is assigned to bResult. For instructions, the description of the return value can be omitted.)
(2) Specify an instruction name.
(3) Specify instruction arguments.

Omitting the specification of arguments in EN for instructions

For instructions, the specification of arguments in EN can be omitted.

In that case, the following program can be described as operation is performed assuming that TRUE is assigned to EN.

```
//Before omission
MOV(TRUE, SrcLabel, DstLabel);
//After omission
MOV(SrcLabel, DstLabel);
```

For cases other than instructions (such as standard functions), EN cannot be omitted.

Control statements

The following table lists the control statements that can be used in ST programs.

Item	Description	Reference
Assignment statement	Assignment statement	Page 95 Assignment statement
Subprogram control statement	Function block and function call statements	Page 97 Subprogram control statements
	RETURN statement	
Select statement	IF statement (IF THEN, IF ELSE, IF ELSIF)	Page 98 Select statements
	CASE statement	
Iteration statement	FOR statement	Page 98 Iteration statements
	WHILE statement	
	REPEAT statement	
	EXIT statement	

Write control statements in one-byte characters.

Assignment statement

Format	Description	Example
<Left side>:=<Right side>;	This statement assigns the result of the right side expression to the label or device on the left side. The data types of the result of the right side expression and the left side need to be the same.	intV1:=0; intV2:=2;

When using an array type label and structure label, note the data types of the left side and right side of the assignment statement.

For array type labels, the data type and the number of elements need to be the same between the left and right sides. When using array type labels, do not specify elements.

Ex.

intAry1:=intAry2;

For structure labels, the data type of the structure needs to be the same between the left and right sides.

Ex.

dutVar1:=dutVar2;

When the right side is the function call expression, the return value of the function is assigned to the left side. The following table lists examples to assign the return value to the label.

Function	Example
Having one input variable (example: ABS)	Output1 := ABS(Input1);
Having three input variables (example: MAX)	Output1 := MAX(Input1 , Input2 , Input3);
Having EN/ENO (standard functions) (example: MAX_E)	Output1 := MAX_E(boolEN , boolENO , Input1 , Input2 , Input3);
Except for standard functions (example: MOV)	boolENO := MOV(boolEN , Input1 , Output1); (The execution result of the function is ENO, and the first argument (variable1) is EN.)

■Automatic conversion of data type

When assignment or arithmetic operational expressions between different data types is described in ST language, one data type may be converted automatically.

Ex.

Example of automatic conversion

```
dintLabel1 := intLabel1;
// Assignment statement: Automatically convert the INT type variable (intLabel1) to a DINT type variable, and assign it to the DINT type variable (dintLabel1).
dintLabel1 := dintLabel2 + intLabel1;
// Arithmetic operational expression: Automatically convert the INT type variable (intLabel1) to a DINT type variable, and perform addition in DINT type.
DMOV(TRUE, wordLabel1, dwordLabel1);
// Instruction, function, and function block call statement: Automatically convert the WORD type input argument (wordLabel1) to a DWORD type variable, and transfer the data.
```

Type conversion is performed in assignment statements, VAR_INPUT part where input argument is passed to function blocks and functions (including instruction, standard functions, and standard function blocks), and arithmetic operational expressions.

To prevent data from being lost during type conversion, conversion is performed only from a smaller size data type to a larger size data type. Type conversion is performed for the following data types among the primitive data types.

Data type	Description
Word [signed]	When the data type is converted to the double word [signed], the value is automatically sign extended. When the data type is converted to the single-precision real number or double-precision real number, the value is automatically converted to the same value as the integer before conversion.*1
Word [unsigned]/bit string [16 bits]	When the data type is converted to the double word [unsigned]/bit string [32 bits] or double word [signed], the value is automatically zero extended.*2 When the data type is converted to the single-precision real number or double-precision real number, the value is automatically converted to the same value as the integer before conversion.*1
Double word [signed]	When the data type is converted to the double-precision real number, the value is automatically converted to the same value as the integer before conversion.
Double word [unsigned]/bit string [32 bits]	
Single-precision real number	When the data type is converted to the double-precision real number, the value is automatically converted to the same value.

*1 When 16-bit data (word [signed] or word [unsigned]/bit string [16 bits]) is passed to the input argument of which data type is ANY_REAL, the data is automatically converted to single-precision real number.

*2 When data (word [unsigned]/bit string [16 bits]) is passed to the input argument of which data type is ANY32, the data is automatically converted to double word [unsigned]/bit string [32 bits].

For the data types other than the above, use the type conversion functions.

Use the type conversion functions for the following conversions as well.

- Type conversion between integral data types with different signs
- Type conversion between data types both of which lose data

For the precautions on assignment of the arithmetic operation result, refer to the following.

☞ Page 99 When the result of an arithmetic operation is assigned

For the precautions on the use of devices, refer to the following.

☞ Page 106 When performing automatic conversion of data type with devices

Subprogram control statements

■Function block call statement

Format	Description
Instance name (input variable1:=variable1,...output variable1=>variable2,...);	Enclose the assignment statements to the input and output variables in '()' after the instance name. When using multiple variables, delimit individual assignment statements with a comma ','.
Instance name.input variable1:=variable1; ⋮ Instance name(); Variable2:=instance name.output variable1;	List the assignment statements to input and output arguments between function block call statements.

The following table lists the symbols used in the arguments of the function block call statements and the expressions that can be assigned.

Type	Description	Symbol used	Assignable expression
EN, VAR_INPUT	Input variable	:=	All expressions
ENO, VAR_OUTPUT, VAR_OUTPUT_RETAIN	Output variable	=>	Only variables
VAR_IN_OUT	Input/output variable	:=	Only variables

The execution result of a function block is stored by specifying an output variable after the instance name with a period "." and assigning the result to the specified variable.

Function block	FB definition	Example
Having one input variable and one output variable	FB name: FBADD FB instance name: FBADD1 Input variable1: IN1 Output variable1: OUT1	FBADD1(IN1:= Input1); Output1 := FBADD1.OUT1;
Having three input variables and two output variables	FB name: FBADD FB instance name: FBADD1 Input variable1: IN1 Input variable2: IN2 Input variable3: IN3 Output variable1: OUT1 Output variable2: OUT2	FBADD1(IN1:= Input1 ,IN2:= Input2 ,IN3:= Input3); Output1 := FBADD1.OUT1; Output2 := FBADD1.OUT2;

■Function call statement

Function without a return value, or includes VAR_OUTPUT variable in parameters can be executed as a statement by adding ";" (semicolon) after the function call statement.

Format	Description
Function name (variable1,variable2,...);	Enclose arguments in "()" following the function name. When using multiple variables, delimit them with a comma ",".

■RETURN statement

Control statement	Format	Description	Example
■RETURN	RETURN;	The statement is used to terminate a program, function block, or function during operation. When the RETURN statement is used in a program, control jumps to the step next to the last statement in the program. When the RETURN statement is used in a function block, control returns from the function block. When the RETURN statement is used in a function, control returns from the function. A single RETURN statement uses one point of the pointer type label in the system.	IF bool1 THEN RETURN; END_IF;

Select statements

Control statement	Format	Description	Example

■IF THEN	IF<boolean expression>THEN
	  <statement···>;
	END_IF;
		The statement is executed if the boolean expression (conditional formula) is TRUE. The statement is not executed if the boolean expression is FALSE.
		Any expression can be used for the boolean expression if it returns TRUE or FALSE as the boolean operation result of a single bit variable expression or a complicated expression including many variables.
			IF bool1 THEN
			  intV1 := intV1 + 1;
			END_IF;

■IF...ELSE	IF<boolean expression>THEN
	  <statement 1···>;
	  ELSE
	  <statement 2···>;
	END_IF;
		Statement 1 is executed if the boolean expression (conditional formula) is TRUE.
		If the boolean expression is FALSE, statement 2 is executed.
			IF bool1 THEN
			  intV3 := intV3 + 1;
			  ELSE
			  intV4 := intV4 + 1;
			END_IF;

■IF...ELSIF	IF<boolean expression 1>THEN
	  <statement 1···>;
	  ELSIF<boolean expression 2>THEN
	  <statement 2···>;
	  ELSIF<boolean expression 3>THEN
	  <statement 3···>;
	END_IF;
		Statement 1 is executed if boolean expression 1 (conditional formula) is TRUE. If boolean expression 1 is FALSE and boolean expression 2 is TRUE, statement 2 is executed.
		If boolean expressions 1 and 2 are FALSE and boolean expression 3 is TRUE, statement 3 is executed.
			IF bool1 THEN
			  intV1 := intV1 + 1;
			  ELSIF bool2 THEN
			  intV2 := intV2 + 2;
			  ELSIF bool3 THEN
			  intV3 := intV3 + 3;
			END_IF;

■CASE	CASE<integer expression>OF
	  <selected integer 1>:
	  <statement 1···>;
	  <selected integer 2>:
	  <statement 2···>;
	  ⋮
	  <selected integer n>:
	  <statement n···>;
	  ELSE
	  <statement n+1···>;
	END_CASE;
		The statement that has a selected integer value matching the value of the integer expression (conditional formula) is executed, and if no statement has a matching value, the statement following the ELSE statement is executed.
		The CASE statement can be used to execute a select statement according to a single integer value or the integer value of the result of a complicated expression.
			CASE intV1 OF
			  1:
			  bool1 := TRUE;
			  2:
			  bool2 := TRUE;
			  ELSE
			  intV1 := intV1 + 1;
			END_CASE;

Iteration statements

Control statement	Format	Description	Example

■FOR...DO	FOR<iteration variable initialization>
	  TO<final value>
	  BY<increase expression>DO
	  <statement···>;
	END_FOR;
		Data to be used as an iteration variable is initialized.
		One or more statements between the DO statement and the END_FOR statement are executed repeatedly, adding or subtracting the initialized iteration variable according to the increase expression until the final value is exceeded.
		The iteration variable after the FOR...DO statement is completed retains the value at the end of the processing.
			FOR intV1 := 0
			  TO 30
			  BY 1 DO
			  intV3 := intV1 + 1;
			END_FOR;

■WHILE...DO	WHILE<boolean expression>DO
	  <statement···>;
	END_WHILE;
		One or more statements are executed while the boolean expression (conditional formula) is TRUE.
		The boolean expression is determined before execution of the statement and, if it is FALSE, any statement in the DO...END_WHILE statement is not executed. Since <boolean expression> in the WHILE statement is only necessary to return TRUE or FALSE as the result, any expression that can be specified in <boolean expression> of the IF statement can be used.
			WHILE intV1 = 30 DO
			  intV1 := intV1 + 1;
			END_WHILE;

■REPEAT...UNTIL	REPEAT
	  <statement···>;
	  UNTIL<boolean expression>
	END_REPEAT;
		One or more statements are executed while the boolean expression (conditional formula) is FALSE.
		The boolean expression is determined after execution of the statement and, if the value is TRUE, any statement in the REPEAT...UNTIL statement is not executed. Since <boolean expression> in the REPEAT statement is only necessary to return TRUE or FALSE as the result, any expression that can be specified in <boolean expression> of the IF statement can be used.
			REPEAT
			  intV1 := intV1 + 1;
			  UNTIL intV1 = 30
			END_REPEAT;

Control statement	Format	Description	Example
■EXIT	EXIT;	This statement can be used only within an iteration statement to terminate the iteration statement in the middle of processing. When the EXIT statement is reached during execution of the iteration loop, the iteration loop processing after the EXIT statement is not executed. The program execution continues from the line next to the one where the iteration statement was terminated.	FOR intV1 := 0 TO 10 BY 1 DO IF intV1 > 10 THEN EXIT; END_IF; END_FOR;

Precautions

■When an assignment statement is used

- Up to 255 characters can be assigned to a string. If characters are assigned exceeding the valid range, a conversion error occurs.
- Timer type and counter type contacts and coils cannot be used at the left side of an assignment statement.
- Instances of function blocks cannot be used at the left side of an assignment statement. Use the input variable, output variable, or external variable of an instance.

■When the result of an arithmetic operation is assigned

When assigning the result of an arithmetic operation to a data type variable with larger data size, convert the variable of the arithmetic operational expression to the data type of the left side in advance.

Ex.

To assign the arithmetic operation result with a data size of 16 bits (INT type) to the 32-bit data type (DINT type):

```
varDint1 := varInt1 * 10; //The varInt1 is an INT type variable, and the varDint1 is a DINT type variable.
```

The result of the arithmetic operation will be the same data type as the input operand. For this reason, if the operation result of varInt1*10 in the above program exceeds the INT type range (-32768 to 32767), the operation result of overflow or underflow is assigned to varDint1.

In this case, convert the operand of the operational expression to the data type of the left side in advance.

```
varDint2 := INT_TO_DINT( varInt1 ); //An INT type variable is converted to a DINT type variable.
varDint1 := varDint2 * 10; //Multiplication is performed in DINT type, and the operation result is assigned.
```

■When a sign inversion operator is used in the arithmetic operational expression

If a sign inversion operator is used for the minimum value of each data type, the value remains unchanged.

For example, the minimum value of INT type data will be -(-32768) = -32768.

If a sign inversion operator is used for variables targeted for automatic conversion of data type, an intended operation result may not be obtained.

Ex.

When the value of varInt1 (INT type) is -32768 and the value of varDint1 (DINT type) is 0

```
varDint2 := -varInt1 + varDint1;
```

In this program, the value of (-varInt1) remains unchanged. Therefore, the value, -32768, is assigned to varDint2.

To use a sign inversion operator in an arithmetic operational expression, automatically convert data type before the arithmetic operation or do not use a sign inverted operator in the program.

Ex.

When data type is automatically converted before the arithmetic operation

```
varDint3 := varInt;
varDint2 := -varDint3 + varDint1;
```

Ex.

When a sign inversion operator is not used

```
varDint2 := varDint1 - varInt1
```

■When the data type is converted from single-precision real number to double-precision real number

When the type conversion function, REAL_TO_LREAL, is executed, an error may occur in the conversion result.
Consequently, when the data type is automatically converted or when a function with a return value of real number type (such as SIN function) is used as the right side of an assignment statement or an operand of arithmetic operational expression, an intended operation result may not be obtained.

Ex.

An error occurs.

```
varReal1 := -1234.567;
varLReal1 := ABS(varReal1);
```

In the above program, the data type of the return value of ABS(varReal1) is single-precision real number. Since the return value is converted to a double-precision real number and assigned to varReal1, an error occurs.
Create a program using a function with the data type same as the assignment target.

Ex.

No error occurs.

```
varLReal2 := -1234.567;
varLReal1 := ABS(varLReal2);
```

■When a bit type label is used

Once the boolean expression (conditional formula) is established in a select or iteration statement and if a bit type label is set to on in <statement>, the state of the label will be always on.

Ex.

Program that keeps the label status on

ST program	Ladder program performing the processing equivalent to ST program

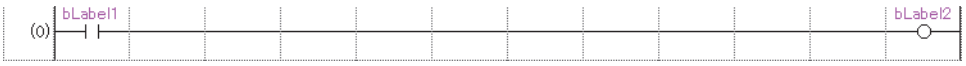
IF bLabel1 THEN
bLabel2 := TRUE;
END_IF;
	

To prevent the label from being always on, add a program that turns off the bit type label as shown below.

Ex.

Program that avoids the label from being always on

ST program*1	Ladder program performing the processing equivalent to ST program

IF bLabel1 THEN
bLabel2 := TRUE;
ELSE
bLabel2 := FALSE;
END_IF;
	

*1 The above program can also be described as follows.
bLabel2:=bLabel1;
or
OUT(bLabel1,bLabel2);
Note however that if the OUT instruction is used in the <statement>, the program will be in the same state as a program that keeps the label status on.

■When a timer function block or counter function block is used

In a select statement, a boolean expression (conditional formula) differs from the execution conditions of timer function blocks or counter function blocks.

Ex.

Timer function block

Program example before change

```
IF  bLabel1  THEN
    TIMER_100_FB_M_1 (Coil := bLabel2, Preset := wLabel3, ValueIn := wLabel4, ValueOut => wLabel5, Status => bLabel6);
END_IF;
(*When bLabel1 is on and bLabel2 is also on, counting starts.*)
(*When bLabel1 is on and bLabel2 is off, the counted value is cleared.*)
(*When bLabel1 is off and bLabel2 is on, counting stops. The counted value is not cleared.*)
(*When bLabel1 is off and bLabel2 is also off, counting stops. The counted value is not cleared.*)
```

Program example after change

```
TIMER_100_FB_M_1 (Coil := (bLabel1 & bLabel2), Preset := wLabel3, ValueIn := wLabel4 , ValueOut => wLabel5, Status => bLabel6);
```

Ex.

Counter function block

Program example before change

```
IF  bLabel1  THEN
    COUNTER_FB_M_1 (Coil := bLabel2, Preset := wLabel3, ValueIn := wLabel4, ValueOut => wLabel5, Status => bLabel6);
END_IF;
(*When bLabel1 is on and bLabel2 is on/off, the value is incremented by one.*)
(*When bLabel1 is off and bLabel2 is on/off, the value is not counted.*)
(*The counting operation does not depend on the on/off status of bLabel1.*)
```

Program example after change

```
COUNTER_FB_M_1 (Coil := (bLabel1 & bLabel2), Preset := wLabel3, ValueIn := wLabel4, ValueOut => wLabel5, Status => bLabel6);
```

The above examples of programs before change cause problems because the statement related to the timer or counter is not executed unless the select statement is established.

To operate the timer or counter on the basis of the bLabel1 condition and bLabel1 AND condition, do not use control statements but use only function blocks.

The timer and counter can be operated by using the programs after change.

■When a FOR...DO statement is used

- A structure member or array element cannot be used for the iteration variable.
- Match the type used in the iteration variable with the types of <final value expression> and <increase expression>.
- <Increase expression> can be omitted. If omitted, <increase expression> is assumed to be 1.
- If 0 is assigned to <increase expression>, the portions after the FOR statement may no longer be executed or an infinite loop may occur.
- In the FOR...DO statement, the iteration variable count processing is performed after execution of <statement···>in the FOR statement. If the count processing is executed in such a way that it exceeds the maximum value of the iteration variable data type or is below the minimum value, an infinite loop occurs.

■When instructions executed at the rising edge or falling edge are used

- If the execution condition of the IF or CASE statement has become FALSE or CASE mismatch, the execution result of the rising edge or falling edge in the IF or CASE statement during the previous scan will not be initialized and will be carried over to the next execution.
- After an iteration statement (FOR, WHILE, or REPEAT statement) ends, the execution result of the rising edge or falling edge in the iteration statement during the previous scan will not be initialized and will be carried over to the next execution.
- The following table lists the operations of when instructions executed at the rising edge or falling edge used in the IF or CASE statement.

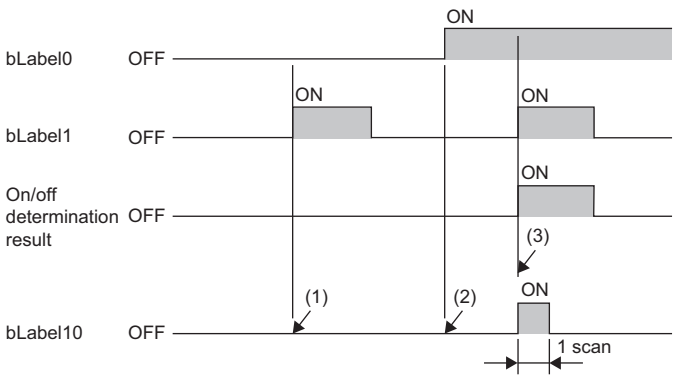
Condition			Operation result		
Conditional formula of IF or CASE statement	**Instruction execution condition (EN)**	**On/off determination result of the instruction in the last scan**	**On/off determination result of the instruction**	**Instruction at the rising edge**	**Instruction at the falling edge**
TRUE or CASE match	TRUE	On	On	Not executed	Not executed
		Off	On	Executed	Not executed
	FALSE	On	Off	Not executed	Executed
		Off	Off	Not executed	Not executed
FALSE or CASE mismatch	TRUE	On	Off	Not executed	Not executed*1
		Off	Off	Not executed	Not executed
	FALSE	On	Off	Not executed	Not executed*1
		Off	Off	Not executed	Not executed

*1 On the falling edge (on to off), the instruction is not executed because the condition of the IF or CASE statement is not satisfied.

Ex.

When the PLS instruction (execution condition: rising edge) is used in the IF statement

```
IF bLabel0 THEN
    PLS(bLabel1, bLabel10);
END_IF;
```



(1) When bLabel0 is off (the conditional formula of the IF statement is FALSE), the on/off determination result will be off. The PLS instruction is not executed. (The bLabel10 remains off.)

(2) When bLabel0 is on (the conditional formula of the IF statement is TRUE) and bLabel1 is off (the instruction execution condition is off), the on/off determination result will be off. The PLS instruction is not executed. (The bLabel10 remains off.)

(3) When bLabel0 is on (the conditional formula of the IF statement is TRUE) and bLabel1 is also on (the instruction execution condition is on), the on/off determination result will be off to on (rising edge). The PLS instruction is executed. (The bLabel10 is on for one scan.)

- To execute instructions at the rising edge or falling edge in an iteration statement (FOR, WHILE, or REPEAT statement), use the edge relay (V) or perform index modification. In this case, one point of the edge relay (V) is used for each instruction that uses the edge relay (V) in the system. For this reason, secure the edge relay (V) for the number of instructions used in addition to the number of points used in the iteration statement.

Ex.

When instructions executed at the rising edge or falling edge are used in the FOR statement

Example of using the edge relay (V) in one location
(The edge relay (V) is used up to a total of 11 points (V0 to V10 for the INC instruction).)

```
FOR  Z0 := 0  TO  9  BY  1  DO
    INC(EGP(M100Z0 , V0Z0) , D100Z0);
END_FOR;
```

Example of using the edge relay (V) in two locations
(The edge relay (V) is used up to a total of 22 points (V0 to V10 for the INC instruction, V11 to V21 for the DEC instruction).)

```
FOR  Z0 := 0  TO  9  BY  1  DO
    INC(EGP(M100Z0 , V0Z0) , D100Z0);
    DEC(EGF(M200Z0 , V11Z0) , D200Z0);
END_FOR;
```

■When the master control instruction is used

Operations when the master control is off will be as follows.

- A statement in the select statement (IF or CASE statement) or iteration statement (FOR, WHILE, or REPEAT statement) performs no processing.
- For a statement outside the select statement or iteration statement, no processing is performed if it is an assignment statement, and a statement itself is not executed if it is other than assignment statement.

Ex.

Statement in the select statement (IF statement)

```
MC(M0 , N1 , M1);  //Master control is off.
IF  M2  THEN
    M3 := M4; // M3 retains the value in the last scan because no processing is performed when the master control is off.
END_IF;
M20 := MCR(M0, N1);
```

Ex.

Statement (bit assignment statement) outside the select statement or iteration statement

```
MC(M0 , N1 , M1);  //Master control is off.
M3 := M4;  //M3 retains the value in the last scan because no processing is performed when the master control is off.
M20 := MCR(M0, N1);
```

Ex.

Statement (OUT instruction) outside the select statement or iteration statement

```
MC(M0 , N1 , M1);  //Master control is off.
OUT(M2, M3);  //M3 turns off because the instruction is not executed when the master control is off.
M20 := MCR(M0, N1);
```

Constants

Notation of constants

In the ST program, the following notation of constant is available in addition to the standard notation.

Applicable data type	Type	Notation[*1]	Example
Bit	Boolean	Add "BOOL#" before the boolean value to be used.	BOOL#1, BOOL#0 BOOL#TRUE, BOOL#FALSE
Word [unsigned]/bit string [16 bits]	Binary	Add "UINT#2#" or "WORD#2#" before a binary number.[*2]	UINT#2#10101010 WORD#2#10101010
	Octal	Add "UINT#8#" or "WORD#8#" before an octal number.[*2]	UINT#8#3370 WORD#8#3370
	Decimal	Add "UINT#" or "WORD#" before a decimal number.[*2]	UINT#123 WORD#123
	Hexadecimal	Add "UINT#16#" or "WORD#16#" before a hexadecimal number.[*2]	UINT#16#FF WORD#16#FF
Double word [unsigned]/bit string [32 bits]	Binary	Add "UDINT#2#" or "DWORD#2#" before a binary number.[*2]	UDINT#2#1100110011001100 DWORD#2#1100110011001100
	Octal	Add "UDINT#8#" or "DWORD#8#" before an octal number.[*2]	UDINT#8#33703370 DWORD#8#33703370
	Decimal	Add "UDINT#" or "DWORD#" before a decimal number.[*2]	UDINT#456789 DWORD#456789
	Hexadecimal	Add "UDINT#16#" or "DWORD#16#" before a hexadecimal number.[*2]	UDINT#16#FFFF DWORD#16#FFFF
Word [signed]	Binary	Add "INT#2#" before a binary number.	INT#2#01010101
	Octal	Add "INT#8#" before an octal number.	INT#8#3370
	Decimal	Add "INT#" before a decimal number.	INT#-123
	Hexadecimal	Add "INT#16#" before a hexadecimal number.	INT#16#1F
Double word [signed]	Binary	Add "DINT#2#" before a binary number.	DINT#2#0011001100110011
	Octal	Add "DINT#8#" before an octal number.	DINT#8#33703370
	Decimal	Add "DINT#" before a decimal number.	DINT#-456789
	Hexadecimal	Add "DINT#16#" before a hexadecimal number.	DINT#16#1F1F
Single-precision real number	Real number	Add "REAL#" before a real number.	REAL#2.34
	Real number (exponential notation)		REAL#1.0E6
Double-precision real number	Real number	Add "LREAL#" before a real number.	LREAL#-2.34
	Real number (exponential notation)		LREAL#1.001E16
String	STRING	Enclose a string (ASCII, Shift JIS) in single quotation marks (').	'ABC'
String [Unicode]	WSTRING	Enclose a Unicode string in double quotation marks (").	"ABC"

*1 Notation of constants are not case-sensitive. Notation of constants with K, H, or E cannot be used simultaneously.

*2 ANY_NUM of operand for arithmetic operator, ANY_NUM of arguments for a function call statement, function block call statement, and function call expression, use the notation of constants with "UINT#" or "UDINT#". For the notation of constants with "WORD#" or "DWORD#", an error occurs during the conversion because the data type is recognized as a bit string.

For the notations of constants other than the above, refer to the following.

User's manual for the controller used

Point

In the notation of binary, octal, decimal, hexadecimal, and real numbers, the numbers can be separated using an underscore "_" to make programs easy to see. The binary number notation of double word [unsigned] can be described as follows.

UDINT#2#1100_1100_1100_1100

The numerical value separation using an underscore "_" is ignored in program processing.

Labels and devices

Specification method

Labels and devices can be directly described on ST programs. Labels and devices can be used for the left or right side of an expression or as an argument or return value of a standard function/function block.

For details on labels and devices, refer to the following.

User's manual for the controller used

■Notation of devices with a type specifier

A word device can be used as any data type by adding a device type specifier to its name.

Device specifier	Data type	Example	Description of example
None	Generic data type ANY16. Word [signed], when only devices are used in such as an arithmetic operational expression. However, when it is specified as a device without a type specifier in a FUN/FB argument, it becomes an argument-defined data type.	D0	D0 without a type specifier
:U	Word [unsigned]/bit string [16 bits]	D0:U	D0 specified as the word [unsigned]/bit string [16 bits]
:D	Double word [signed]	D0:D	D0 and D1 specified as the double word [signed]
:UD	Double word [unsigned]/bit string [32 bits]	D0:UD	D0 and D1 specified as the double word [unsigned]/bit string [32 bits]
:E	Single-precision real number	D0:E	D0 and D1 specified as the single-precision real number
:ED	Double-precision real number	D0:ED	D0, D1, D2, and D3 specified as the double-precision real number

The following devices can use a device type specifier.

- Data register (D)
- Link register (W)
- Link direct device (J□\W□)
- Module access device (U□\G□)
- CPU buffer memory access device (U3E□\G□)
- File register (R/ZR)
- Refresh data register (RD)

A device type specifier cannot be added to a digit-specified or indirect specified device.

■Device specification method

Devices can be specified in the following methods.

- Index modification
- Indirect specification
- Bit specification (☞ Page 31 Bit data)
- Digit specification (☞ Page 32 16-bit data (word data), ☞ Page 35 32-bit data (double word data))

For details on index modification and indirect specification, refer to the following.

User's manual for the controller used

Precautions

- The pointer type cannot be used in ST programs.
- When the timer, counter, or retentive timer device is used as current value, the data type will be the word [unsigned]/bit string [16 bits]. When the long timer, long counter, or long retentive timer device is used as current value, the data type will be the double word [unsigned]/bit string [32 bits].
- When performing assignment using digit specification, match the data type between the left and right sides.

Ex.

D0:=K5X40;

In this example, a program error occurs because K5X40 is the double word type and D0 is the word type.

- If the right side is greater than the left side when performing assignment using digit specification, data is transferred to within the range of the number of applicable points of the left side.

Ex.

K5X40:=2#1011_1101_1111_0111_0011_0001;

In this example, K5X40 has 20 applicable points and therefore 1101_1111_0111_0011_0001 (20 digits) is assigned to K5X40.

- When using the current value (such as TNn) of the counter (C), timer (T), or retentive timer (ST) as a type other than the word [unsigned]/bit string [16 bits], or when using the current value (such as LTNn) of the long counter (LC), long timer (LT), or long retentive timer (LST) as a type other than the double word [unsigned]/bit string [32 bits], use the type conversion functions.

Ex.

varInt:=WORD_TO_INT(T0);(*A type conversion function is used.*)

■When performing automatic conversion of data type with devices

Add a device type specifier when using a word device in a data type other than word [signed]. (☞ Page 105 Notation of devices with a type specifier)

Ex.

When transferring the values of D2 and D3 to dwordLabel1, a double word [unsigned] label

```
//Example of when adding a device type specifier and transferring the values in a correct data type
dwordLabel1 := D2:UD;
//Since the data type of D2: UD (D2 with a device type specifier) is double word [unsigned], the values of D2 and D3 are transferred to dwordLabel1.

//Example of an unintended transfer result
dwordLabel1 := D2;
//Since the data type of D2 without a device type specifier is word [signed], the data type is automatically converted into double word [unsigned] and data is transferred to dwordLabel1.
//Therefore, the value of D3 is not transferred but only the value of D2 is transferred.
```

Comments

The following table lists the comments that can be used in ST programs.

Type	Symbol	Description	Example
Single-line comment	//	The portion from the start symbol "//" to the end of the line is regarded as a comment.	//comment
Multiple-line comment	(**)	The portion from the start symbol "(*" to the end symbol "*)" is regarded as a comment. A line feed can be inserted in a comment.	■No line feed (*comment*) ■With a line feed (*line-1 comment line-2 comment*)
	/**/	The portion from the start symbol "/*" to the end symbol "*/" is regarded as a comment. A line feed can be inserted in a comment.	■No line feed /*comment*/ ■With a line feed /*line-1 comment line-2 comment*/

Do not write a comment containing an end symbol in a multiple-line comment.

5 FBD/LD

FBD/LD (function block diagram/ladder diagram) is a graphic language which describes programs by connecting blocks that perform predefined processing, variable elements, and constant elements along the flow of data and signals.

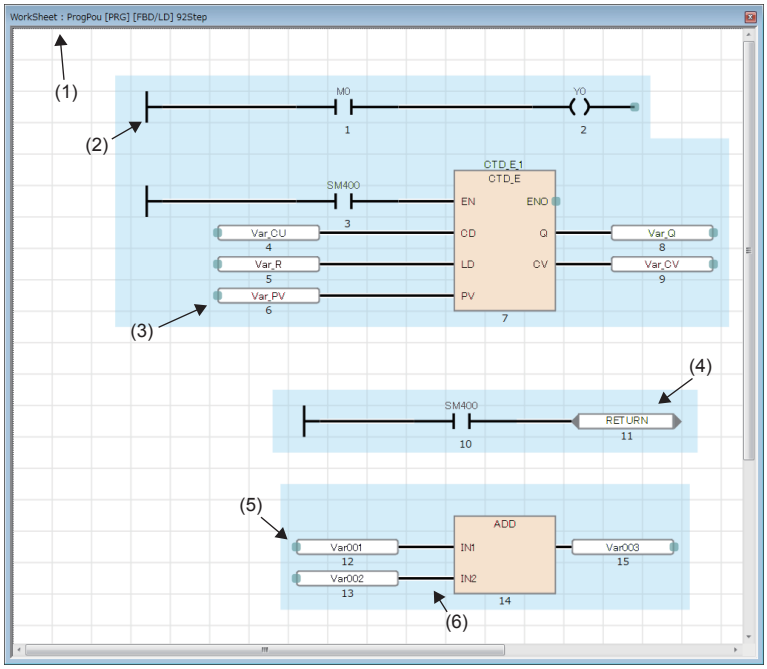
Point

- This chapter describes the operation and specifications of the FBD/LD. For the operation method of the engineering tool for creating an FBD/LD program, refer to the following.

GX Works3 Operating Manual

5.1 Configuration

The following are the programs written in FBD/LD.



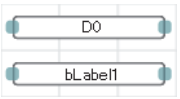
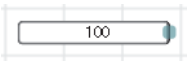
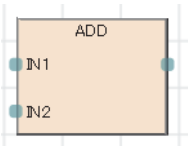
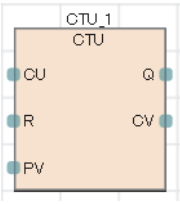
(1) Worksheet
(2) LD element
(3) FBD element
(4) Common element
(5) Connection point
(6) Connection line

In FBD/LD programs, data flows from the output point of a function block, function, variable (label or device), or constant to the input point of another function block or variable.

Program elements

FBD elements

The following table lists FBD elements consisting an FBD/LD program.

Item		Description
Variable		Stores a value. A specific data type is assigned to each variable, and only data of the assigned data type are stored. Labels and devices can be specified as a variable.
Constant		Outputs a specified value.
Function (FUN)		Executes a function. • How to create functions (GX Works3 Operating Manual) • Standard function (☞ Page 177 Standard Functions)
Function block (FB)		Executes a function block. • How to create function blocks (GX Works3 Operating Manual) • Standard function block (☞ Page 188 Standard Function Blocks) • Motion control function block (☞ Page 190 MOTION CONTROL FUNCTION BLOCK LIST) • Module function blocks (Function Block Reference for the module used)

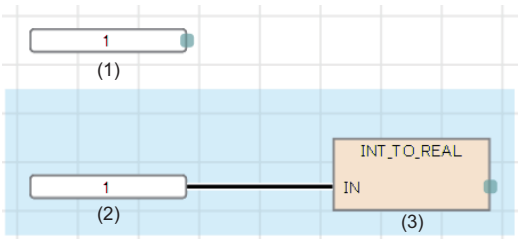
■Data type of constant elements

The data type of the value input to a constant is not determined when the value is input. It is determined when the constant element is connected to another FBD element with a connection line. The data type will be the same as that of the connected FBD element.

Ex.

When the constant value is 1

The possible data types are BOOL, WORD, DWORD, INT, DINT, REAL, and LREAL, but the data type of the input value cannot be determined at this point. The data type is determined when the constant is connected to another FBD element.



(1) The data type has not been determined.
(2) INT data type
(3) INT data type

■Automatic conversion of data type

If the data types differ between connected elements, one data type may be converted automatically.

To prevent data from being lost during type conversion, conversion is performed only from a smaller size data type to a larger size data type. The data type automatic conversion processing in FBD/LD is same as that in ST language. For details, refer to the following.

☞ Page 96 Automatic conversion of data type

■Input/output points of a function

- Connect all input points of a function with another element.
- The data type is assigned to each input variable and output variable of a function. Match the data type of an element connected to the input point or output point with that of the input variable or output variable.

■Precautions for connecting the output variables of the function with EN and function block with EN to another element

When ENO of the function with EN and function block with EN are FALSE (not executed), the program operation differs depending on the element connected.

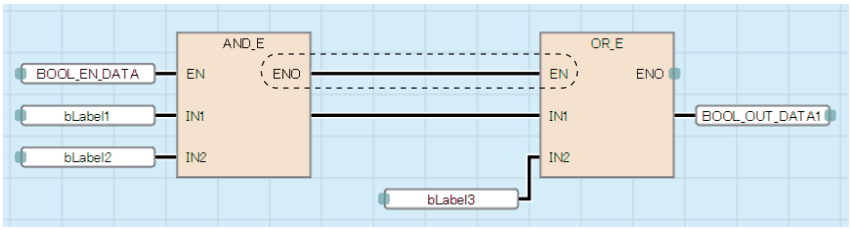
Element connected	Program operation when ENO of the function with EN and function block with EN are FALSE (not executed)
• Function • Function block	The values of the input variables connected to the output variables of the function with EN and function block with EN are undefined.
Coil	The values of the coils connected to the output variables of the function with EN and function block with EN are FALSE. When inverse coils are connected, the values are TRUE.
Other than above	The values of the elements connected to the output variables of the function with EN and function block with EN are not changed. (The last values are input.)

Unintended operation may occur if the output variable of the function block with EN is directly connected to the input variable of other functions or function blocks.

To prevent an undefined value from being used, refer to either of the following examples for the connection.

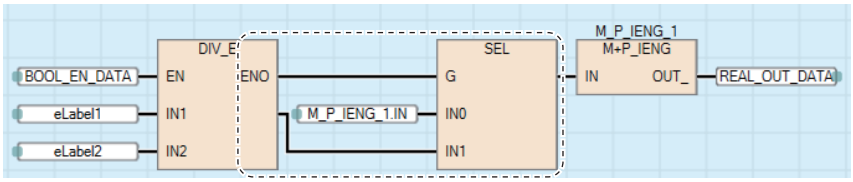
Ex.

Connect elements so that the function with EN or function block with EN is the connection destination. Connect ENO (connection source) to EN (connection destination) as well.



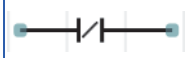
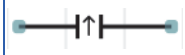
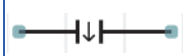
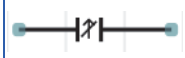
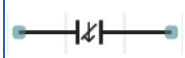
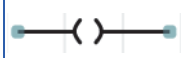
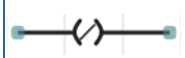
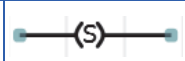
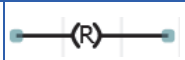
Ex.

Connect elements using the SEL (selection function) so that the last value is input if ENO is FALSE.



LD elements

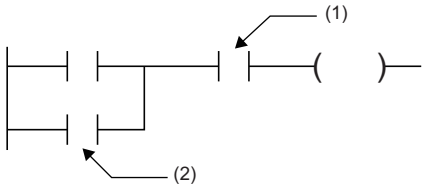
The following table lists the LD elements that can be used in FBD/LD programs.

Item		Description
Left rail		An element that indicates a rail. A start point of ladder program. The output of the left rail is always on.
Normally open contact		Energized when the specified device or label is on.
Normally closed contact		Energized when the specified device or label is off.
Rising edge pulse		Energized on the rising edge (off to on) of the specified device or label.
Falling edge pulse		Energized on the falling edge (on to off) of the specified device or label.
Negated rising edge pulse		Energized when the specified device or label is off, on, or on the falling edge (on to off).
Negated falling edge pulse		Energized when the specified device or label is off, on, or on the rising edge (off to on).
Coil		Outputs the operation result to the specified device or label.
Inverse coil		When the operation result turns off, the specified device or label turns on.
Set coil		When the operation result turns on, the specified device or label turns on. The device or label keeps on state even after the operation result turns off.
Reset coil		When the operation result turns on, the specified device or label turns off. If the operation result is off, the status of the device or label does not change.

■AND operation and OR operation of contact elements

At each contact, an AND operation or OR operation is performed depending on the connection status in the ladder diagram, and the operation result is output.

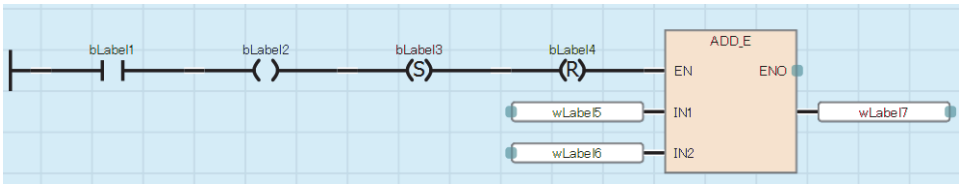
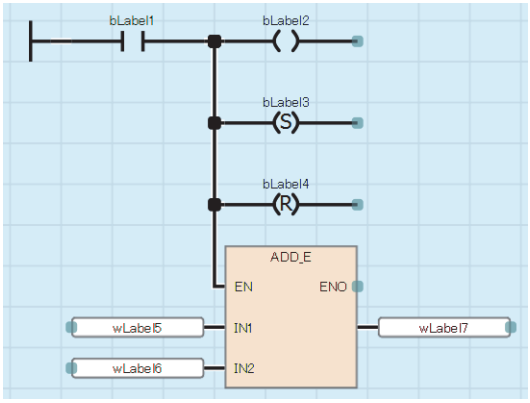
- Series connection (1): An AND operation is performed with the previous operation result.
- Parallel connection (2): An OR operation is performed with the previous operation result.



(1) Contact connected in series
(2) Contact connected in parallel

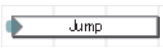
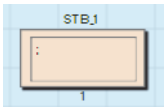
■When another program element is connected to a connection point of coil for output

When another program element is connected to a connection point of coil for output in series, the operation is the same as that of the parallel connection.

Item	Description
Program example	
Operation at the above example	

Common elements

The following table lists the common elements that can be used in FBD/LD programs.

Item		Description
Jump		Jumps the execution processing to a jump element. Processing between this element and a jump label is not performed. Whether to perform jump processing or not is controlled by inputting on/off information to the element. On: Jump processing performed Off: Jump processing not performed
Jump label		A jump destination of a jump element in the same program. When the jump processing is performed, the program continues from the processing located after the jump label.
Connector		Used as a substitute for a connection line. Processing moves to a connector element used as a pair. One or more input connectors can be used as a pair of one output connector.
Return		Stops the execution processing after this element. Use the element not to execute the program, functions, and function blocks after the element. Whether to perform return processing or not is controlled by inputting on/off information to the element. On: Return processing performed Off: Return processing not performed
Comment		Used to write a comment.
Inline ST		Displays an ST program in the FBD/LD editor. Double-clicking an inserted inline ST element displays the ST editor for editing or monitoring an ST program. For details, refer to the following. ☞ Page 118 Inline ST

■Jump element

- If the timer whose coil is on is jumped, the time cannot be measured correctly.
- A jump label can be located before the jump element in the program sequence. If located, create a program so that the watchdog timer setting value is not exceeded. (How to exit from a loop must be considered.)
- Specify a pointer type local label for a jump element and a jump label. The structure members cannot be used.
- The CJ, SCJ, and JMP instructions (for pointer branch) cannot be used. Use a jump element when jump processing is required.
- Processing cannot be jumped to/from the outside of the program block.

Jump related operation	Availability
Jumping to the outside of the program block*1	Not supported
Jumping from the outside of the program block*1	Not supported
Calling a subroutine program	Supported
Being called as a subroutine program	Not supported

*1 Branching by using the BREAK instruction is included.

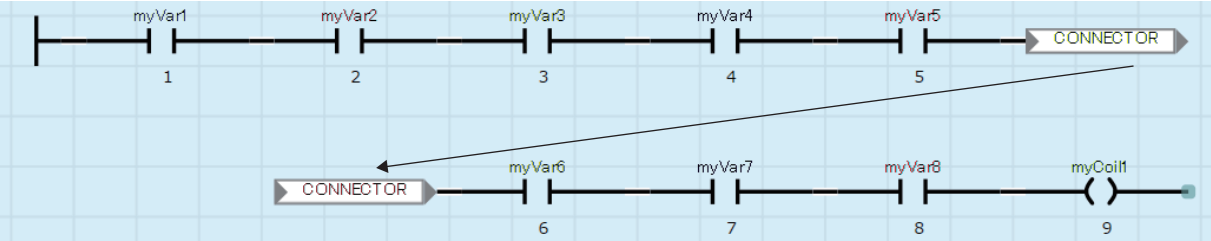
■Return element

- The operation of a return element differs depending on the POU (program, function, or function block) used.

POU used	Description
Program	Execution of the program is stopped.
Function	Execution of the function is stopped, and processing returns to the step next to the instruction that called the function.
Function block	Execution of the function block is stopped, and processing returns to the step next to the instruction that called the function block.

■Connector element

A connector element is used to place the program within the area to be displayed in the FBD/LD editor or to be printed.

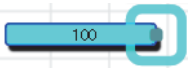
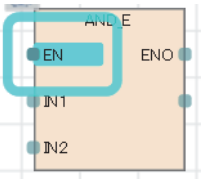
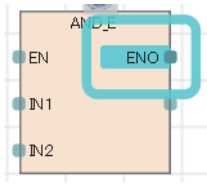
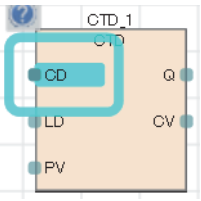
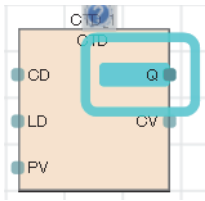


Connection line

A connection line is a line that connects the end points of FBD element, LD element, and common element.
When connected, the value is passed from the left end to the right end of the line. The data types of the program elements connected must be identical, or support automatic data type conversion.

Connection point

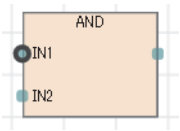
A connection point is an end point of FBD element, LD element, and common element for connecting them to create an FBD/LD program.
The left-side end point is for input, and the right-side end point is for output.

Item	Connection point for input	Connection point for output
Contact		
Coil		
Variable		
Constant	—	
Function		 The return value of the function is not displayed.
Function block		

When connected, the connection point is hidden.

■I/O point inversion

The input value to a program element or the output value from a program element can be inverted at the connection point.
The connection point where the value is inverted (FALSE to TRUE or TRUE to FALSE) is displayed with a black circle.

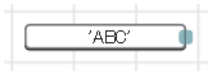
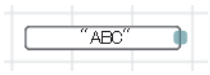


The value of the following data types can be inverted: BOOL, WORD, DWORD, ANY_BIT, and ANY_BOOL.

Constant

Notation of constants

The following table lists the notations of strings in FBD/LD programs.

Data type		Notation	Example
String	STRING	Enclose a string (ASCII, Shift JIS) in single quotation marks (').	
String [Unicode]	WSTRING	Enclose a Unicode string in double quotation marks (").	

For the notations of constants other than the above, refer to the following.

📖User's manual for the controller used

Labels and devices

Specification method

Labels and devices can be directly described on FBD/LD programs. Labels and devices can be used as an input point or output point of a program element and an argument or return value of a standard function/function block.

For details on labels and devices, refer to the following.

📖User's manual for the controller used

■Notation of devices with a type specifier

A word device can be used as any data type by adding a device type specifier to its name.

The device type specifiers and applicable devices are the same as those for ST programs. For details, refer to the following.

☞ Page 105 Notation of devices with a type specifier

If the data type of a word device is not specified, it will be determined by the device type.

Word device	Data type
Current value of the timer (TN), current value of the retentive timer (STN), current value of the counter (CN)	WORD
Current value of the long timer (LTN), current value of the long retentive timer (LSTN), current value of the long counter (LCN)	DWORD
Long index register (LZ)	DINT
Other than above	ANY16

Precautions

■When labels are used

- Local devices cannot be used as an array index. To use local devices as an array index, assign the target device to another device, and specify the assigned device.

■When performing automatic conversion of data type with devices

Add a device type specifier when using a word device in a data type other than word [signed]. (☞ Page 116 Notation of devices with a type specifier)

Ex.

When the values of D2 and D3 are transferred to dwordLabel1, a double word [unsigned] label

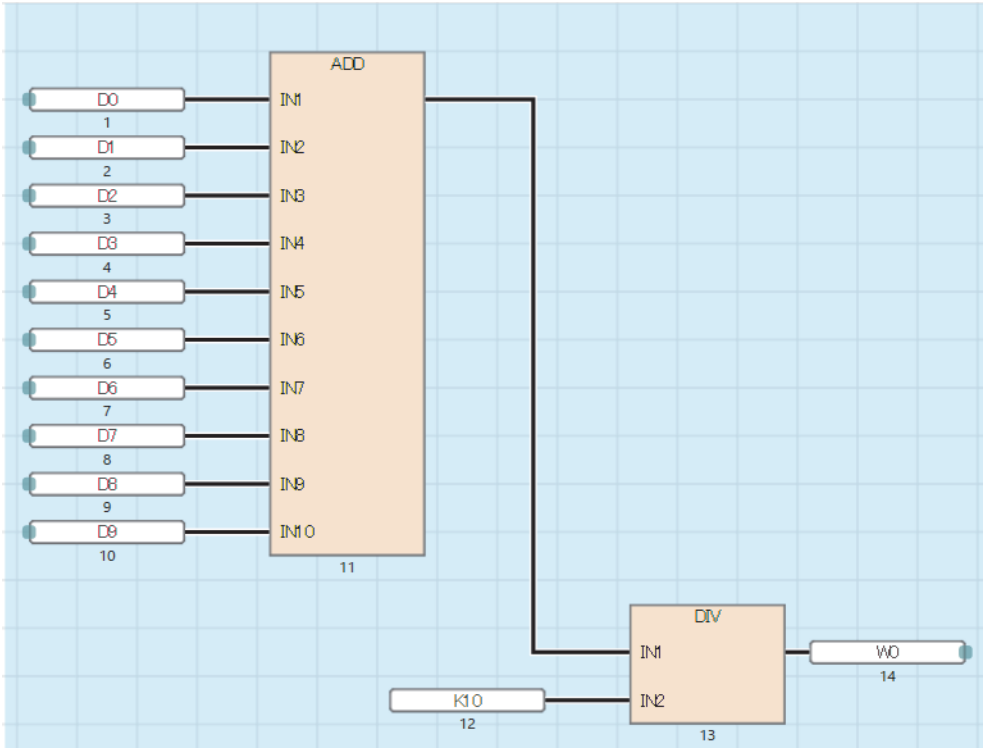
Example of when adding a device type specifier and transferring the values in a correct data type	Example of an unintended transfer result
	
Since the data type of D2:UD (D2 with a device type specifier) is double word [unsigned], the values of D2 and D3 are transferred to dwordLabel1.	Since the data type of D2 without a device type specifier is word [signed], the data type is automatically converted into double word [unsigned] and data is transferred to dwordLabel1. Therefore, the value of D3 is not transferred but only the value of D2 is transferred.

5.2 Inline ST

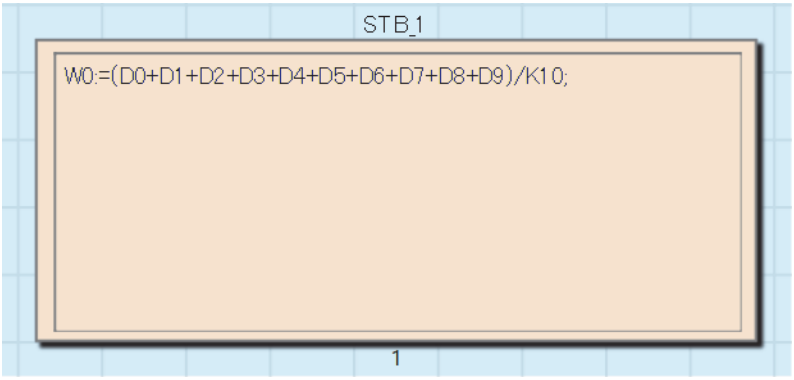
The inline ST is the function used to create an inline ST element that displays an ST program in the FBD/LD editor, and edit and monitor it.

This function enables to create complex numerical operations and string processing easily in FBD/LD programs.

- Program that does not use the inline ST



- Program that uses the inline ST



Specifications

Double-clicking an inserted inline ST element displays the ST editor for editing or monitoring an ST program.

For the specifications of the ST program described to the inline ST, refer to the specifications of the ST language.

☞ Page 91 STRUCTURED TEXT LANGUAGE

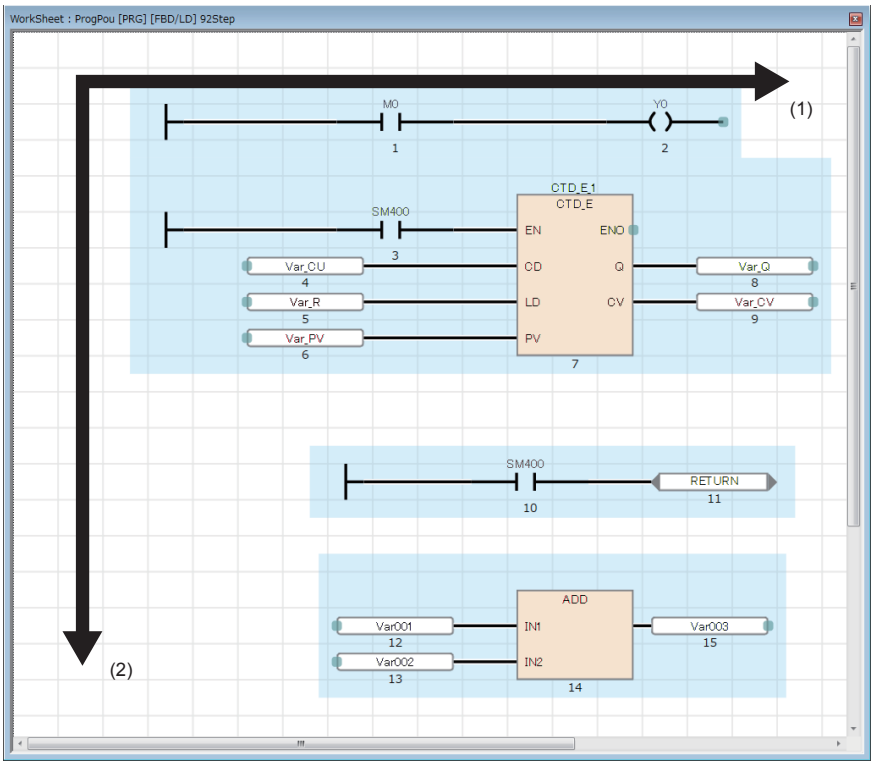
Precautions

- Up to 64 inline ST elements can be inserted into a single POU of an FBD/LD program.
- Using a RETURN statement in the inline ST causes the processing in the inline ST element to end, instead of the processing of the program block.
- Since inline ST elements have no connection points, an inline ST element inserted into an FBD/LD program is executed every scan.

5.3 Program Execution Order

Execution order of program elements

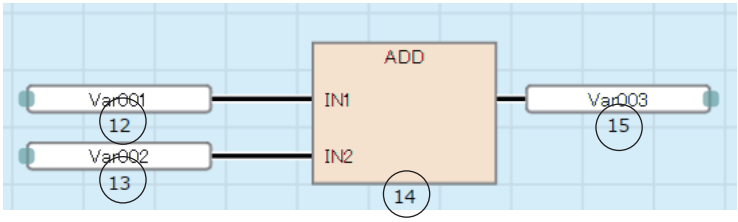
The execution order of program elements in the FBD/LD editor is determined by the location and connecting status.



(1) Program elements are executed from left to right.

(2) Program elements are executed from top to bottom.

The execution order is displayed under each program element. After the program is converted, the determined execution order is displayed.

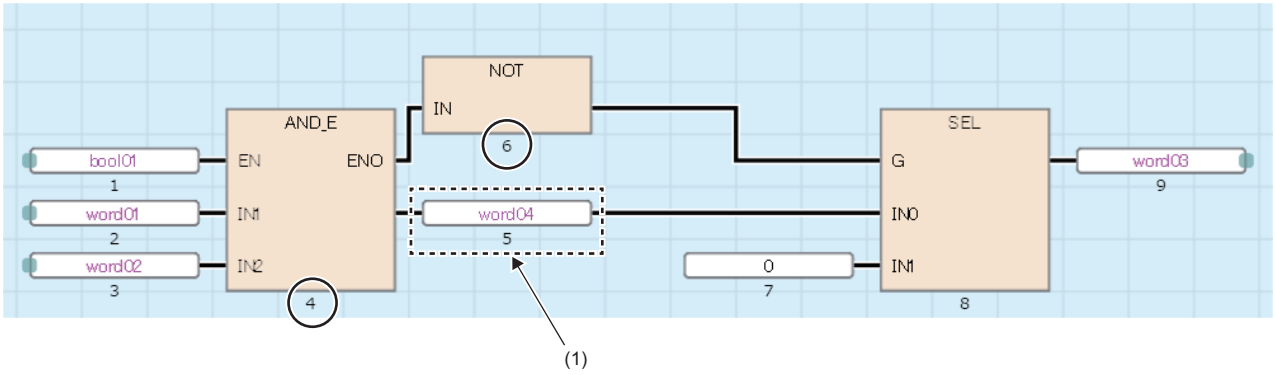


Precautions

For a program that uses functions, do not directly connect the return value of a function and an input variable of another function, but connect a variable element between them.

Ex.

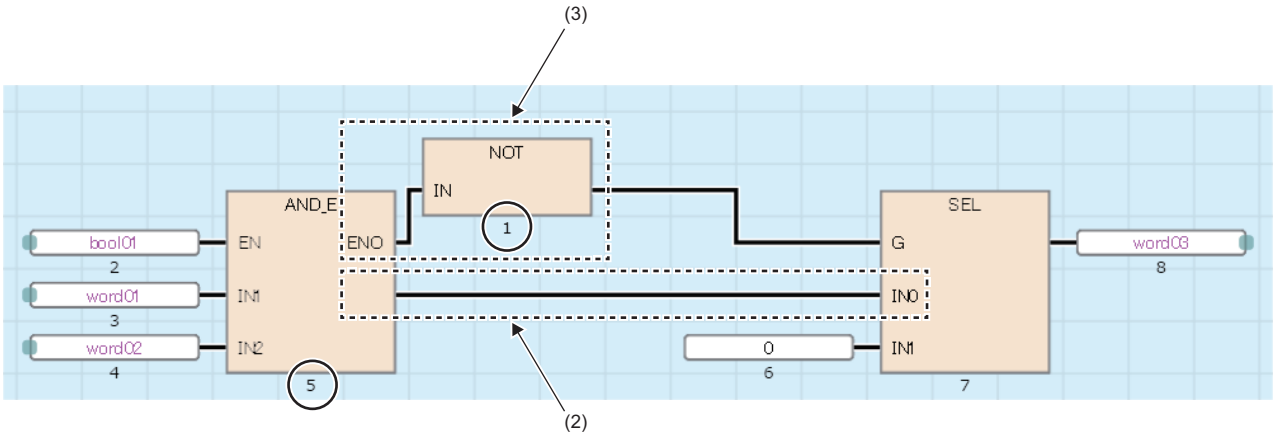
When the variable element (1) is connected between the return value and input variable



Connecting the return value of a function and the input variable of another function directly may lead an execution order to an unintended one.

Ex.

An unintended execution order



Since both the output variable (3) that comes from another program element and the return value (2) of the program element arranged on the left are connected as inputs of the program element arranged on the right, the execution order has changed.

PART 2 LISTS OF INSTRUCTIONS AND FUN/FB

This part consists of the following chapters.

6 INSTRUCTION LIST

7 NETWORK INSTRUCTION LIST

8 STANDARD FUNCTION/STANDARD FUNCTION BLOCK LISTS

9 MOTION CONTROL FUNCTION BLOCK LIST

10 NETWORK FUNCTION BLOCK LIST

6 INSTRUCTION LIST

The following table summarizes how to read the instruction lists.

Item	Description
Instruction symbol	An instruction name
Processing details	An overview of the instruction
Reference	Section where detailed information is described

6.1 Sequence Instructions

Contact instructions

■Operation start, series connection, parallel connection

Instruction symbol	Processing details	Reference
LD	Outputs the on/off information of the specified device as the operation result. (Normally open contact operation start instruction)	Page 202 LD, LDI, AND, ANI, OR, ORI
LDI	Outputs the on/off information of the specified device as the operation result. (Normally closed contact operation start instruction)	
AND	Performs an AND operation between the on/off information of the specified device and the previous operation result, and output the operation result. (Normally open contact series connection instruction)	
ANI	Performs an AND operation between the on/off information of the specified device and the previous operation result, and output the operation result. (Normally closed contact series connection instruction)	
OR	Performs an OR operation between the on/off information of the specified device and the previous operation result, and output the operation result. (Single normally open contact parallel connection instruction)	
ORI	Performs an OR operation between the on/off information of the specified device and the previous operation result, and output the operation result. (Single normally closed contact parallel connection instruction)	

■Pulse operation start, pulse series connection, pulse parallel connection

Instruction symbol	Processing details	Reference
LDP	Turns on only at the rising edge (off to on) of the specified bit device. (Rising edge pulse operation start instruction)	Page 204 LDP, LDF, ANDP, ANDF, ORP, ORF
LDF	Turns on only at the falling edge (on to off) of the specified bit device. (Falling edge pulse operation start instruction)	
ANDP	Performs an AND operation with the previous operation result. (Rising edge pulse series connection instruction)	
ANDF	Performs an AND operation with the previous operation result. (Falling edge pulse series connection instruction)	
ORP	Performs an OR operation with the previous operation result. (Rising edge pulse parallel connection instruction)	
ORF	Performs an OR operation with the previous operation result. (Falling edge pulse parallel connection instruction)	

■Pulse NOT operation start, pulse NOT series connection, pulse NOT parallel connection

Instruction symbol	Processing details	Reference
LDPI	Turns on when the specified device is off, on, or at the falling edge (on to off). (Rising edge pulse NOT operation start instruction)	Page 207 LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI
LDFI	Turns on when the specified device is at the rising edge (off to on), off, or on. (Falling edge pulse NOT operation start instruction)	
ANDPI	Performs an AND operation with the previous operation result. (Rising edge pulse NOT series connection instruction)	
ANDFI	Performs an AND operation with the previous operation result. (Falling edge pulse NOT series connection instruction)	
ORPI	Performs an OR operation with the previous operation result. (Rising edge pulse NOT parallel connection instruction)	
ORFI	Performs an OR operation with the previous operation result. (Falling edge pulse NOT parallel connection instruction)	

Association instructions

■Ladder block series/parallel connection

Instruction symbol	Processing details	Reference
ANB	Performs AND operations between logical blocks (series connection between logical blocks)	Page 210 ANB, ORB
ORB	Performs OR operations between logical blocks (series connection between logical blocks)	

■Inverting the operation result

Instruction symbol	Processing details	Reference
INV	Inverts the operation result up to just before the INV instruction.	Page 211 INV

■Converting the operation result into a pulse

Instruction symbol	Processing details	Reference
MEP	Turns on at the rising edge (off to on) of the operation result up to the MEP instruction.	Page 212 MEP, MEF
MEF	Turns on at the falling edge (on to off) of the operation result up to the MEF instruction.	

■Converting the edge relay operation result into a pulse

Instruction symbol	Processing details	Reference
EGP	Stores the operation result up to the EGP instruction in the edge relay (V). The instruction turns on at the rising edge (off to on) of the operation result.	Page 213 EGP, EGF
EGF	Stores the operation result up to the EGF instruction in the edge relay (V). The instruction turns on at the falling edge (on to off) of the operation result.	

Output instructions

■Out (excluding the timer, counter, and annunciator)

Instruction symbol	Processing details	Reference
OUT	Outputs the operation result to the specified device.	Page 215 OUT

■Timer, long timer

Instruction symbol	Processing details	Reference
OUT T	Starts time measurement when the operation result up to the OUT instruction is on. When time is up, the normally open contact turns on (continuity state) and the normally closed contact turns off (non-continuity state). • OUT T: Low-speed timer instruction • OUTH T: High-speed timer instruction • OUT ST: Low-speed retentive timer instruction • OUTH ST: High-speed retentive timer instruction • OUT LT: Low-speed long timer instruction • OUT LST: Low-speed long retentive timer instruction	Page 217 OUT T, OUTH T, OUT ST, OUTH ST
OUTH T		
OUT ST		
OUTH ST		
OUT LT		Page 220 OUT LT, OUT LST
OUT LST		

■Counter, long counter

Instruction symbol	Processing details	Reference
OUT C	Increments the current counter value (count value) by one when the operation result up to the OUT instruction turns on. When the count value reaches the set value, the normally open contact of the counter turns on (continuity state) and the normally closed contact turns off (non-continuity state). • OUT C: Counter • OUT LC: Long counter	Page 223 OUT C
OUT LC		Page 225 OUT LC

■Annunciator

Instruction symbol	Processing details	Reference
OUT F	Outputs the operation result up to the OUT F instruction to the specified annunciator.	Page 227 OUT F

■Setting devices (excluding annunciator)

Instruction symbol	Processing details	Reference
SET	Turns on the specified bit.	Page 228 SET

■Resetting devices (excluding annunciator)

Instruction symbol	Processing details	Reference
RST	Turns off the specified device. For the timer and counter, the instruction clears the current value to 0 and turns off the contact or coil.	Page 230 RST

■Setting/resetting annunciator

Instruction symbol	Processing details	Reference
SET F	Turns on the specified annunciator.	Page 232 SET F
RST F	Turns off the specified annunciator.	Page 234 RST F

■Rising/falling edge output

Instruction symbol	Processing details	Reference
PLS	Turns on the specified device for one scan on the rising edge (off to on) of the execution command.	Page 236 PLS
PLF	Turns on the specified device for one scan on the falling edge (on to off) of the execution command.	Page 238 PLF

■Inverting the bit device output

Instruction symbol	Processing details	Reference
FF	Inverts the status of the specified device.	Page 240 FF

■Converting the direct access output into a pulse

Instruction symbol	Processing details	Reference
DELTA	Converts the specified direct access output (DY) into pulse output.	Page 241 DELTA(P)
DELTAP		

Shift instructions

■Shifting bit devices

Instruction symbol	Processing details	Reference
SFT	Shifts the on/off state of the device area just before the one specified to the specified device area, and turns off the shift source device.	Page 243 SFT(P)
SFTP		

Master control instructions

■Setting/resetting a master control

Instruction symbol	Processing details	Reference
MC	Starts a master control.	Page 245 MC, MCR
MCR	Ends a master control.	

Termination instructions

■Ending the main routine program

Instruction symbol	Processing details	Reference
FEND	Separates the main routine program from subroutine programs and interrupt programs in a program file.	Page 249 FEND

■Ending the sequence program

Instruction symbol	Processing details	Reference
END	Indicates the end of a program.	Page 250 END

Stop instruction

■Stopping the sequence program

Instruction symbol	Processing details	Reference
STOP	Stops the operation of the controller. (The operation of this instruction is the same as setting the switch of the controller to the STOP position.)	Page 252 STOP

No operation instruction

■No operation

Instruction symbol	Processing details	Reference
NOPLF	This instruction is a no-operation instruction and has no impact on the previous operations.	Page 253 NOPLF

6.2 Basic Instructions

Comparison operation instructions

■Comparing 16-bit binary data

Instruction symbol	Processing details	Reference
LD=, AND=, OR=	Compares the two sets of 16-bit binary data specified. (Devices are used as normally open contacts.)	Page 254 LD□(_U), AND□(_U), OR□(_U)
LD=_U, AND=_U, OR=_U		
LD<>, AND<>, OR<>		
LD<>_U, AND<>_U, OR<>_U		
LD>, AND>, OR>		
LD>_U, AND>_U, OR>_U		
LD<=, AND<=, OR<=		
LD<=_U, AND<=_U, OR<=_U		
LD<, AND<, OR<		
LD<_U, AND<_U, OR<_U		
LD>=, AND>=, OR>=		
LD>=_U, AND>=_U, OR>=_U		

■Comparing 32-bit binary data

Instruction symbol	Processing details	Reference
LDD=, ANDD=, ORD=	Compares the two sets of 32-bit binary data specified. (Devices are used as normally open contacts.)	Page 256 LDD□(_U), ANDD□(_U), ORD□(_U)
LDD=_U, ANDD=_U, ORD=_U		
LDD<>, ANDD<>, ORD<>		
LDD<>_U, ANDD<>_U, ORD<>_U		
LDD>, ANDD>, ORD>		
LDD>_U, ANDD>_U, ORD>_U		
LDD<=, ANDD<=, ORD<=		
LDD<=_U, ANDD<=_U, ORD<=_U		
LDD<, ANDD<, ORD<		
LDD<_U, ANDD<_U, ORD<_U		
LDD>=, ANDD>=, ORD>=		
LDD>=_U, ANDD>=_U, ORD>=_U		

■Comparing 16-bit binary block data

Instruction symbol	Processing details	Reference
BKCMP=, BKCMP<>, BKCMP>, BKCMP<=, BKCMP<, BKCMP>=	Compares the two sets of 16-bit binary block data specified.	Page 258 BKCMP□(P)(_U)
BKCMP=P, BKCMP<>P, BKCMP>P, BKCMP<=P, BKCMP<P, BKCMP>=P		
BKCMP=_U, BKCMP<>_U, BKCMP>_U, BKCMP<=_U, BKCMP<_U, BKCMP>=_U		
BKCMP=P_U, BKCMP<>P_U, BKCMP>P_U, BKCMP<=P_U, BKCMP<P_U, BKCMP>=P_U		

■Comparing 32-bit binary block data

Instruction symbol	Processing details	Reference
DBKCMP=, DBKCMP<>, DBKCMP>, DBKCMP<=, DBKCMP<, DBKCMP>=	Compares the two sets of 32-bit binary block data specified.	Page 261 DBKCMP□(P)(_U)
DBKCMP=P, DBKCMP<>P, DBKCMP>P, DBKCMP<=P, DBKCMP<P, DBKCMP>=P		
DBKCMP=_U, DBKCMP<>_U, DBKCMP>_U, DBKCMP<=_U, DBKCMP<_U, DBKCMP>=_U		
DBKCMP=P_U, DBKCMP<>P_U, DBKCMP>P_U, DBKCMP<=P_U, DBKCMP<P_U, DBKCMP>=P_U		

Arithmetic operation instructions

■Adding/subtracting 16-bit binary data

Instruction symbol	Processing details	Reference
+	Adds the two sets of 16-bit binary data specified. (Two operands)	Page 264 +(P)(_U) [when two operands are set]
+P		
+_U		
+P_U		
+	Adds the two sets of 16-bit binary data specified. (Three operands)	Page 266 +(P)(_U) [when three operands are set]
+P		
+_U		
+P_U		
-	Performs subtraction between the two sets of 16-bit binary data specified. (Two operands)	Page 268 -(P)(_U) [when two operands are set]
-P		
-_U		
-P_U		
-	Performs subtraction between the two sets of 16-bit binary data specified. (Three operands)	Page 270 -(P)(_U) [when three operands are set]
-P		
-_U		
-P_U		

■Adding/subtracting 32-bit binary data

Instruction symbol	Processing details	Reference
D+	Adds the two sets of 32-bit binary data specified. (Two operands)	Page 272 D+(P)(_U) [when two operands are set]
D+P		
D+_U		
D+P_U		
D+	Adds the two sets of 32-bit binary data specified. (Three operands)	Page 274 D+(P)(_U) [when three operands are set]
D+P		
D+_U		
D+P_U		
D-	Performs subtraction between the two sets of 32-bit binary data specified. (Two operands)	Page 276 D-(P)(_U) [when two operands are set]
D-P		
D-_U		
D-P_U		
D-	Performs subtraction between the two sets of 32-bit binary data specified. (Three operands)	Page 278 D-(P)(_U) [when three operands are set]
D-P		
D-_U		
D-P_U		

■Multiplying/dividing 16-bit binary data

Instruction symbol	Processing details	Reference
*	Multiplies the two sets of 16-bit binary data specified.	Page 280 *(P)(_U)
*P		
*_U		
*P_U		
/	Performs division between the two sets of 16-bit binary data specified.	Page 282 /(P)(_U)
/P		
/_U		
/P_U		

■Multiplying/dividing 32-bit binary data

Instruction symbol	Processing details	Reference
D*	Multiplies the two sets of 32-bit binary data specified.	Page 284 D*(P)(_U)
D*P		
D*_U		
D*P_U		
D/	Performs division between the two sets of 32-bit binary data specified.	Page 286 D/(P)(_U)
D/P		
D/_U		
D/P_U		

■Adding/subtracting BCD 4-digit data

Instruction symbol	Processing details	Reference
B+	Adds the two sets of BCD 4-digit data specified. (Two operands)	Page 288 B+(P) [when two operands are set]
B+P		
B+	Adds the two sets of BCD 4-digit data specified. (Three operands)	Page 289 B+(P) [when three operands are set]
B+P		
B-	Performs subtraction between the two sets of BCD 4-digit data specified. (Two operands)	Page 291 B-(P) [when two operands are set]
B-P		
B-	Performs subtraction between the two sets of BCD 4-digit data specified. (Three operands)	Page 292 B-(P) [when three operands are set]
B-P		

■Adding/subtracting BCD 8-digit data

Instruction symbol	Processing details	Reference
DB+	Adds the two sets of BCD 8-digit data specified. (Two operands)	Page 294 DB+(P) [when two operands are set]
DB+P		
DB+	Adds the two sets of BCD 8-digit data specified. (Three operands)	Page 295 DB+(P) [when three operands are set]
DB+P		
DB-	Performs subtraction between the two sets of BCD 8-digit data specified. (Two operands)	Page 297 DB-(P) [when two operands are set]
DB-P		
DB-	Performs subtraction between the two sets of BCD 8-digit data specified. (Three operands)	Page 298 DB-(P) [when three operands are set]
DB-P		

■Multiplying/dividing BCD 4-digit data

Instruction symbol	Processing details	Reference
B*	Multiplies the two sets of BCD 4-digit data specified.	Page 300 B*(P)
B*P		
B/	Performs division between the two sets of BCD 4-digit data specified.	Page 302 B/(P)
B/P		

■Multiplying/dividing BCD 8-digit data

Instruction symbol	Processing details	Reference
DB*	Multiplies the two sets of BCD 8-digit data specified.	Page 304 DB*(P)
DB*P		
DB/	Performs division between the two sets of BCD 8-digit data specified.	Page 306 DB/(P)
DB/P		

■Adding/subtracting 16-bit binary block data

Instruction symbol	Processing details	Reference
BK+	Adds the two 16-bit binary data blocks specified.	Page 308 BK+(P)(_U)
BK+P		
BK+_U		
BK+P_U		
BK-	Performs subtraction between the two 16-bit binary data blocks specified.	Page 310 BK-(P)(_U)
BK-P		
BK-_U		
BK-P_U		

■Adding/subtracting 32-bit binary block data

Instruction symbol	Processing details	Reference
DBK+	Adds the two 32-bit binary data blocks specified.	Page 312 DBK+(P)(_U)
DBK+P		
DBK+_U		
DBK+P_U		
DBK-	Performs subtraction between the two 32-bit binary data blocks specified.	Page 315 DBK-(P)(_U)
DBK-P		
DBK-_U		
DBK-P_U		

■Incrementing/decrementing 16-bit binary data

Instruction symbol	Processing details	Reference
INC	Increments the specified 16-bit binary data by one.	Page 318 INC(P)(_U)
INCP		
INC_U		
INCP_U		
DEC	Decrements the specified 16-bit binary data by one.	Page 319 DEC(P)(_U)
DECP		
DEC_U		
DECP_U		

■Incrementing/decrementing 32-bit binary data

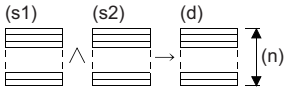
Instruction symbol	Processing details	Reference
DINC	Increments the specified 32-bit binary data by one.	Page 320 DINC(P)(_U)
DINCP		
DINC_U		
DINCP_U		
DDEC	Decrements the specified 32-bit binary data by one.	Page 321 DDEC(P)(_U)
DDECP		
DDEC_U		
DDECP_U		

Logical operation instructions

■Performing an AND operation on 16-bit/32-bit data

Instruction symbol	Processing details	Reference
WAND WANDP	Performs an AND operation on the two sets of 16-bit binary data specified. (Two operands)	Page 323 WAND(P) [when two operands are set]
WAND WANDP	Performs an AND operation on the two sets of 16-bit binary data specified. (Three operands)	Page 325 WAND(P) [when three operands are set]
DAND DANDP	Performs an AND operation on the two sets of 32-bit binary data specified. (Two operands)	Page 327 DAND(P) [when two operands are set]
DAND DANDP	Performs an AND operation on the two sets of 32-bit binary data specified. (Three operands)	Page 329 DAND(P) [when three operands are set]

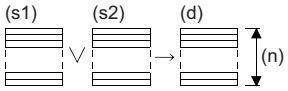
■Performing an AND operation on 16-bit block data

Instruction symbol	Processing details	Reference
BKAND BKANDP	Performs an AND operation on the two 16-bit binary data blocks specified. 	Page 331 BKAND(P)

■Performing an OR operation on 16-bit/32-bit data

Instruction symbol	Processing details	Reference
WOR WORP	Performs an OR operation on the two sets of 16-bit binary data specified. (Two operands)	Page 333 WOR(P) [when two operands are set]
WOR WORP	Performs an OR operation on the two sets of 16-bit binary data specified. (Three operands)	Page 334 WOR(P) [when three operands are set]
DOR DORP	Performs an OR operation on the two sets of 32-bit binary data specified. (Two operands)	Page 336 DOR(P) [when two operands are set]
DOR DORP	Performs an OR operation on the two sets of 32-bit binary data specified. (Three operands)	Page 338 DOR(P) [when three operands are set]

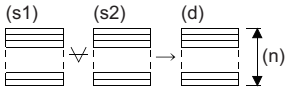
■Performing an OR operation on 16-bit block data

Instruction symbol	Processing details	Reference
BKOR BKORP	Performs an OR operation on the two 16-bit binary data blocks specified. 	Page 340 BKOR(P)

■Performing an XOR operation on 16-bit/32-bit data

Instruction symbol	Processing details	Reference
WXOR WXORP	Performs an XOR operation on the two sets of 16-bit binary data specified. (Two operands)	Page 342 WXOR(P) [when two operands are set]
WXOR WXORP	Performs an XOR operation on the two sets of 16-bit binary data specified. (Three operands)	Page 343 WXOR(P) [when three operands are set]
DXOR DXORP	Performs an XOR operation on the two sets of 32-bit binary data specified. (Two operands)	Page 345 DXOR(P) [when two operands are set]
DXOR DXORP	Performs an XOR operation on the two sets of 32-bit binary data specified. (Three operands)	Page 347 DXOR(P) [when three operands are set]

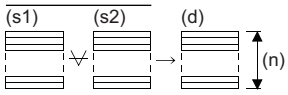
■Performing an XOR operation on 16-bit block data

Instruction symbol	Processing details	Reference
BKXOR BKXORP	Performs an XOR operation on the two 16-bit binary data blocks specified. 	Page 349 BKXOR(P)

■Performing an XNOR operation on 16-bit/32-bit data

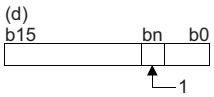
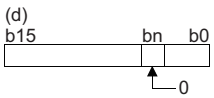
Instruction symbol	Processing details	Reference
WXNR WXNRP	Performs an XNOR operation on the two sets of 16-bit binary data specified. (Two operands)	Page 351 WXNR(P) [when two operands are set]
WXNR WXNRP	Performs an XNOR operation on the two sets of 16-bit binary data specified. (Three operands)	Page 352 WXNR(P) [when three operands are set]
DXNR DXNRP	Performs an XNOR operation on the two sets of 32-bit binary data specified. (Two operands)	Page 354 DXNR(P) [when two operands are set]
DXNR DXNRP	Performs an XNOR operation on the two sets of 32-bit binary data specified. (Three operands)	Page 356 DXNR(P) [when three operands are set]

■Performing an XNOR operation on 16-bit block data

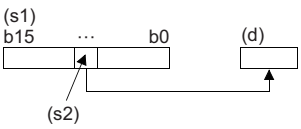
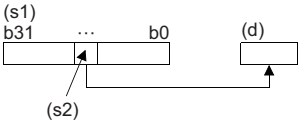
Instruction symbol	Processing details	Reference
BKXNR BKXNRP	Performs an XNOR operation on the two 16-bit binary data blocks specified. 	Page 358 BKXNR(P)

Bit processing instructions

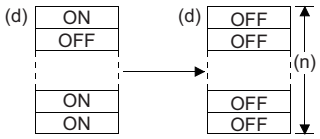
■Setting/resetting a bit in the word device

Instruction symbol	Processing details	Reference
BSET BSETP	Sets the 'n'th bit in the specified word device to 1. 	Page 360 BSET(P)
BRST BRSTP	Resets the 'n'th bit in the specified word device to 0. 	Page 362 BRST(P)

■Performing a bit test

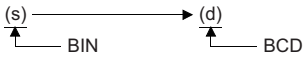
Instruction symbol	Processing details	Reference
TEST TESTP	Extracts the 'n'th bit in the specified word device. 	Page 364 TEST(P)
DTEST DTESTP	Extracts the 'n'th bit in the specified double-word device. 	Page 366 DTEST(P)

■Batch-resetting bit devices

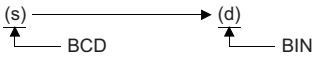
Instruction symbol	Processing details	Reference
BKRST BKRSTP	Resets the (n) points of bit devices starting from the bit device specified. 	Page 368 BKRST(P)

Data conversion instructions

■Converting binary data to BCD 4-digit/8-digit data

Instruction symbol	Processing details	Reference
BCD BCDP	Converts the specified 16-bit binary data (0 to 9999) to BCD 4-digit data. 	Page 390 BCD(P)
DBCD DBCDP	Converts the specified 32-bit binary data (0 to 99999999) to BCD 8-digit data. 	Page 392 DBCD(P)

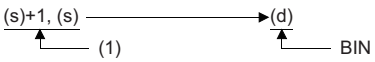
■Converting BCD 4-digit/8-digit data to 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
BIN BINP	Converts the specified BCD 4-digit data (0 to 9999) to 16-bit binary data. 	Page 394 BIN(P)
DBIN DBINP	Converts the specified BCD 8-digit data (0 to 99999999) to 32-bit binary data. 	Page 396 DBIN(P)

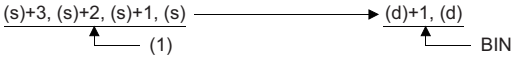
■Converting single-precision real number to 16-bit/32-bit signed binary data

Instruction symbol	Processing details	Reference
FLT2INT FLT2INTP	Converts the specified single-precision real number (-32768 to 32767) to 16-bit signed binary data.  (1) Real number	Page 398 FLT2INT(P)
FLT2DINT FLT2DINTP	Converts the specified single-precision real number (-2147483648 to 2147483647) to 32-bit signed binary data.  (1) Real number	Page 402 FLT2DINT(P)

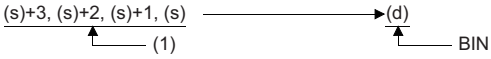
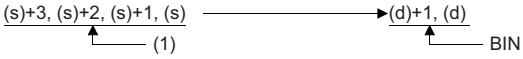
■Converting single-precision real number to 16-bit/32-bit unsigned binary data

Instruction symbol	Processing details	Reference
FLT2UINT FLT2UINTP	Converts the specified single-precision real number (0 to 65535) to 16-bit unsigned binary data.  (1) Real number	Page 400 FLT2UINT(P)
FLT2UDINT FLT2UDINTP	Converts the specified single-precision real number (0 to 4294967295) to 32-bit unsigned binary data.  (1) Real number	Page 404 FLT2UDINT(P)

■Converting double-precision real number to 16-bit/32-bit signed binary data

Instruction symbol	Processing details	Reference
DBL2INT DBL2INTP	Converts the specified double-precision real number (-32768 to 32767) to 16-bit signed binary data.  (1) Real number	Page 406 DBL2INT(P)
DBL2DINT DBL2DINTP	Converts the specified double-precision real number (-2147483648 to 2147483647) to 32-bit signed binary data.  (1) Real number	Page 410 DBL2DINT(P)

■Converting double-precision real number to 16-bit/32-bit unsigned binary data

Instruction symbol	Processing details	Reference
DBL2UINT DBL2UINTP	Converts the specified double-precision real number (0 to 65535) to 16-bit unsigned binary data.  (1) Real number	Page 408 DBL2UINT(P)
DBL2UDINT DBL2UDINTP	Converts the specified double-precision real number (0 to 4294967295) to 32-bit unsigned binary data.  (1) Real number	Page 412 DBL2UDINT(P)

■Converting 16-bit signed binary data to 16-bit/32-bit unsigned binary data

Instruction symbol	Processing details	Reference
INT2UINT INT2UINTP	Converts the 16-bit signed binary data in the device specified by (s) to 16-bit unsigned binary data, and stores the converted data in the device specified by (d).	Page 414 INT2UINT(P)
INT2UDINT INT2UDINTP	Converts the 16-bit signed binary data in the device specified by (s) to 32-bit unsigned binary data, and stores the converted data in the device specified by (d).	Page 418 INT2UDINT(P)

■Converting 16-bit signed binary data to 32-bit signed binary data

Instruction symbol	Processing details	Reference
INT2DINT INT2DINTP	Converts the 16-bit signed binary data in the device specified by (s) to 32-bit signed binary data, and stores the converted data in the device specified by (d).	Page 416 INT2DINT(P)

■Converting 16-bit unsigned binary data to 16-bit/32-bit signed binary data

Instruction symbol	Processing details	Reference
UINT2INT	Converts the 16-bit unsigned binary data in the device specified by (s) to 16-bit singed binary data, and stores the converted data in the device specified by (d).	Page 420 UINT2INT(P)
UINT2INTP		
UINT2DINT	Converts the 16-bit unsigned binary data in the device specified by (s) to 32-bit singed binary data, and stores the converted data in the device specified by (d).	Page 422 UINT2DINT(P)
UINT2DINTP		

■Converting 16-bit unsigned binary data to 32-bit unsigned binary data

Instruction symbol	Processing details	Reference
UINT2UDINT	Converts the 16-bit unsigned binary data in the device specified by (s) to 32-bit unsigned binary data, and stores the converted data in the device specified by (d).	Page 424 UINT2UDINT(P)
UINT2UDINTP		

■Converting 32-bit signed binary data to 16-bit signed binary data

Instruction symbol	Processing details	Reference
DINT2INT	Converts the 32-bit signed binary data in the device specified by (s) to 16-bit signed binary data, and stores the converted data in the device specified by (d).	Page 426 DINT2INT(P)
DINT2INTP		

■Converting 32-bit signed binary data to 16-bit/32-bit unsigned binary data

Instruction symbol	Processing details	Reference
DINT2UINT	Converts the 32-bit signed binary data in the device specified by (s) to 16-bit unsigned binary data, and stores the converted data in the device specified by (d).	Page 428 DINT2UINT(P)
DINT2UINTP		
DINT2UDINT	Converts the 32-bit signed binary data in the device specified by (s) to 32-bit unsigned binary data, and stores the converted data in the device specified by (d).	Page 430 DINT2UDINT(P)
DINT2UDINTP		

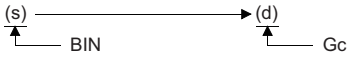
■Converting 32-bit unsigned binary data to 16-bit/32-bit signed binary data

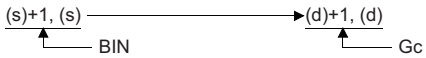
Instruction symbol	Processing details	Reference
UDINT2INT	Converts the 32-bit unsigned binary data in the device specified by (s) to 16-bit singed binary data, and stores the converted data in the device specified by (d).	Page 432 UDINT2INT(P)
UDINT2INTP		
UDINT2DINT	Converts the 32-bit unsigned binary data in the device specified by (s) to 32-bit singed binary data, and stores the converted data in the device specified by (d).	Page 436 UDINT2DINT(P)
UDINT2DINTP		

■Converting 32-bit unsigned binary data to 16-bit unsigned binary data

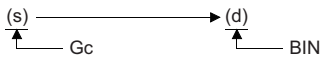
Instruction symbol	Processing details	Reference
UDINT2UINT	Converts the 32-bit unsigned binary data in the device specified by (s) to 16-bit unsigned binary data, and stores the converted data in the device specified by (d).	Page 434 UDINT2UINT(P)
UDINT2UINTP		

■Converting 16-bit/32-bit binary data to 16-bit/32-bit binary Gray code data

Instruction symbol	Processing details	Reference
GRY	Converts the specified 16-bit binary data (-32768 to 32767) to 16-bit binary Gray code data.  Gc: Gray code	Page 438 GRY(P)(_U)
GRYP		
GRY_U	Converts the specified 16-bit binary data (0 to 65535) to 16-bit binary Gray code data.  Gc: Gray code	
GRYP_U		

Instruction symbol	Processing details	Reference
DGRY DGRYP	Converts the specified 32-bit binary data (-2147483648 to 2147483647) to 32-bit binary Gray code data.  Gc: Gray code	Page 440 DGRY(P)(_U)
DGRY_U DGRYP_U	Converts the specified 32-bit binary data (0 to 4294967295) to 32-bit binary Gray code data.  Gc: Gray code	

■Converting 16-bit/32-bit binary Gray code data to 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
GBIN GBINP	Converts the specified 16-bit binary Gray code data (-32768 to 32767) to 16-bit binary data.  Gc: Gray code	Page 442 GBIN(P)(_U)
GBIN_U GBINP_U	Converts the specified 16-bit binary Gray code data (0 to 65535) to 16-bit binary data.  Gc: Gray code	
DGBIN DGBINP	Converts the specified 32-bit binary Gray code data (-2147483648 to 2147483647) to 32-bit binary data.  Gc: Gray code	Page 444 DGBIN(P)(_U)
DGBIN_U DGBINP_U	Converts the specified 32-bit binary Gray code data (0 to 4294967295) to 32-bit binary data.  Gc: Gray code	

■Converting 16-bit binary data block to BCD 4-digit data block

Instruction symbol	Processing details	Reference
BKBCD BKBCDP	Batch-converts the (n) points of binary data in the device starting from the one specified by (s) to BCD data, and stores the converted data in the device specified by (d) and later.	Page 446 BKBCD(P)

■Converting BCD 4-digit block data to 16-bit binary block data

Instruction symbol	Processing details	Reference
BKBIN BKBINP	Batch-converts the (n) points of BCD data in the device starting from the one specified by (s) to binary data, and stores the converted data in the device specified by (d) and later.	Page 448 BKBIN(P)

■Converting decimal ASCII data to 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
DABIN DABINP DABIN_U DABINP_U	Converts the decimal ASCII value in the device specified by (s) to 1-word binary data, and stores the converted data in the word device number specified by (d).	Page 450 DABIN(P)(_U)
DDABIN DDABINP DDABIN_U DDABINP_U	Converts the decimal ASCII value in the device specified by (s) to 2-word binary data, and stores the converted data in the word device number specified by (d).	Page 453 DDABIN(P)(_U)

■Converting hexadecimal ASCII data to 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
HABIN HABINP	Converts the hexadecimal ASCII value in the device specified by (s) to 1-word binary data, and stores the converted data in the word device number specified by (d).	Page 457 HABIN(P)
DHABIN DHABINP	Converts the hexadecimal ASCII value in the device specified by (s) to 2-word binary data, and stores the converted data in the word device number specified by (d).	Page 459 DHABIN(P)

■Converting decimal ASCII data to BCD 4-digit/8-digit data

Instruction symbol	Processing details	Reference
DABCD	Converts the decimal ASCII value in the device specified by (s) to 1-word BCD data, and stores the converted data in the word device number specified by (d).	Page 462 DABCD(P)
DABCDP		
DDABCD	Converts the decimal ASCII value in the device specified by (s) to 2-word BCD data, and stores the converted data in the word device number specified by (d).	Page 465 DDABCD(P)
DDABCDP		

■Converting decimal string data to 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
VAL	Converts a string including the decimal point in the device specified by (s) to 1-word binary data and the number of decimal positions, and stores the converted data in the device areas specified by (d1) and (d2).	Page 468 VAL(P)(_U)
VALP		
VAL_U		
VALP_U		
DVAL	Converts a string including the decimal point in the device specified by (s) to 2-word binary data and the number of decimal positions, and stores the converted data in the device areas specified by (d1) and (d2).	Page 471 DVAL(P)(_U)
DVALP		
DVAL_U		
DVALP_U		

■Converting hexadecimal ASCII to hexadecimal binary data

Instruction symbol	Processing details	Reference
ASC2INT	Converts the hexadecimal ASCII data in the word device specified by (s) and later to binary data by the number of characters specified by (n), and stores the converted data in the device number specified by (d) and later.	Page 474 ASC2INT(P)
ASC2INTP		

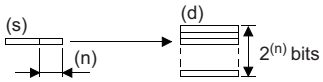
■Converting single-precision real number to BCD format data

Instruction symbol	Processing details	Reference
EMOD	Converts the 32-bit floating-point data in the device specified by (s1) to BCD of the number of decimal positions specified by (s2), and stores the converted data in the device specified by (d).	Page 476 EMOD(P)
EMODP		

■Two's complement of 16-bit/32-bit binary data (sign inversion)

Instruction symbol	Processing details	Reference
NEG	Inverts the sign of 16-bit binary device.	Page 478 NEG(P)
NEGP		
DNEG	Inverts the sign of 32-bit binary device.	Page 480 DNEG(P)
DNEGP		

■Decoding 8-bit data to 256-bit data

Instruction symbol	Processing details	Reference
DECO	Decodes the lower (n) bits of the specified device.	Page 481 DECO(P)
DECOP		

■Encoding 256-bit data to 8-bit data

Instruction symbol	Processing details	Reference
ENCO	Encodes the bit data of 'n'th power of 2.	Page 483 ENCO(P)
ENCOP		

■Decoding data to seven-segment display data

Instruction symbol	Processing details	Reference
SEG SEGP	Decodes the data consisting of 0 to F specified by the lower 4 bits of the device to seven-segment display data. 	Page 485 SEG(P)

■Separating data in units of 4 bits

Instruction symbol	Processing details	Reference
DIS DISP	Separates the 16-bit in the device specified by (s) in units of 4 bits, and stores the separated data in the (n) points of 4 low-order bits in the device starting from the one specified by (d). (n<4)	Page 488 DIS(P)

■Combining data in units of 4 bits

Instruction symbol	Processing details	Reference
UNI UNIP	Adds the (n) points of 4 low-order bit data in the device starting from the one specified by (s), and stores the connected data in the device specified by (d). (n<4)	Page 490 UNI(P)

■Separating/combining data in units of bits

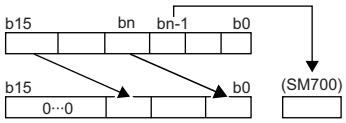
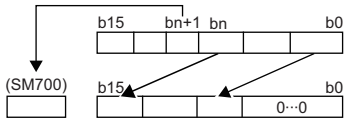
Instruction symbol	Processing details	Reference
NDIS NDISP	Separates the data in the device, specified by (s1) and later, to the bits in the device specified by (s2) and later, and stores the separated data in order in the device starting from the one specified by (d).	Page 492 NDIS(P)
NUNI NUNIP	Connects the data in the device specified by (s1) and later, in units of bits in the device specified by (s2) and later, and stores the connected data in order in the device starting from the one specified by (d).	Page 494 NUNI(P)

■Separating/combining data in units of bytes

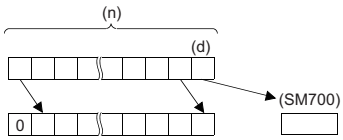
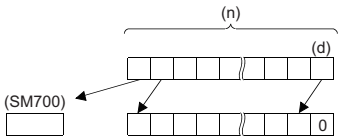
Instruction symbol	Processing details	Reference
WTOB WTOBP	Converts the (n) points of 16 bit data in the device specified by (s) in units of 8 bits, and stores the converted data in order in the device starting from the one specified by (d).	Page 496 WTOB(P)
BTOW BTOWP	Connects 8 low-order bits of the (n) points of 16 bit data in the device specified by (s) to 16 bits, and stores the connected data in order in the device starting from the one specified by (d).	Page 498 BTOW(P)

Shift instructions

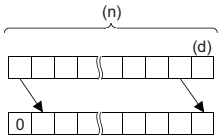
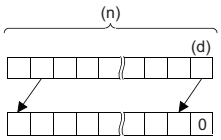
■Shifting 16-bit binary data to the right/left by n bit(s)

Instruction symbol	Processing details	Reference
SFR SFRP	Shifts the 16-bit binary data in the specified device to the right. In the empty area after the shift, 0 is stored. 	Page 370 SFR(P)
SFL SFLP	Shifts the 16-bit binary data in the specified device to the left. In the empty area after the shift, 0 is stored. 	Page 372 SFL(P)

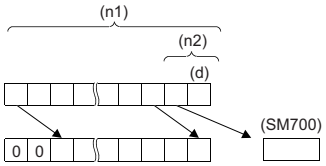
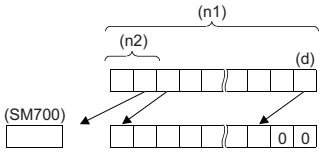
■Shifting n-bit data to the right/left by one bit

Instruction symbol	Processing details	Reference
BSFR BSFRP	Shifts the n points of data starting from the specified device to the right by one bit. In the empty area after the shift, 0 is stored. 	Page 374 BSFR(P)
BSFL BSFLP	Shifts the n points of data starting from the specified device to the left by one bit. In the empty area after the shift, 0 is stored. 	Page 376 BSFL(P)

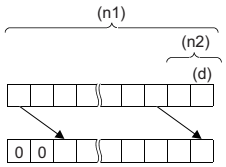
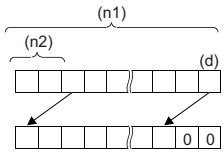
■Shifting n-word data to the right/left by one word

Instruction symbol	Processing details	Reference
DSFR DSFRP	Shifts the n points of data starting from the specified device to the right by one word. In the empty area after the shift, 0 is stored. 	Page 378 DSFR(P)
DSFL DSFLP	Shifts the n points of data starting from the specified device to the left by one word. In the empty area after the shift, 0 is stored. 	Page 380 DSFL(P)

■Shifting n-bit data to the right/left by n bit(s)

Instruction symbol	Processing details	Reference
SFTBR SFTBRP	Shifts the n-bit data starting from the specified device to the right by n bit(s). In the empty area after the shift, 0 is stored. 	Page 382 SFTBR(P)
SFTBL SFTBLP	Shifts the n-bit data starting from the specified device to the left by n bit(s). In the empty area after the shift, 0 is stored. 	Page 384 SFTBL(P)

■Shifting n-word data to the right/left by n word(s)

Instruction symbol	Processing details	Reference
SFTWR SFTWRP	Shifts the n-word data starting from the specified device to the right by n word(s). In the empty area after the shift, 0 is stored. 	Page 386 SFTWR(P)
SFTWL SFTWLP	Shifts the n-word data starting from the specified device to the left by n word(s). In the empty area after the shift, 0 is stored. 	Page 388 SFTWL(P)

Data transfer instructions

■Transferring 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
MOV MOVP	Transfers the 16-bit binary data in the device specified. (s) ⟶ (d)	Page 500 MOV(P)
DMOV DMOVP	Transfers the 32-bit binary data in the device specified. (s)+1, (s) ⟶ (d)+1, (d)	Page 502 DMOV(P)

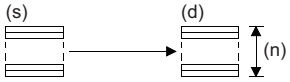
■Inverting and transferring 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
CML CMLP	Inverts the specified 16-bit binary data bit by bit, and transfers the inverted data. $\overline{(s)}$ ⟶ (d)	Page 504 CML(P)
DCML DCMLP	Inverts the specified 32-bit binary data bit by bit, and transfers the inverted data. $\overline{(s)+1, (s)}$ ⟶ (d)+1, (d)	Page 506 DCML(P)

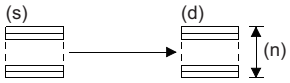
■Inverting and transferring 1-bit data

Instruction symbol	Processing details	Reference
CMLB CMLBP	Inverts the bit data in the device specified by (s), and stores the inverted data in the device specified by (d).	Page 508 CMLB(P)

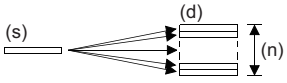
■Transferring 16-bit binary data block (16 bits)

Instruction symbol	Processing details	Reference
BMOV BMOVP	Batch-transfers the (n) points of 16-bit binary data starting from the device specified.  (n) = 1 to 65535	Page 510 BMOV(P)

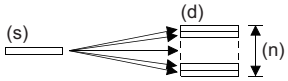
■Transferring 16-bit binary data block (32 bits)

Instruction symbol	Processing details	Reference
BMOVL BMOVLP	Batch-transfers the (n) points of 16-bit binary data starting from the device specified.  (n) = 1 to 4294967295	Page 512 BMOVL(P)

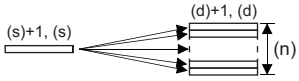
■Transferring the same 16-bit binary data block (16 bits)

Instruction symbol	Processing details	Reference
FMOV FMOVP	Transfers 16-bit binary data to the (n) points starting from the device specified.  (n) = 1 to 65535	Page 514 FMOV(P)

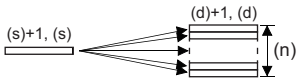
■Transferring the same 16-bit binary data block (32 bits)

Instruction symbol	Processing details	Reference
FMOVL FMOVLP	Transfers 16-bit binary data to the (n) points starting from the device specified.  (n) = 1 to 4294967295	Page 516 FMOVL(P)

■Transferring the same 32-bit binary data block (16 bits)

Instruction symbol	Processing details	Reference
DFMOV DFMOVP	Transfers 32-bit binary data to the (n) points starting from the device specified.  (n) = 1 to 65535	Page 518 DFMOV(P)

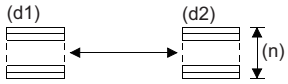
■Transferring the same 32-bit binary data block (32 bits)

Instruction symbol	Processing details	Reference
DFMOVL DFMOVLP	Transfers 32-bit binary data to the (n) points starting from the device specified.  (n) = 1 to 4294967295	Page 520 DFMOVL(P)

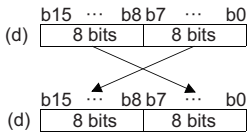
■Exchanging 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
XCH XCHP	Exchanges the 16-bit binary data specified. (d1) ◄——► (d2)	Page 522 XCH(P)
DXCH DXCHP	Exchanges the 32-bit binary data specified. (d1)+1, (d1) ◄——► (d2)+1, (d2)	Page 523 DXCH(P)

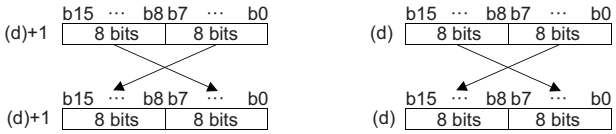
■Exchanging 16-bit binary block data

Instruction symbol	Processing details	Reference
BXCH BXCHP	Exchanges the (n) points of 16-bit binary data starting from the devices specified. 	Page 524 BXCH(P)

■Exchanging the upper and lower bytes of 16-bit binary data

Instruction symbol	Processing details	Reference
SWAP SWAPP	Exchanges upper and lower 8-bit data in the specified device. 	Page 526 SWAP(P)

■Exchanging the upper and lower bytes of 32-bit binary data

Instruction symbol	Processing details	Reference
DSWAP DSWAPP	Exchanges upper and lower 8-bit data in the specified device. 	Page 527 DSWAP(P)

■Transferring 1-bit data

Instruction symbol	Processing details	Reference
MOVB MOVBP	Stores the bit data in the device specified by (s) in the device specified by (d).	Page 528 MOVB(P)

■Transferring n-bit data

Instruction symbol	Processing details	Reference
BLKMOVB BLKMOVBP	Batch-transfers the (n) points of bit data in the device starting from the one specified by (s) to the (n) points of bit data in the device starting from the one specified by (d).	Page 529 BLKMOVB(P)

6.3 Application Instructions

Program control

Program branch instructions

■Pointer branch

Instruction symbol	Processing details	Reference
CJ	Executes the program specified by the pointer number within the same program file.	Page 532 CJ, SCJ, JMP
SCJ	Executes the program specified by the pointer number within the same program file starting with the next scan.	
JMP	Unconditionally executes the program specified by the pointer number within the same program file.	

■Jumping to END

Instruction symbol	Processing details	Reference
GOEND	Invokes a jump to the FEND or END instruction within the same program file.	Page 535 GOEND

Program execution control instructions

■Disabling/enabling interrupt programs

Instruction symbol	Processing details	Reference
DI	Disables the execution of interrupt programs.	Page 536 DI, EI
EI	Clears the interrupt disabled state.	

■Disabling the interrupt program with specified priority or lower

Instruction symbol	Processing details	Reference
DI	This instruction disables execution of interrupt programs with the specified priority or lower.	Page 539 DI

■Interrupt program mask

Instruction symbol	Processing details	Reference
IMASK	Enables or disables the execution of the interrupt program with the specified interrupt pointer number.	Page 544 IMASK

■Disabling/enabling the specified interrupt pointer

Instruction symbol	Processing details	Reference
SIMASK	Enables or disables the execution of the interrupt program with the specified interrupt pointer number.	Page 546 SIMASK

■Returning from the interrupt program

Instruction symbol	Processing details	Reference
IRET	Indicates the end of the processing of an interrupt program.	Page 548 IRET

■Resetting the watchdog timer

Instruction symbol	Processing details	Reference
WDT	Resets the watchdog timer.	Page 549 WDT(P)
WDTP		

Structure creation instructions

■Performing the FOR to NEXT instruction loop

Instruction symbol	Processing details	Reference
FOR	Executes the processing between FOR to NEXT (n) times.	Page 550 FOR, NEXT
NEXT		

■Forcibly terminating the FOR to NEXT instruction loop

Instruction symbol	Processing details	Reference
BREAK	These instructions forcibly terminate the loop processing between the FOR and NEXT instructions, and pass the control to the processing of FOR to NEXT instruction or later.	Page 553 BREAK(P)
BREAKP		

■Calling a subroutine program

Instruction symbol	Processing details	Reference
CALL	Executes a subroutine program specified by (P) when the input condition is met. (For (s1) to (s5), specify the arguments to be passed to the subroutine program.)	Page 555 CALL(P)
CALLP		

■Returning from the subroutine program called

Instruction symbol	Processing details	Reference
RET	Indicates the end of a subroutine program.	Page 561 RET

■Calling a subroutine program and turning the output off

Instruction symbol	Processing details	Reference
FCALL	Performs non-execution processing of the subroutine program specified by (P) when the input conditions are not met. (For (s1) to (s5), specify the arguments to be passed to the subroutine program.)	Page 562 FCALL(P)
FCALLP		

■Calling a subroutine program in the specified program file

Instruction symbol	Processing details	Reference
ECALL	Executes the subroutine program specified by (P) of the specified program when the input conditions are met. (For (s1) to (s5), specify the arguments to be passed to the subroutine program.)	Page 568 ECALL(P)
ECALLP		

■Calling a subroutine program in the specified program file and turning the output off

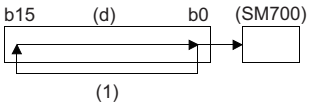
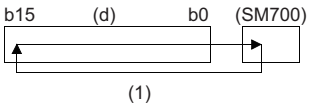
Instruction symbol	Processing details	Reference
EFCALL	Performs non-execution processing of the subroutine program specified by (P) of the specified program when the input conditions are not met. (For (s1) to (s5), specify the arguments to be passed to the subroutine program.)	Page 575 EFCALL(P)
EFCALLP		

■Calling a subroutine program

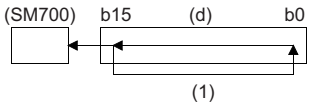
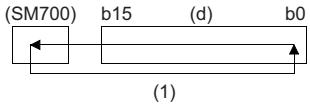
Instruction symbol	Processing details	Reference
XCALL	Executes a subroutine program specified by (P) when the input condition is met. Perform non-execution processing of the subroutine program specified by (P) when the input conditions are not met. (For (s1) to (s5), specify the arguments to be passed to the subroutine program.)	Page 580 XCALL

Rotation instructions

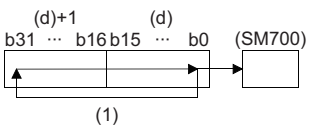
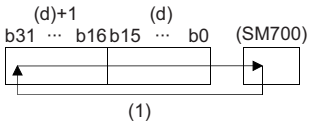
■Rotating 16-bit binary data to the right

Instruction symbol	Processing details	Reference
ROR RORP	Rotates the 16-bit binary data to the right by (n) bit(s) (not including the carry flag).  (1) Right rotation by (n) bits	Page 585 ROR(P), RCR(P)
RCR RCRP	Rotates the 16-bit binary data to the right by (n) bit(s) (including the carry flag).  (1) Right rotation by (n) bits	

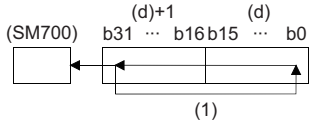
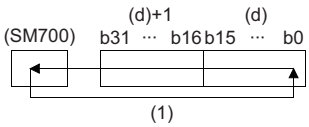
■Rotating 16-bit binary data to the left

Instruction symbol	Processing details	Reference
ROL ROLP	Rotates the 16-bit binary data to the left by (n) bit(s) (not including the carry flag).  (1) Left rotation by (n) bits	Page 588 ROL(P), RCL(P)
RCL RCLP	Rotates the 16-bit binary data to the left by (n) bit(s) (including the carry flag).  (1) Left rotation by (n) bits	

■Rotating 32-bit binary data to the right

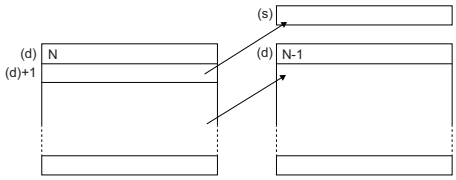
Instruction symbol	Processing details	Reference
DROR DRORP	Rotates the 32-bit binary data to the right by (n) bit(s) (not including the carry flag).  (1) Right rotation by (n) bits	Page 591 DROR(P), DRCR(P)
DRCR DRCRP	Rotates the 32-bit binary data to the right by (n) bit(s) (including the carry flag).  (1) Right rotation by (n) bits	

■Rotating 32-bit binary data to the left

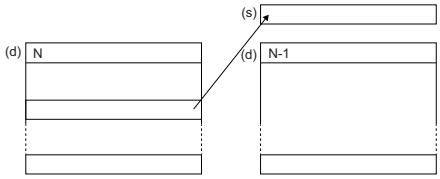
Instruction symbol	Processing details	Reference
DROL DROLP	Rotates the 32-bit binary data to the left by (n) bit(s) (not including the carry flag).  (1) Left rotation by (n) bits	Page 593 DROL(P), DRCL(P)
DRCL DRCLP	Rotates the 32-bit binary data to the left by (n) bit(s) (including the carry flag).  (1) Left rotation by (n) bits	

Data table operation instructions

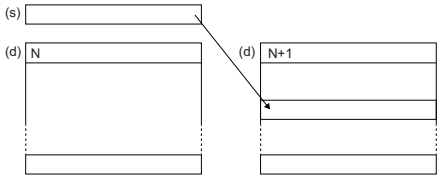
■Reading the oldest data from the data table

Instruction symbol	Processing details	Reference
FIFR FIFRP	Stores the data first stored in the table in the specified device.  N: Number of data	Page 595 FIFR(P)

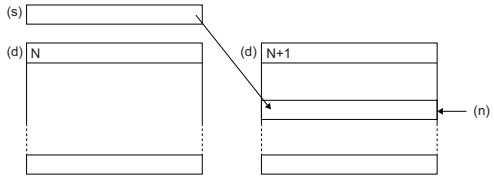
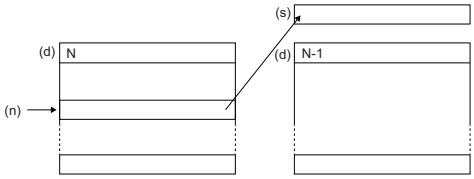
■Reading the newest data from the data table

Instruction symbol	Processing details	Reference
FPOP FPOPP	Stores the data last stored in the table in the specified device.  N: Number of data	Page 597 FPOP(P)

■Writing data to the data table

Instruction symbol	Processing details	Reference
FIFW FIFWP	Stores 16-bit binary data to the specified data table.  N: Number of data	Page 599 FIFW(P)

■Inserting/deleting data to/from the data table

Instruction symbol	Processing details	Reference
FINS FINSP	Inserts 16-bit binary data to the (n)th position in the specified data table.  N: Number of data	Page 601 FINS(P)
FDEL FDELP	Deletes the data at the (n)th position in the data table.  N: Number of data	Page 603 FDEL(P)

Reading/writing data instructions

■Reading data from the data memory

Instruction symbol	Processing details	Reference
S.DEVLD SP.DEVLD	Reads data from the device data storage file in data memory.	Page 606 S(P).DEVLD

■Writing data to the data memory

Instruction symbol	Processing details	Reference
SP.DEVST	Writes the specified number of points of data to the device data storage file in data memory.	Page 608 SP.DEVST

File operation instructions

■Reading data from the specified file

Instruction symbol	Processing details	Reference
SP.FREAD	Reads data from the specified file.	Page 610 SP.FREAD

■Writing data to the specified file

Instruction symbol	Processing details	Reference
SP.FWRITE	Writes data to the specified file.	Page 624 SP.FWRITE

■Deleting the specified file

Instruction symbol	Processing details	Reference
SP.FDELETE	Deletes the specified file or folder.	Page 638 SP.FDELETE

■Copying the specified file

Instruction symbol	Processing details	Reference
SP.FCOPY	Copies the specified file or folder.	Page 641 SP.FCOPY

■Moving the specified file

Instruction symbol	Processing details	Reference
SP.FMOVE	Moves the specified file or folder.	Page 646 SP.FMOVE

■Renaming the specified file

Instruction symbol	Processing details	Reference
SP.FRENAME	Renames the specified file or folder.	Page 651 SP.FRENAME

■Acquiring the status of the specified file

Instruction symbol	Processing details	Reference
SP.FSTATUS	Acquires the status of the specified file or folder.	Page 655 SP.FSTATUS

Data control instructions

■Upper and lower limit control of 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
LIMIT	Controls the output value depending on whether the specified 16-bit binary bit value is within the upper and lower limits.	Page 661 LIMIT(P)(_U)
LIMITP		
LIMIT_U		
LIMITP_U		
DLIMIT	Controls the output value depending on whether the specified 32-bit binary bit value is within the upper and lower limits.	Page 663 DLIMIT(P)(_U)
DLIMITP		
DLIMIT_U		
DLIMITP_U		

■Dead band control of 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
BAND	Controls the output value depending on whether the specified 16-bit binary bit value is within the upper and lower limits of the dead band.	Page 665 BAND(P)(_U)
BANDP		
BAND_U		
BANDP_U		
DBAND	Controls the output value depending on whether the specified 32-bit binary bit value is within the upper and lower limits of the dead band.	Page 667 DBAND(P)(_U)
DBANDP		
DBAND_U		
DBANDP_U		

■Zone control of 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
ZONE ZONEP ZONE_U ZONEP_U	Adds a bias value to the specified input value (16-bit binary).	Page 669 ZONE(P)(_U)
DZONE DZONEP DZONE_U DZONEP_U	Adds a bias value to the specified input value (32-bit binary).	Page 671 DZONE(P)(_U)

■Scaling 16-bit binary data/32-bit binary data/double-precision real number (point coordinates)

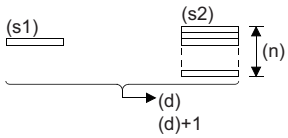
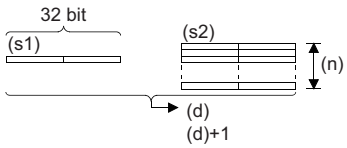
Instruction symbol	Processing details	Reference
SCLN SCLNP SCLN_U SCLNP_U	Scales the scaling conversion data (16-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.	Page 673 SCLN(P)(_U)
SCL SCLP SCL_U SCLP_U	Scales the scaling conversion data (16-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.	Page 677 SCL(P)(_U)
DSCLN DSCLNP DSCLN_U DSCLNP_U	Scales the scaling conversion data (32-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.	Page 696 DSCL2N(P)(_U)
DSCL DSCLP DSCL_U DSCLP_U	Scales the scaling conversion data (32-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.	Page 684 DSCL(P)(_U)
EDSCLN EDSCLNP	Scales the scaling conversion data (double-precision real number) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.	Page 687 EDSCLN(P)

■Scaling 16-bit/32-bit binary data (XY coordinates)

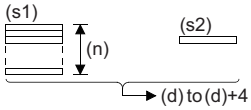
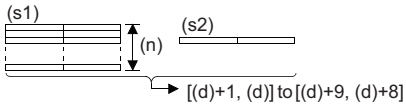
Instruction symbol	Processing details	Reference
SCL2N SCL2NP SCL2N_U SCL2NP_U	Scales the scaling conversion data (16-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.	Page 691 SCL2N(P)(_U)
SCL2 SCL2P SCL2_U SCL2P_U	Scales the scaling conversion data (16-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.	Page 694 SCL2(P)(_U)
DSCL2N DSCL2NP DSCL2N_U DSCL2NP_U	Scales the scaling conversion data (32-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.	Page 699 DSCL2(P)(_U)
DSCL2 DSCL2P DSCL2_U DSCL2P_U	Scales the scaling conversion data (32-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.	Page 699 DSCL2(P)(_U)

Data processing instructions

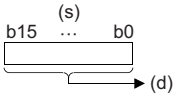
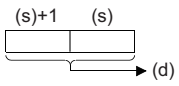
■Searching 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
SERDATA SERDATAP	Searches the (n) points in the 16-bit binary data specified by (s2) for the 16-bit binary data specified by (s1).  (d): Location (d)+1: Number of matches	Page 701 SERDATA(P)
DSERDATA DSERDATAP	Searches the (n) points in the 32-bit binary data specified by (s2) for the 32-bit binary data specified by (s1).  (d): Location (d)+1: Number of matches	Page 703 DSERDATA(P)

■Searching 16-bit/32-bit binary data (minimum, match, maximum)

Instruction symbol	Processing details	Reference
SERMM SERMMP	Searches the (n) points in the 16-bit binary data specified by (s1) for the same data as the 16-bit binary data specified by (s2), the minimum value, and the maximum value. 	Page 705 SERMM(P)
DSERMM DSERMMP	Searches the (n) points in the 32-bit binary data specified by (s1) for the same data as the 32-bit binary data specified by (s2), the minimum value, and the maximum value. 	Page 707 DSERMM(P)

■Checking 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
SUM SUMP	Stores the total number of "1" bits in the 16-bit binary data stored in the specified device.  (d): Number of 1s	Page 709 SUM(P)
DSUM DSUMP	Stores the total number of "1" bits in the 32-bit binary data stored in the specified device.  (d): Number of 1s	Page 710 DSUM(P)

■Searching for the maximum value of 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
MAX	Searches the (n) points of data in the device specified by (s) in units of 16 bits, and stores the maximum value in the device specified by (d).	Page 711 MAX(P)(_U)
MAXP		
MAX_U		
MAXP_U		
DMAX	Searches the (n) points of data in the device specified by (s) in units of 32 bits, and stores the maximum value in the device specified by (d).	Page 713 DMAX(P)(_U)
DMAXP		
DMAX_U		
DMAXP_U		

■Searching for the minimum value of 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
MIN	Searches the (n) points of data in the device specified by (s) in units of 16 bits, and stores the minimum value in the device specified by (d).	Page 715 MIN(P)(_U)
MINP		
MIN_U		
MINP_U		
DMIN	Searches the (n) points of data in the device specified by (s) in units of 32 bits, and stores the minimum value in the device specified by (d).	Page 717 DMIN(P)(_U)
DMINP		
DMIN_U		
DMINP_U		

■Sorting 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
SORTD	Sorts the (n) points of data in the device specified by (s1) in units of 16 bits. ("(n)×((n)-1)÷2" scanning required)	Page 719 SORTD(_U)
SORTD_U		
DSORTD	Sorts the (n) points of data in the device specified by (s1) in units of 32 bits. ("(n)×((n)-1)÷2" scanning required)	Page 721 DSORTD(_U)
DSORTD_U		

■Adding 16-bit binary data

Instruction symbol	Processing details	Reference
WSUM	Adds the (n) points of 16-bit binary data in the device starting from the one specified by (s), and stores the result in the device specified by (d).	Page 723 WSUM(P)(_U)
WSUM_U		
WSUMP		
WSUMP_U		

■Adding 32-bit binary data

Instruction symbol	Processing details	Reference
DWSUM	Adds the (n) points of 32-bit binary data in the device starting from the one specified by (s), and stores the result in the device specified by (d).	Page 725 DWSUM(P)(_U)
DWSUM_U		
DWSUMP		
DWSUMP_U		

■Calculating the mean value of 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
MEAN	Calculates the average value of the (n) points of 16-bit data in the device starting from the one specified by (s), and stores the average value in the device specified by (d).	Page 727 MEAN(P)(_U)
MEANP		
MEAN_U		
MEANP_U		
DMEAN	Calculates the average value of the (n) points of 32-bit data in the device starting from the one specified by (s), and stores the average value in the device specified by (d).	Page 729 DMEAN(P)(_U)
DMEANP		
DMEAN_U		
DMEANP_U		

■Calculating the square root of 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
SQRT	Performs a square root operation of the specified 16-bit binary data. $\sqrt{(s)} \rightarrow (d)$	Page 731 SQRT(P)
SQRTP		
DSQRT	Performs a square root operation of the specified 32-bit binary data. $\sqrt{(s)+1, (s)} \rightarrow (d)$	Page 732 DSQRT(P)
DSQRTP		

■CRC operation

Instruction symbol	Processing details	Reference
CRC	Generates the CRC value for (n) 8-bit data (unit: byte) starting from the device specified by (s), and store the CRC value to the device specified by (d).	Page 733 CRC(P)
CRCP		

Debugging and failure diagnostic

Debugging and failure diagnostic instructions

■Resetting the error display and the annunciator display

Instruction symbol	Processing details	Reference
LEDR	This instruction resets the self-diagnostic error (continuation error) display and the annunciator display of the controller.	Page 735 LEDR

String processing

String processing instructions

■Comparing string data

Instruction symbol	Processing details	Reference
LD$=, AND$=, OR$=	Compares the string specified by (s1) with the string specified by (s2) (character by character).	Page 737 LD$☐, AND$☐, OR$☐
LD$<>, AND$<>, OR$<>		
LD$>, AND$>, OR$>		
LD$<=, AND$<=, OR$<=		
LD$<, AND$<, OR$<		
LD$>=, AND$>=, OR$>=		

■Concatenating string data

Instruction symbol	Processing details	Reference
$+	Connects the strings in the device specified by (s) to those in the device specified by (d), and stores the connected data in the device specified by (d) and later.	Page 740 $+(P) [when two operands are set]
$+P		
$+	Connects the strings in the device specified by (s2) to those in the device specified by (s1), and stores the connected data in the device specified by (d) and later.	Page 742 $+(P) [when three operands are set]
$+P		

■Transferring string data

Instruction symbol	Processing details	Reference
$MOV	Transfers the strings in the device specified by (s) to the device specified by (d) and later.	Page 744 $MOV(P)
$MOVP		
$MOV_WS	Transfers the Unicode strings in the device specified by (s) to the device specified by (d) and later.	Page 746 $MOV(P)_WS
$MOVP_WS		

■Converting 16-bit/32-bit binary data to decimal ASCII

Instruction symbol	Processing details	Reference
BINDA	Converts the 1-word binary data in the device specified by (s) to decimal ASCII data, and stores the converted data in the word device number specified by (d) and later.	Page 748 BINDA(P)(_U)
BINDAP		
BINDA_U		
BINDAP_U		
DBINDA	Converts the 2-word binary data in the device specified by (s) to decimal ASCII data, and stores the converted data in the word device number specified by (d) and later.	Page 753 DBINDA(P)(_U)
DBINDAP		
DBINDA_U		
DBINDAP_U		

■Converting 16-bit/32-bit binary data to hexadecimal ASCII

Instruction symbol	Processing details	Reference
BINHA	Converts the 1-word binary data in the device specified by (s) to hexadecimal ASCII data, and stores the converted data in the word device number specified by (d) and later.	Page 758 BINHA(P)
BINHAP		
DBINHA	Converts the 2-word binary data in the device specified by (s) to hexadecimal ASCII data, and stores the converted data in the word device number specified by (d) and later.	Page 762 DBINHA(P)
DBINHAP		

■Converting 16-bit/32-bit binary data to string data

Instruction symbol	Processing details	Reference
STR	Converts the 1-word binary data in the device specified by (s2) to a decimal string consisting of the total number of digits and the number of digits in the decimal part in the device specified by (s1), and stores the converted data in the device specified by (d).	Page 766 STR(P)(_U)
STRP		
STR_U		
STRP_U		
DSTR	Converts the 2-word binary data in the device specified by (s2) to a decimal string consisting of the total number of digits and the number of digits in the decimal part in the device specified by (s1), and stores the converted data in the device specified by (d).	Page 769 DSTR(P)(_U)
DSTRP		
DSTR_U		
DSTRP_U		

■Converting BCD 4-digit/8-digit data to decimal ASCII code

Instruction symbol	Processing details	Reference
BCDDA	Converts the 1-word BCD data in the device specified by (s) to decimal ASCII data, and stores the converted data in the word device number specified by (d) and later.	Page 772 BCDDA(P)
BCDDAP		
DBCDDA	Converts the 2-word BCD data in the device specified by (s) to decimal ASCII data, and stores the converted data in the word device number specified by (d) and later.	Page 776 DBCDDA(P)
DBCDDAP		

■Converting single-precision real number to string data

Instruction symbol	Processing details	Reference
ESTR	Converts the single-precision real number in the device specified by (s1) to a string, and stores the converted data in the device specified by (d).	Page 781 ESTR(P)
ESTRP		

■Converting hexadecimal binary data to hexadecimal ASCII code

Instruction symbol	Processing details	Reference
INT2ASC	Converts the 1-word binary data in the device number specified by (s) and later to hexadecimal ASCII, and stores the converted data by the number of characters in the device specified by (n) in the word device number specified by (d) and later.	Page 787 INT2ASC(P)
INT2ASCP		

■Converting multiple data for string concatenation

Instruction symbol	Processing details	Reference
SPF	Converts multiple data to a string according to the specified display format, concatenates the string with any string, and outputs them as a single string.	Page 789 SPF(P)
SPFP		

■Converting Unicode string to Shift JIS string

Instruction symbol	Processing details	Reference
WS2SJIS	Converts the Unicode string in the device specified by (s) to the shift JIS string, and stores the converted data in the device specified by (d).	Page 805 WS2SJIS(P)
WS2SJISP		

■Converting Shift JIS string to Unicode string (without byte order mark)

Instruction symbol	Processing details	Reference
SJIS2WS	Converts the shift JIS string in the device specified by (s) to a Unicode string, and stores the converted data in the device specified by (d).	Page 808 SJIS2WS(P)
SJIS2WSP		

■Converting Shift JIS to Unicode (with byte order mark)

Instruction symbol	Processing details	Reference
SJIS2WSB	Converts the shift JIS string in the device specified by (s) to the Unicode string, add a byte order mark to the head of the converted data, and stores it in the device specified by (d).	Page 810 SJIS2WSB(P)
SJIS2WSBP		

■Detecting a string length

Instruction symbol	Processing details	Reference
LEN	Stores the length (the number of characters) of the string data, which is stored in the device specified by (s), in the device specified by (d).	Page 812 LEN(P)
LENP		

■Extracting string data from the right/left

Instruction symbol	Processing details	Reference
RIGHT	Stores the (n) characters from the last character of the string, which is stored in the device specified by (s), in the device specified by (d).	Page 814 RIGHT(P)
RIGHTP		
LEFT	Stores the (n) characters from the first character of the string, which is stored in the device specified by (s), in the device specified by (d).	Page 816 LEFT(P)
LEFTP		

■Extracting/replacing the specified string data

Instruction symbol	Processing details	Reference
MIDR	Retrieves the string in the device specified by (s1) by the number of specified characters from the location in the device specified by (s2), and stores the retrieved data in the device specified by (d).	Page 818 MIDR(P)
MIDRP		
MIDW	Retrieves the specified number of characters from the string in the device specified by (s1), and stores the retrieved data at the location specified by (s2) in the string stored in the device specified by (d).	Page 820 MIDW(P)
MIDWP		

■Searching string data

Instruction symbol	Processing details	Reference
INSTR	Searches the string in the device specified by (s2), starting from the (s3)th character, for the string in the device specified by (s1), and stores the matching location in the device specified by (d).	Page 823 INSTR(P)
INSTRP		

■Inserting string data

Instruction symbol	Processing details	Reference
STRINS	Inserts the string data in the device specified by (s1) to the (s2)th character (insertion position) from the head of the string data in the device specified by (d).	Page 825 STRINS(P)
STRINSP		

■Deleting string data

Instruction symbol	Processing details	Reference
STRDEL	Deletes the (n) characters starting from the position (deletion start position) specified by the (s)th character from the head of the string data in the device specified by (d).	Page 827 STRDEL(P)
STRDELP		

Real value processing

Floating-point instruction

■Comparing single-precision real numbers

Instruction symbol	Processing details	Reference
LDE=, ANDE=, ORE=	Performs a comparison operation of a single-precision real number. (Devices are used as a normally open contact.)	Page 829 LDE□, ANDE□, ORE□
LDE<>, ANDE<>, ORE<>		
LDE>, ANDE>, ORE>		
LDE<=, ANDE<=, ORE<=		
LDE<, ANDE<, ORE<		
LDE>=, ANDE>=, ORE>=		

■Comparing double-precision real numbers

Instruction symbol	Processing details	Reference
LDED=, ANDED=, ORED=	Performs a comparison operation of a double-precision real number. (Devices are used as a normally open contact.)	Page 831 LDED□, ANDED□, ORED□
LDED<>, ANDED<>, ORED<>		
LDED>, ANDED>, ORED>		
LDED<=, ANDED<=, ORED<=		
LDED<, ANDED<, ORED<		
LDED>=, ANDED>=, ORED>=		

■Adding/subtracting single-precision real numbers

Instruction symbol	Processing details	Reference
E+	Adds single-precision real numbers. (Using two operands)	Page 834 E+(P) [when two operands are set]
E+P		
E+	Adds single-precision real numbers. (Using three operands)	Page 836 E+(P) [when three operands are set]
E+P		
E-	Performs subtraction between single-precision real numbers. (Using two operands)	Page 838 E-(P) [when two operands are set]
E-P		
E-	Performs subtraction between single-precision real numbers. (Using three operands)	Page 840 E-(P) [when three operands are set]
E-P		

■Adding/subtracting double-precision real numbers

Instruction symbol	Processing details	Reference
ED+	Adds double-precision real numbers. (Using two operands)	Page 842 ED+(P) [when two operands are set]
ED+P		
ED+	Adds double-precision real numbers. (Using three operands)	Page 844 ED+(P) [when three operands are set]
ED+P		
ED-	Performs subtraction between double-precision real numbers. (Using two operands)	Page 846 ED-(P) [when two operands are set]
ED-P		
ED-	Performs subtraction between double-precision real numbers. (Using three operands)	Page 848 ED-(P) [when three operands are set]
ED-P		

■Multiplying/dividing single-precision real numbers

Instruction symbol	Processing details	Reference
E*	Multiplies single-precision real numbers.	Page 850 E*(P)
E*P		
E/	Performs division between single-precision real numbers.	Page 852 E/(P)
E/P		

■Multiplying/dividing double-precision real numbers

Instruction symbol	Processing details	Reference
ED*	Multiplies double-precision real numbers.	Page 854 ED*(P)
ED*P		
ED/	Performs division between double-precision real numbers.	Page 856 ED/(P)
ED/P		

■Converting 16-bit/32-bit signed binary data to single-precision real number

Instruction symbol	Processing details	Reference
INT2FLT	Converts the 16-bit signed binary data in the device specified by (s) to a single-precision real number, and stores the converted data in the device specified by (d).	Page 858 INT2FLT(P)
INT2FLTP		
DINT2FLT	Converts the 32-bit signed binary data in the device specified by (s) to a single-precision real number, and stores the real number in the device specified by (d).	Page 860 DINT2FLT(P)
DINT2FLTP		

■Converting 16-bit/32-bit unsigned binary data to single-precision real number

Instruction symbol	Processing details	Reference
UINT2FLT	Converts the 16-bit unsigned binary data in the device specified by (s) to a single-precision real number, and stores the real number in the device specified by (d).	Page 859 UINT2FLT(P)
UINT2FLTP		
UDINT2FLT	Converts the 32-bit unsigned binary data in the device specified by (s) to a single-precision real number, and stores the real number in the device specified by (d).	Page 862 UDINT2FLT(P)
UDINT2FLTP		

■Converting double-precision real number to single-precision real number

Instruction symbol	Processing details	Reference
DBL2FLT	Converts the double-precision real number in the device specified by (s) to a single-precision real number, and stores the real number in the device specified by (d).	Page 864 DBL2FLT(P)
DBL2FLTP		

■Converting 16-bit/32-bit signed binary data to double-precision real number

Instruction symbol	Processing details	Reference
INT2DBL	Converts the 16-bit signed binary data in the device specified by (s) to a double-precision real number, and stores the real number in the device specified by (d).	Page 866 INT2DBL(P)
INT2DBLP		
DINT2DBL	Converts the 32-bit signed binary data in the device specified by (s) to a double-precision real number, and stores the real number in the device specified by (d).	Page 868 DINT2DBL(P)
DINT2DBLP		

■Converting 16-bit/32-bit unsigned binary data to double-precision real number

Instruction symbol	Processing details	Reference
UINT2DBL	Converts the 16-bit unsigned binary data in the device specified by (s) to a double-precision real number, and stores the real number in the device specified by (d).	Page 867 UINT2DBL(P)
UINT2DBLP		
UDINT2DBL	Converts the 32-bit unsigned binary data in the device specified by (s) to a double-precision real number, and stores the real number in the device specified by (d).	Page 869 UDINT2DBL(P)
UDINT2DBLP		

■Converting single-precision real number to double-precision real number

Instruction symbol	Processing details	Reference
FLT2DBL	Converts the single-precision real number in the device specified by (s) to a double-precision real number, and stores the double-precision real number in the device specified by (d).	Page 870 FLT2DBL(P)
FLT2DBLP		

■Converting string data to single-precision real number

Instruction symbol	Processing details	Reference
EVAL EVALP	Converts the string in the device specified by (s) to a single-precision real number, and stores the converted data in the device specified by (d).	Page 872 EVAL(P)

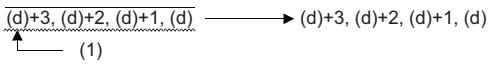
■Converting BCD format data to single-precision real number

Instruction symbol	Processing details	Reference
EREXP EREXPP	Converts the BCD data in the device specified by (s1) to a single-precision real number with the number of decimal positions specified by (s2), and stores the converted data in the device specified by (d).	Page 876 EREXP(P)

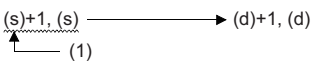
■Inverting the sign of single-precision real number

Instruction symbol	Processing details	Reference
ENEG ENEGP	Inverts the sign of single-precision real number data.  (1) Real number	Page 878 ENEG(P)

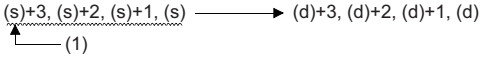
■Inverting the sign of double-precision real number

Instruction symbol	Processing details	Reference
EDNEG EDNEGP	Inverts the sign of double-precision real number data.  (1) Real number	Page 879 EDNEG(P)

■Transferring single-precision real number

Instruction symbol	Processing details	Reference
EMOV EMOVP	Transfers single-precision real number data to the specified device.  (1) Real number	Page 880 EMOV(P)

■Transferring double-precision real number

Instruction symbol	Processing details	Reference
EDMOV EDMOVP	Transfers double-precision real number data to the specified device.  (1) Real number	Page 881 EDMOV(P)

■Calculating the sine of single-precision real number

Instruction symbol	Processing details	Reference
SIN SINP	Calculates the sine of the angle specified by a single-precision real number.	Page 882 SIN(P)

■Calculating the cosine of single-precision real number

Instruction symbol	Processing details	Reference
COS COSP	Calculates the cosine of the angle specified by a single-precision real number.	Page 884 COS(P)

■Calculating the tangent of single-precision real number

Instruction symbol	Processing details	Reference
TAN TANP	Calculates the tangent of the angle specified by a single-precision real number.	Page 886 TAN(P)

■Calculating the arc sine of single-precision real number

Instruction symbol	Processing details	Reference
ASIN	Calculates the angle from the sine specified by a single-precision real number.	Page 888 ASIN(P)
ASINP		

■Calculating the arc cosine of single-precision real number

Instruction symbol	Processing details	Reference
ACOS	Calculates the angle from the cosine specified by a single-precision real number.	Page 890 ACOS(P)
ACOSP		

■Calculating the arc tangent of single-precision real number

Instruction symbol	Processing details	Reference
ATAN	Calculates the angle from the tangent specified by a single-precision real number.	Page 892 ATAN(P)
ATANP		

■Calculating the sine of double-precision real number

Instruction symbol	Processing details	Reference
SIND	Calculates the sine of the angle specified by a double-precision real number.	Page 894 SIND(P)
SINDP		

■Calculating the cosine of double-precision real number

Instruction symbol	Processing details	Reference
COSD	Calculates the cosine of the angle specified by a double-precision real number.	Page 896 COSD(P)
COSDP		

■Calculating the tangent of double-precision real number

Instruction symbol	Processing details	Reference
TAND	Calculates the tangent of the angle specified by a double-precision real number.	Page 898 TAND(P)
TANDP		

■Calculating the arc sine of double-precision real number

Instruction symbol	Processing details	Reference
ASIND	Calculates the angle from the sine specified by a double-precision real number.	Page 900 ASIND(P)
ASINDP		

■Calculating the arc cosine of double-precision real number

Instruction symbol	Processing details	Reference
ACOSD	Calculates the angle from the cosine specified by a double-precision real number.	Page 902 ACOSD(P)
ACOSDP		

■Calculating the arc tangent of double-precision real number

Instruction symbol	Processing details	Reference
ATAND	Calculates the angle from the tangent specified by a double-precision real number.	Page 904 ATAND(P)
ATANDP		

■Calculating the sine of BCD data

Instruction symbol	Processing details	Reference
BSIN	Calculates the sine of the angle specified by a BCD value.	Page 906 BSIN(P)
BSINP	SIN (s) → (d), (d)+1, (d)+2 (d): Sign (d)+1: Integral part (d)+2: Decimal part	

■Calculating the cosine of BCD data

Instruction symbol	Processing details	Reference
BCOS BCOSP	Calculates the cosine of the angle specified by a BCD value. COS (s) ⟶ (d), (d)+1, (d)+2 (d): Sign (d)+1: Integral part (d)+2: Decimal part	Page 908 BCOS(P)

■Calculating the tangent of BCD data

Instruction symbol	Processing details	Reference
BTAN BTANP	Calculates the tangent of the angle specified by a BCD value. TAN(s) ⟶ (d), (d)+1, (d)+2 (d): Sign (d)+1: Integral part (d)+2: Decimal part	Page 910 BTAN(P)

■Calculating the arc sine of BCD data

Instruction symbol	Processing details	Reference
BASIN BASINP	Calculates the arc sine of the angle specified by a BCD value. SIN^{-1} (s) ⟶ (d), (d)+1, (d)+2 (d): Sign (d)+1: Integral part (d)+2: Decimal part	Page 912 BASIN(P)

■Calculating the arc cosine of BCD data

Instruction symbol	Processing details	Reference
BACOS BACOSP	Calculates the arc cosine of the angle specified by a BCD value. COS^{-1} (s) ⟶ (d), (d)+1, (d)+2 (d): Sign (d)+1: Integral part (d)+2: Decimal part	Page 914 BACOS(P)

■Calculating the arc tangent of BCD data

Instruction symbol	Processing details	Reference
BATAN BATANP	Calculates the arc tangent of the angle specified by a BCD value. TAN^{-1} (s) ⟶ (d), (d)+1, (d)+2 (d): Sign (d)+1: Integral part (d)+2: Decimal part	Page 916 BATAN(P)

■Converting single-precision real number angle to radian

Instruction symbol	Processing details	Reference
RAD RADP	Converts the unit of the measure of angle from the degree specified by a single-precision real number to radian. Converting degree to radian (s)+1, (s) ⟶ (d)+1, (d)	Page 918 RAD(P)

■Converting single-precision real number radian to angle

Instruction symbol	Processing details	Reference
DEG DEGP	Converts the unit of the measure of angle from the radian specified by a single-precision real number to the degree. Converting radian to degree (s)+1, (s) ⟶ (d)+1, (d)	Page 920 DEG(P)

■Converting double-precision real number angle to radian

Instruction symbol	Processing details	Reference
RADD RADDP	Converts the unit of the measure of angle from the degree specified by a single-precision real number to radian. Converting degree to radian (s)+3, (s)+2, (s)+1, (s) ⟶ (d)+3, (d)+2, (d)+1, (d)	Page 922 RADD(P)

■Converting double-precision real number radian to angle

Instruction symbol	Processing details	Reference
DEGD DEGDP	Converts the unit of the measure of angle from the radian specified by a double-precision real number to the degree. Converting radian to degree (s)+3, (s)+2, (s)+1, (s) ⟶ (d)+3, (d)+2, (d)+1, (d)	Page 924 DEGD(P)

■Calculating the square root of single-precision real number

Instruction symbol	Processing details	Reference
ESQRT ESQRTP	Calculates the square root of the value specified by a single-precision real number. $\sqrt{(s)+1, (s)}$ ⟶ (d)+1, (d)	Page 926 ESQRT(P)

■Calculating the square root of double-precision real number

Instruction symbol	Processing details	Reference
EDSQRT EDSQRTP	Calculates the square root of the value specified by a double-precision real number. $\sqrt{(s)+3, (s)+2, (s)+1, (s)}$ ⟶ (d)+3, (d)+2, (d)+1, (d)	Page 928 EDSQRT(P)

■Calculating the exponent of single-precision real number

Instruction symbol	Processing details	Reference
EXP EXPP	Calculates the exponent of the value specified by a single-precision real number.	Page 930 EXP(P)

■Calculating the exponent of double-precision real number

Instruction symbol	Processing details	Reference
EXPD EXPDP	Calculates the exponent of the value specified by a double-precision real number.	Page 932 EXPD(P)

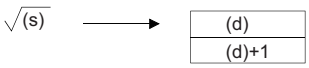
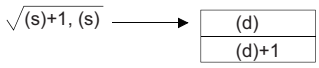
■Calculating the natural logarithm of single-precision real number

Instruction symbol	Processing details	Reference
LOG LOGP	Calculates the logarithm using the natural logarithm (e) of the value specified by a single-precision real number as the base.	Page 934 LOG(P)

■Calculating the natural logarithm of double-precision real number

Instruction symbol	Processing details	Reference
LOGD LOGDP	Calculates the logarithm using the natural logarithm (e) of the value specified by a double-precision real number as the base.	Page 936 LOGD(P)

■Calculating the square root of BCD 4-digit/8-digit data

Instruction symbol	Processing details	Reference
BSQRT BSQRTP	Calculates the square root of the value specified by a BCD 4-digit data.  (d): Integral part (d)+1: Decimal part	Page 938 BSQRT(P)
BDSQRT BDSQRTP	Calculates the square root of the value specified by a BCD 8-digit data.  (d): Integral part (d)+1: Decimal part	Page 940 BDSQRT(P)

■Calculating the exponentiation of single-precision real number

Instruction symbol	Processing details	Reference
POW POWP	Calculates the exponentiation of a single-precision real number.	Page 942 POW(P)

■Calculating the exponentiation of double-precision real number

Instruction symbol	Processing details	Reference
POWD POWDP	Calculates the exponentiation of a double-precision real number.	Page 944 POWD(P)

■Calculating the common logarithm of single-precision real number

Instruction symbol	Processing details	Reference
LOG10 LOG10P	Calculates the logarithm using the common logarithm (using 10 as the base) of the value specified by a single-precision real number.	Page 946 LOG10(P)

■Calculating the common logarithm of double-precision real number

Instruction symbol	Processing details	Reference
LOG10D LOG10DP	Calculates the logarithm using the common logarithm (using 10 as the base) of the value specified by a double-precision real number.	Page 948 LOG10D(P)

■Searching the maximum value of single-precision real number

Instruction symbol	Processing details	Reference
EMAX EMAXP	Searches for the maximum value in the (n) points of single-precision real number block data in the device starting from the one specified by (s), and stores the maximum value in the search result (maximum value) in the device specified by (d).	Page 950 EMAX(P)

■Searching the maximum value of double-precision real number

Instruction symbol	Processing details	Reference
EDMAX EDMAXP	Searches for the maximum value in the (n) points of double-precision real number block data in the device starting from the one specified by (s), and stores the maximum value in the search result (maximum value) in the device specified by (d).	Page 952 EDMAX(P)

■Searching the minimum value of single-precision real number

Instruction symbol	Processing details	Reference
EMIN EMINP	Searches for the minimum value in the (n) points of single-precision real number block data in the device starting from the one specified by (s), and stores the maximum value in the search result (minimum value) in the device specified by (d).	Page 954 EMIN(P)

■Searching the minimum value of double-precision real number

Instruction symbol	Processing details	Reference
EDMIN	Searches for the minimum value in the (n) points of double-precision real number block data in the device starting from the one specified by (s), and stores the maximum value in the search result (minimum value) in the device specified by (d).	Page 956 EDMIN(P)
EDMINP		

Random number

Random number instructions

■Generating random number, changing random sequence

Instruction symbol	Processing details	Reference
RND	Generates a random number between 0 and less than 32767, and stores the random number in the device specified by (d).	Page 958 RND(P)
RNDP		
SRND	Changes the random number sequence according to the content of the 16-bit binary data stored in the device specified by (s).	Page 959 SRND(P)
SRNDP		

Device operation

Index register instructions

■Saving/returning all data of the index register

Instruction symbol	Processing details	Reference
ZPUSH	Saves data of the index register to the area specified by (d).	Page 960 ZPUSH(P)
ZPUSHP		
ZPOP	Reads the data, which has been saved to the area specified by (d)and later, into the index register.	Page 963 ZPOP(P)
ZPOPP		

■Saving/returning the selected data of the index register and long index register

Instruction symbol	Processing details	Reference
ZPUSH	Saves the contents of the index register and long index register specified by (s) to the area specified by (d).	Page 964 ZPUSH(P)
ZPUSHP		
ZPOP	Reads the data, which has been saved to the area specified by (d), into the index register and long index register.	Page 968 ZPOP(P)
ZPOPP		

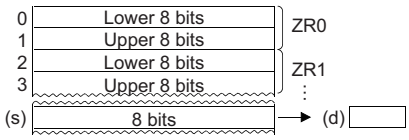
File register operation instructions

■Switching the file register block number

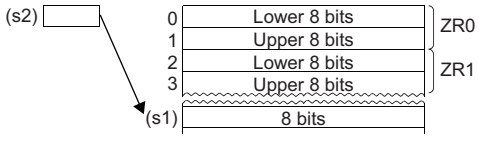
Instruction symbol	Processing details	Reference
RSET	Switches the block number of the file register used in the program to that stored in the device specified by (s).	Page 970 RSET(P)
RSETP		

File register read/write instructions

■Reading 1-byte data from the file register

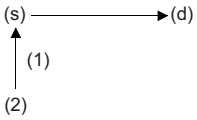
Instruction symbol	Processing details	Reference
ZRRDB	Reads the data from the file register with the specified serial byte number.	Page 972 ZRRDB(P)
ZRRDBP		

■Writing 1-byte data to the file register

Instruction symbol	Processing details	Reference
ZRWRB	Writes the data in the lower bits of the specified device to the file register with the specified serial byte number.	Page 974 ZRWRB(P)
ZRWRBP		

Indirect address read instructions

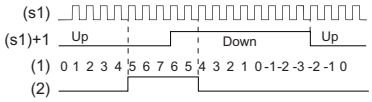
■Reading the indirect address

Instruction symbol	Processing details	Reference
ADRSET ADRSETP	Reads the indirect address of the specified device.  (1) Indirect address of specified device (2) Device name	Page 977 ADRSET(P)

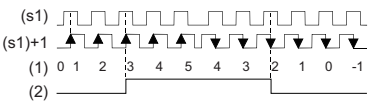
Timer, counter

Special counter instructions

■Counting up or down the current value (1-phase input)

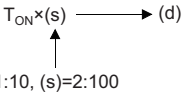
Instruction symbol	Processing details	Reference
UDCNT1	Updates the current value of the specified counter.  (1) Current value of Cn (2) Contact of Cn	Page 979 UDCNT1

■Counting up or down the current value (2-phase input)

Instruction symbol	Processing details	Reference
UDCNT2	Updates the current value of the counter depending on the status of phases A and B pulses.  (1) Current value of Cn (2) Contact of Cn	Page 981 UDCNT2

Special timer instructions

■Teaching timer

Instruction symbol	Processing details	Reference
TTMR	Measures the on time of the measurement command in seconds, multiplies it by a multiplier, and stores the operation result.  (s)=0:1, (s)=1:10, (s)=2:100 T_{ON}: On time of TTMR	Page 983 TTMR

■Special function timer

Instruction symbol	Processing details	Reference
STMR	Performs the following operations at the four points from the bit device specified by (d) according to on/off of the input condition of the instruction. • (d)+0: Off delay timer output • (d)+1: After-off one-shot timer output • (d)+2: After-on one-shot timer output • (d)+3: On delay + off delay timer	Page 985 STMR

Clock

Clock instructions

■Reading clock data

Instruction symbol	Processing details	Reference
DATERD DATERDP	Reads "year, month, day, hour, minute, second, and day of week" from the clock element of the controller. (d) (d)+1 (d)+2 (d)+3 (d)+4 (d)+5 (d)+6 (d): Year (d)+1: Month (d)+2: Day (d)+3: Hour (d)+4: Minute (d)+5: Second (d)+6: Day of week	Page 988 DATERD(P)

■Writing clock data

Instruction symbol	Processing details	Reference
DATEWR DATEWRP	Writes the clock data stored in the specified device and later to the clock element of the controller. (s) (s)+1 (s)+2 (s)+3 (s)+4 (s)+5 (s)+6 (s): Year (s)+1: Month (s)+2: Day (s)+3: Hour (s)+4: Minute (s)+5: Second (s)+6: Day of week	Page 990 DATEWR(P)

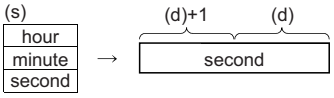
■Adding clock data

Instruction symbol	Processing details	Reference
DATE+ DATE+P	Adds time data. (s1) hour, minute, second + (s2) hour, minute, second → (d) hour, minute, second	Page 992 DATE+(P)

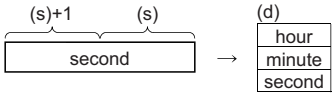
■Subtracting clock data

Instruction symbol	Processing details	Reference
DATE- DATE-P	Subtracts time data. (s1) hour, minute, second - (s2) hour, minute, second → (d) hour, minute, second	Page 994 DATE-(P)

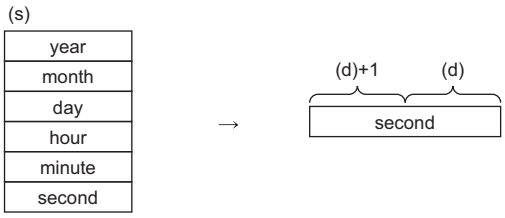
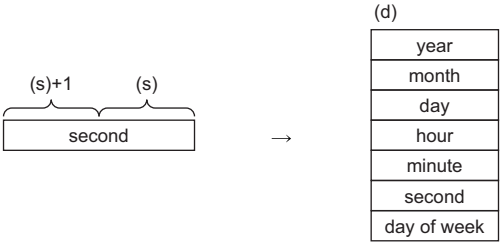
■Converting time data from hour/minute/second to second

Instruction symbol	Processing details	Reference
TIME2SEC TIME2SECP	Converts time data from hour/minute/second to second. 	Page 996 TIME2SEC(P)

■Converting time data from second to hour/minute/second

Instruction symbol	Processing details	Reference
SEC2TIME SEC2TIMEP	Converts time data from second to hour/minute/second. 	Page 998 SEC2TIME(P)

■Converting date and time data

Instruction symbol	Processing details	Reference
DATE2SEC DATE2SECP DATE2SEC_U DATE2SECP_U	Converts date and time data (year/month/day/time/minute/second) into second data. 	Page 1000 DATE2SEC(P)(_U)
SEC2DATE SEC2DATEP SEC2DATE_U SEC2DATEP_U	Converts second data into date and time data (year/month/day/time/minute/second/day of week). 	Page 1002 SEC2DATE(P)(_U)

■Comparing date data

Instruction symbol	Processing details	Reference
LDDT=, ANDDT=, ORDT= LDDT<>, ANDDT<>, ORDT<> LDDT>, ANDDT>, ORDT> LDDT<=, ANDDT<=, ORDT<= LDDT<, ANDDT<, ORDT< LDDT>=, ANDDT>=, ORDT>=	Compares the specified date data, or compares the date data with the current date.	Page 1004 LDDT□, ANDDT□, ORDT□

■Comparing time data

Instruction symbol	Processing details	Reference
LDTM=, ANDTM=, ORTM=	Compares the specified time data, or compares the specified time data with the current time.	Page 1008 LDTM□, ANDTM□, ORTM□
LDTM<>, ANDTM<>, ORTM<>		
LDTM>, ANDTM>, ORTM>		
LDTM<=, ANDTM<=, ORTM<=		
LDTM<, ANDTM<, ORTM<		
LDTM>=, ANDTM>=, ORTM>=		

■Outputting a comparison result of time data

Instruction symbol	Processing details	Reference
TCMP	Compares the time data to be compared that is specified by (s1), (s2), and (s3) with the time data specified by (s4), and according to the result (small, match, or large), (d), (d)+1, or (d)+2 is turned on.	Page 1011 TCMP(P)
TCMPP		

■Reading expansion clock data

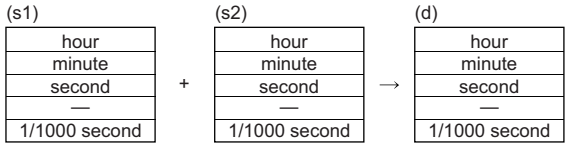
Instruction symbol	Processing details	Reference
S.DATERD	Reads clock data including millisecond from the clock elements in the controller.	Page 1013 S(P).DATERD
SP.DATERD		

(d)
(d)+1
(d)+2
(d)+3
(d)+4
(d)+5
(d)+6
(d)+7

(d): Year
(d)+1: Month
(d)+2: Day
(d)+3: Hour
(d)+4: Minute
(d)+5: Second
(d)+6: Day of week
(d)+7: Millisecond

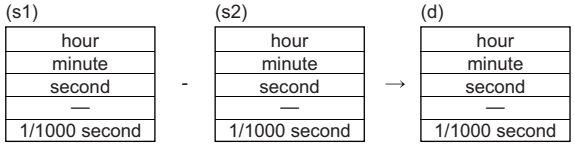
■Adding expansion clock data

Instruction symbol	Processing details	Reference
S.DATE+	Adds time data.	Page 1015 S(P).DATE+
SP.DATE+		



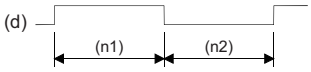
■Subtracting expansion clock data

Instruction symbol	Processing details	Reference
S.DATE-	Subtracts time data.	Page 1017 S(P).DATE-
SP.DATE-		



Timing check instructions

■Generating timing pulses

Instruction symbol	Processing details	Reference
DUTY	Turns on the user timing clock for the specified number of scans and off for the specified number of scans.  (n1): (n1) scans (n2): (n2) scans (d): SM420 to SM424	Page 1019 DUTY

■Measuring time of the specified data

Instruction symbol	Processing details	Reference
TIMCHK	Measures the on time of the input condition and, if the on time has continued as specified or longer, turns on the device specified by (d).	Page 1021 TIMCHK

Module access

Module access instructions

■Performing I/O refresh

Instruction symbol	Processing details	Reference
RFS	Performs partial refresh of the relevant input/output during one scan.	Page 1023 RFS(P)
RFSP		

■Selecting refresh to be performed

Instruction symbol	Processing details	Reference
COM	Performs refresh and service processing for various modules when the input condition is met.	Page 1025 COM(P)
COMP		

■Performing module refresh

Instruction symbol	Processing details	Reference
S.ZCOM	Performs refresh processing for the specified modules.	Page 1027 S(P).ZCOM
SP.ZCOM		

■Reading 1-word/2-word data from another module (16-bit specification)

Instruction symbol	Processing details	Reference
FROM	Reads (n) words of data in units of 16 bits from the buffer memory areas of the intelligent function modules.	Page 1029 FROM(P), DFROM(P)
FROMP		
DFROM	Reads (n)×2 words of data in units of 16 bits from the buffer memory areas of the intelligent function modules.	
DFROMP		

■Writing 1-word/2-word data to a module (16-bit specification)

Instruction symbol	Processing details	Reference
TO	Writes (n) words of data in units of 16 bits to the buffer memory areas of the intelligent function module and host controller.	Page 1032 TO(P), DTO(P)
TOP		
DTO	Writes (n)×2 words of data in units of 16 bits to the buffer memory areas of the intelligent function module and host controller.	
DTOP		

■Reading 1-word/2-word data from another module (32-bit specification)

Instruction symbol	Processing details	Reference
FROMD	Reads (n) words of data in units of 32 bits from the buffer memory areas of the intelligent function modules.	Page 1037 FROMD(P), DFROMD(P)
FROMDP		
DFROMD	Reads (n)×2 words of data in units of 32 bits from the buffer memory areas of the intelligent function modules.	
DFROMDP		

■Writing 1-word/2-word data to a module (32-bit specification)

Instruction symbol	Processing details	Reference
TOD	Writes (n) words of data in units of 32 bits to the buffer memory areas of the intelligent function module and host controller.	Page 1041 TOD(P), DTOD(P)
TODP		
DTOD	Writes (n)×2 words of data in units of 32 bits to the buffer memory areas of the intelligent function module and host controller.	
DTODP		

■Reading the module model name

Instruction symbol	Processing details	Reference
TYPERD	Reads the module model name mounted on the slot specified by (H), and stores the model name in the device areas specified by (d) and later.	Page 1046 TYPERD(P)
TYPERDP		

■Reading module specific information

Instruction symbol	Processing details	Reference
UNIINFRD	Reads the module information by the number of points specified by (n) from the module specified by (H), and stores the information in the device areas specified by (d) and later.	Page 1050 UNIINFRD(P)
UNIINFRDP		

PID operation instruction

PID operation

Instruction symbol	Processing details	Reference
PID	Performs PID operation using the values set in (s1) to (s3), and stores the operation result in (d) at each cycle of sampling time.	Page 1069 PID

PID control instructions

PID control instructions (inexact differential)

■Registering the PID control data to the controller

Instruction symbol	Processing details	Reference
S.PIDINIT	Stores the PID control data by the number of loops used that is set in the device number specified by (s) and later altogether in the controller to enable PID control.	Page 1084 S(P).PIDINIT
SP.PIDINIT		

■Performing PID operation

Instruction symbol	Processing details	Reference
S.PIDCONT	Measures the sampling cycle and performs PID operation.	Page 1087 S(P).PIDCONT
SP.PIDCONT		

■Stopping/starting the operation of specified loop number

Instruction symbol	Processing details	Reference
S.PIDSTOP	Stops the PID operation of the loop number in the device specified by (s).	Page 1090 S(P).PIDSTOP
SP.PIDSTOP		
S.PIDRUN	Starts the PID operation of the loop number in the device specified by (s).	Page 1091 S(P).PIDRUN
SP.PIDRUN		

■Changing the parameters of specified loop number

Instruction symbol	Processing details	Reference
S.PIDPRMW	Changes the operation parameter of the loop number in the device specified by (s1) to the PID control data stored in the device number specified by (s2) and later.	Page 1092 S(P).PIDPRMW
SP.PIDPRMW		

PID control instructions (exact differential)

■Registering the PID control data to the controller

Instruction symbol	Processing details	Reference
PIDINIT	Stores the PID control data by the number of loops used that is set in the device number specified by (s) and later altogether in the controller to enable PID control.	Page 1096 PIDINIT(P)
PIDINITP		

■Performing PID operation

Instruction symbol	Processing details	Reference
PIDCONT	Measures the sampling cycle and performs PID operation.	Page 1098 PIDCONT(P)
PIDCONTP		

■Stopping/starting the operation of specified loop number

Instruction symbol	Processing details	Reference
PIDSTOP	Stops the PID operation of the loop number in the device specified by (s).	Page 1101 PIDSTOP(P)
PIDSTOPP		
PIDRUN	Starts the PID operation of the loop number in the device specified by (s).	Page 1102 PIDRUN(P)
PIDRUNP		

■Changing the parameters of specified loop number

Instruction symbol	Processing details	Reference
PIDPRMW	Changes the operation parameter of the loop number in the device specified by (s1) to the PID control data stored in the device number specified by (s2) and later.	Page 1103 PIDPRMW(P)
PIDPRMWP		

Add-on instructions

Add-on execution instruction

Instruction symbol	Processing details	Reference
ADDONEXE	Instructs to execute an installed add-on.	Page 1105 ADDONEXE(P)
ADDONEXEP		

7 NETWORK INSTRUCTION LIST

7.1 Ethernet Instructions

Open/close processing instructions

■Opening a connection

Instruction symbol	Processing details	Reference
GP.SOCOPEN SP.SOCOPEN	Opens the connection specified by (s1).	Page 1110 GP.SOCOPEN, SP.SOCOPEN

■Closing a connection

Instruction symbol	Processing details	Reference
GP.SOCCLOSE SP.SOCCLOSE	Closes the connection specified by (s1). (Closing a connection)	Page 1115 GP.SOCCLOSE, SP.SOCCLOSE

Socket communications instructions

■Reading receive data during the END processing

Instruction symbol	Processing details	Reference
GP.SOCRCV SP.SOCRCV	Reads the receive data of the connection specified by (s1) during END processing from the socket communication receive data area.	Page 1117 GP.SOCRCV, SP.SOCRCV

■Reading receive data when the instruction is executed

Instruction symbol	Processing details	Reference
G.SOCRCVS S.SOCRCVS	Reads the receive data of the connection specified by (s) during instruction execution from the socket communication receive data area.	Page 1121 G.SOCRCVS, S.SOCRCVS

■Sending data

Instruction symbol	Processing details	Reference
GP.SOCSND SP.SOCSND	Sends the data in the device specified by (s3) to the external device of the connection specified by (s1).	Page 1124 GP.SOCSND, SP.SOCSND

■Reading connection information

Instruction symbol	Processing details	Reference
GP.SOCCINF SP.SOCCINF	Reads the connection information of the connection specified by (s1).	Page 1128 GP.SOCCINF, SP.SOCCINF

■Changing the communication target (UDP/IP)

Instruction symbol	Processing details	Reference
GP.SOCCSET SP.SOCCSET	Changes the communication target IP address and port number of the connection specified by (s1). (UDP/IP communications only)	Page 1131 GP.SOCCSET, SP.SOCCSET

■Changing the receive mode

Instruction symbol	Processing details	Reference
GP.SOCRMODE SP.SOCRMODE	Changes the TCP/TLS receive mode and receive data size for the connection specified by (s1).	Page 1133 GP.SOCRMODE, SP.SOCRMODE

■Reading socket communications receive data

Instruction symbol	Processing details	Reference
G.SOCRDATA	Reads data by the number of words specified by (n) from the socket communication receive data area of the connection specified by (s1), and stores them in the device specified by (d) and later.	Page 1137 G(P).SOCRDATA, S(P).SOCRDATA
GP.SOCRDATA		
S.SOCRDATA		
SP.SOCRDATA		

Predefined protocol support function instruction

■Executing the registered protocols

Instruction symbol	Processing details	Reference
GP.ECPRTCL	Executes the protocol specified by the communication protocol support tool of the engineering tool.	Page 1139 GP.ECPRTCL, SP.ECPRTCL
SP.ECPRTCL		

SLMP frame send instruction

■Sending an SLMP frame

Instruction symbol	Processing details	Reference
G.SLMPSND	Sends SLMP messages to the SLMP-compatible device.	Page 1147 G(P).SLMPSND, J(P).SLMPSND, SP.SLMPSND
GP.SLMPSND		
J.SLMPSND		
JP.SLMPSND		
SP.SLMPSND		

■Sending an encrypted SLMP frame

Instruction symbol	Processing details	Reference
G.SLMPSNDC	Sends encrypted SLMP messages to the SLMP-compatible device.	Page 1156 G(P).SLMPSNDC
GP.SLMPSNDC		

File transfer function instructions

■Sending FTP client files

Instruction symbol	Processing details	Reference
GP.FTPCPUT	Sends files in the controller, which are specified by (s2), to the folder path of the FTP server, which is specified by (s3).	Page 1165 GP.FTPCPUT, SP.FTPPUT
SP.FTPPUT		

■Retrieving FTP client files

Instruction symbol	Processing details	Reference
GP.FTPCGET	Retrieves files on the FTP server, which are specified by (s2), to the folder path of the controller, which is specified by (s3).	Page 1171 GP.FTPCGET, SP.FTPGET
SP.FTPGET		

8 STANDARD FUNCTION/STANDARD FUNCTION BLOCK LISTS

How to read the list is shown below.

Item	Description
Function symbol and function block symbol	A function and function block name are shown.
Processing details	An overview of the functions and function blocks is explained.
Reference	Indicates the reference of detailed information.

8.1 Standard Functions

Type conversion functions

■Converting BOOL to WORD/DWORD

Function symbol	Processing details	Reference
BOOL_TO_WORD	Converts a value from BOOL data type to WORD data type.	Page 1178 BOOL_TO_WORD(_E)
BOOL_TO_WORD_E		
BOOL_TO_DWORD	Converts a value from BOOL data type to DWORD data type.	Page 1180 BOOL_TO_DWORD(_E)
BOOL_TO_DWORD_E		

■Converting BOOL to INT/DINT

Function symbol	Processing details	Reference
BOOL_TO_INT	Converts a value from BOOL data type to INT data type.	Page 1181 BOOL_TO_INT(_E)
BOOL_TO_INT_E		
BOOL_TO_DINT	Converts a value from BOOL data type to DINT data type.	Page 1182 BOOL_TO_DINT(_E)
BOOL_TO_DINT_E		

■Converting BOOL to TIME

Function symbol	Processing details	Reference
BOOL_TO_TIME	Converts a value from BOOL data type to TIME data type.	Page 1183 BOOL_TO_TIME(_E)
BOOL_TO_TIME_E		

■Converting BOOL to STRING

Function symbol	Processing details	Reference
BOOL_TO_STRING	Converts a value from BOOL data type to STRING data type.	Page 1184 BOOL_TO_STRING(_E)
BOOL_TO_STRING_E		

■Converting WORD to BOOL

Function symbol	Processing details	Reference
WORD_TO_BOOL	Converts a value from WORD data type to BOOL data type.	Page 1185 WORD_TO_BOOL(_E)
WORD_TO_BOOL_E		

■Converting WORD to DWORD

Function symbol	Processing details	Reference
WORD_TO_DWORD	Converts a value from WORD data type to DWORD data type.	Page 1186 WORD_TO_DWORD(_E)
WORD_TO_DWORD_E		

■Converting WORD to INT/DINT

Function symbol	Processing details	Reference
WORD_TO_INT	Converts a value from WORD data type to INT data type.	Page 1187 WORD_TO_INT(_E)
WORD_TO_INT_E		
WORD_TO_DINT	Converts a value from WORD data type to DINT data type.	Page 1188 WORD_TO_DINT(_E)
WORD_TO_DINT_E		

■Converting WORD to TIME

Function symbol	Processing details	Reference
WORD_TO_TIME	Converts a value from WORD data type to TIME data type.	Page 1190 WORD_TO_TIME(_E)
WORD_TO_TIME_E		

■Converting WORD to STRING

Function symbol	Processing details	Reference
WORD_TO_STRING	Converts a value from WORD data type to STRING data type.	Page 1191 WORD_TO_STRING(_E)
WORD_TO_STRING_E		

■Converting DWORD to BOOL

Function symbol	Processing details	Reference
DWORD_TO_BOOL	Converts a value from DWORD data type to BOOL data type.	Page 1192 DWORD_TO_BOOL(_E)
DWORD_TO_BOOL_E		

■Converting DWORD to WORD

Function symbol	Processing details	Reference
DWORD_TO_WORD	Converts a value from DWORD data type to WORD data type.	Page 1193 DWORD_TO_WORD(_E)
DWORD_TO_WORD_E		

■Converting DWORD to INT/DINT

Function symbol	Processing details	Reference
DWORD_TO_INT	Converts a value from DWORD data type to INT data type.	Page 1195 DWORD_TO_INT(_E)
DWORD_TO_INT_E		
DWORD_TO_DINT	Converts a value from DWORD data type to DINT data type.	Page 1197 DWORD_TO_DINT(_E)
DWORD_TO_DINT_E		

■Converting DWORD to TIME

Function symbol	Processing details	Reference
DWORD_TO_TIME	Converts a value from DWORD data type to TIME data type.	Page 1198 DWORD_TO_TIME(_E)
DWORD_TO_TIME_E		

■Converting DWORD to STRING

Function symbol	Processing details	Reference
DWORD_TO_STRING	Converts a value from DWORD data type to STRING data type.	Page 1199 DWORD_TO_STRING(_E)
DWORD_TO_STRING_E		

■Converting INT to BOOL

Function symbol	Processing details	Reference
INT_TO_BOOL	Converts a value from INT data type to BOOL data type.	Page 1200 INT_TO_BOOL(_E)
INT_TO_BOOL_E		

■Converting INT to WORD/DWORD

Function symbol	Processing details	Reference
INT_TO_WORD	Converts a value from INT data type to WORD data type.	Page 1201 INT_TO_WORD(_E)
INT_TO_WORD_E		
INT_TO_DWORD	Converts a value from INT data type to DWORD data type.	Page 1202 INT_TO_DWORD(_E)
INT_TO_DWORD_E		

■Converting INT to DINT

Function symbol	Processing details	Reference
INT_TO_DINT	Converts a value from INT data type to DINT data type.	Page 1204 INT_TO_DINT(_E)
INT_TO_DINT_E		

■Converting INT to BCD

Function symbol	Processing details	Reference
INT_TO_BCD	Converts a value from INT data type to BCD data type.	Page 1205 INT_TO_BCD(_E)
INT_TO_BCD_E		

■Converting INT to REAL/LREAL

Function symbol	Processing details	Reference
INT_TO_REAL	Converts a value from INT data type to REAL data type.	Page 1207 INT_TO_REAL(_E)
INT_TO_REAL_E		
INT_TO_LREAL	Converts a value from INT data type to LREAL data type.	Page 1208 INT_TO_LREAL(_E)
INT_TO_LREAL_E		

■Converting INT to TIME

Function symbol	Processing details	Reference
INT_TO_TIME	Converts a value from INT data type to TIME data type.	Page 1209 INT_TO_TIME(_E)
INT_TO_TIME_E		

■Converting INT to STRING

Function symbol	Processing details	Reference
INT_TO_STRING	Converts a value from INT data type to STRING data type.	Page 1210 INT_TO_STRING(_E)
INT_TO_STRING_E		

■Converting DINT to BOOL

Function symbol	Processing details	Reference
DINT_TO_BOOL	Converts a value from DINT data type to BOOL data type.	Page 1212 DINT_TO_BOOL(_E)
DINT_TO_BOOL_E		

■Converting DINT to WORD/DWORD

Function symbol	Processing details	Reference
DINT_TO_WORD	Converts a value from DINT data type to WORD data type.	Page 1213 DINT_TO_WORD(_E)
DINT_TO_WORD_E		
DINT_TO_DWORD	Converts a value from DINT data type to DWORD data type.	Page 1215 DINT_TO_DWORD(_E)
DINT_TO_DWORD_E		

■Converting DINT to INT

Function symbol	Processing details	Reference
DINT_TO_INT	Converts a value from DINT data type to INT data type.	Page 1216 DINT_TO_INT(_E)
DINT_TO_INT_E		

■Converting DINT to BCD

Function symbol	Processing details	Reference
DINT_TO_BCD	Converts a value from DINT data type to BCD data type.	Page 1217 DINT_TO_BCD(_E)
DINT_TO_BCD_E		

■Converting DINT to REAL/LREAL

Function symbol	Processing details	Reference
DINT_TO_REAL	Converts a value from DINT data type to REAL data type.	Page 1219 DINT_TO_REAL(_E)
DINT_TO_REAL_E		
DINT_TO_LREAL	Converts a value from DINT data type to LREAL data type.	Page 1220 DINT_TO_LREAL(_E)
DINT_TO_LREAL_E		

■Converting DINT to TIME

Function symbol	Processing details	Reference
DINT_TO_TIME	Converts a value from DINT data type to TIME data type.	Page 1221 DINT_TO_TIME(_E)
DINT_TO_TIME_E		

■Converting DINT to STRING

Function symbol	Processing details	Reference
DINT_TO_STRING	Converts a value from DINT data type to STRING data type.	Page 1222 DINT_TO_STRING(_E)
DINT_TO_STRING_E		

■Converting BCD to INT/DINT

Function symbol	Processing details	Reference
BCD_TO_INT	Converts a value from BCD data type to INT data type.	Page 1224 BCD_TO_INT(_E)
BCD_TO_INT_E		
BCD_TO_DINT	Converts a value from BCD data type to DINT data type.	Page 1226 BCD_TO_DINT(_E)
BCD_TO_DINT_E		

■Converting BCD to STRING

Function symbol	Processing details	Reference
BCD_TO_STRING	Converts a value from BCD data type to STRING data type.	Page 1229 BCD_TO_STRING(_E)
BCD_TO_STRING_E		

■Converting REAL to INT/DINT

Function symbol	Processing details	Reference
REAL_TO_INT	Converts a value from REAL data type to INT data type.	Page 1231 REAL_TO_INT(_E)
REAL_TO_INT_E		
REAL_TO_DINT	Converts a value from REAL data type to DINT data type.	Page 1233 REAL_TO_DINT(_E)
REAL_TO_DINT_E		

■Converting REAL to LREAL

Function symbol	Processing details	Reference
REAL_TO_LREAL	Converts a value from REAL data type to LREAL data type.	Page 1235 REAL_TO_LREAL(_E)
REAL_TO_LREAL_E		

■Converting REAL to STRING

Function symbol	Processing details	Reference
REAL_TO_STRING	Converts a REAL data type value to STRING data type (exponential form).	Page 1236 REAL_TO_STRING(_E)
REAL_TO_STRING_E		

■Converting LREAL to INT/DINT

Function symbol	Processing details	Reference
LREAL_TO_INT	Converts a value from LREAL data type to INT data type.	Page 1239 LREAL_TO_INT(_E)
LREAL_TO_INT_E		
LREAL_TO_DINT	Converts a value from LREAL data type to DINT data type.	Page 1240 LREAL_TO_DINT(_E)
LREAL_TO_DINT_E		

■Converting LREAL to REAL

Function symbol	Processing details	Reference
LREAL_TO_REAL	Converts a value from LREAL data type to REAL data type.	Page 1241 LREAL_TO_REAL(_E)
LREAL_TO_REAL_E		

■Converting TIME to BOOL

Function symbol	Processing details	Reference
TIME_TO_BOOL	Converts a value from TIME data type to BOOL data type.	Page 1242 TIME_TO_BOOL(_E)
TIME_TO_BOOL_E		

■Converting TIME to WORD/DWORD

Function symbol	Processing details	Reference
TIME_TO_WORD	Converts a value from TIME data type to WORD data type.	Page 1243 TIME_TO_WORD(_E)
TIME_TO_WORD_E		
TIME_TO_DWORD	Converts a value from TIME data type to DWORD data type.	Page 1244 TIME_TO_DWORD(_E)
TIME_TO_DWORD_E		

■Converting TIME to INT/DINT

Function symbol	Processing details	Reference
TIME_TO_INT	Converts a value from TIME data type to INT data type.	Page 1245 TIME_TO_INT(_E)
TIME_TO_INT_E		
TIME_TO_DINT	Converts a value from TIME data type to DINT data type.	Page 1246 TIME_TO_DINT(_E)
TIME_TO_DINT_E		

■Converting TIME to STRING

Function symbol	Processing details	Reference
TIME_TO_STRING	Converts a value from TIME data type to STRING data type.	Page 1247 TIME_TO_STRING(_E)
TIME_TO_STRING_E		

■Converting STRING to BOOL

Function symbol	Processing details	Reference
STRING_TO_BOOL	Converts a value from STRING data type to BOOL data type.	Page 1249 STRING_TO_BOOL(_E)
STRING_TO_BOOL_E		

■Converting STRING to WORD/DWORD

Function symbol	Processing details	Reference
STRING_TO_WORD	Converts a value from STRING data type to WORD data type.	Page 1250 STRING_TO_WORD(_E)
STRING_TO_WORD_E		
STRING_TO_DWORD	Converts a value from STRING data type to DWORD data type.	Page 1251 STRING_TO_DWORD(_E)
STRING_TO_DWORD_E		

■Converting STRING to INT/DINT

Function symbol	Processing details	Reference
STRING_TO_INT	Converts a value from STRING data type to INT data type.	Page 1252 STRING_TO_INT(_E)
STRING_TO_INT_E		
STRING_TO_DINT	Converts a value from STRING data type to DINT data type.	Page 1254 STRING_TO_DINT(_E)
STRING_TO_DINT_E		

■Converting STRING to BCD

Function symbol	Processing details	Reference
STRING_TO_BCD	Converts a value from STRING data type to BCD data type.	Page 1256 STRING_TO_BCD(_E)
STRING_TO_BCD_E		

■Converting STRING to REAL

Function symbol	Processing details	Reference
STRING_TO_REAL	Converts a value from STRING data type to REAL data type.	Page 1258 STRING_TO_REAL(_E)
STRING_TO_REAL_E		

■Converting STRING to TIME

Function symbol	Processing details	Reference
STRING_TO_TIME	Converts a value from STRING data type to TIME data type.	Page 1261 STRING_TO_TIME(_E)
STRING_TO_TIME_E		

■Converting bit array to INT/DINT

Function symbol	Processing details	Reference
BITARR_TO_INT	Converts the specified number of bits in a bit array to an INT data type value.	Page 1262 BITARR_TO_INT(_E)
BITARR_TO_INT_E		
BITARR_TO_DINT	Converts the specified number of bits in a bit array to a DINT data type value.	Page 1263 BITARR_TO_DINT(_E)
BITARR_TO_DINT_E		

■Converting INT/DINT to bit array

Function symbol	Processing details	Reference
INT_TO_BITARR	Outputs the lower n bits of the INT data type value to the bit array.	Page 1264 INT_TO_BITARR(_E)
INT_TO_BITARR_E		
DINT_TO_BITARR	Outputs the lower n bits of the DINT data type value to the bit array.	Page 1265 DINT_TO_BITARR(_E)
DINT_TO_BITARR_E		

■Copying the bit array

Function symbol	Processing details	Reference
CPY_BITARR	Copies the bit array by the specified number of bits.	Page 1266 CPY_BITARR(_E)
CPY_BITARR_E		

■Reading/writing/copying the specified bit of the word label

Function symbol	Processing details	Reference
GET_BIT_OF_INT	Reads a value from the specified bit of a word label.	Page 1267 GET_BIT_OF_INT(_E)
GET_BIT_OF_INT_E		
SET_BIT_OF_INT	Writes a value to the specified bit of a word label.	Page 1269 SET_BIT_OF_INT(_E)
SET_BIT_OF_INT_E		
CPY_BIT_OF_INT	Copies the specified bit of the word label to the specified bit of another word label.	Page 1271 CPY_BIT_OF_INT(_E)
CPY_BIT_OF_INT_E		

■Getting the start data

Function symbol	Processing details	Reference
GET_BOOL_ADDR	Outputs the top data of the specified data as the BOOL, INT, or WORD type data.	Page 1273 GET_BOOL_ADDR, GET_INT_ADDR, GET_WORD_ADDR
GET_INT_ADDR		
GET_WORD_ADDR		

Single variable functions

■Calculating the absolute value

Function symbol	Processing details	Reference
ABS	Outputs the absolute value of an input value.	Page 1274 ABS(_E)
ABS_E		

■Calculating the square root

Function symbol	Processing details	Reference
SQRT	Calculates the square root of an input value.	Page 1276 SQRT(_E)
SQRT_E		

■Calculating the natural logarithm

Function symbol	Processing details	Reference
LN	Outputs the natural logarithm (logarithm with base e) of an input value.	Page 1277 LN(_E)
LN_E		

■Calculating the common logarithm

Function symbol	Processing details	Reference
LOG	Outputs the common logarithm (logarithm with base 10) of an input value.	Page 1278 LOG(_E)
LOG_E		

■Calculating the exponent

Function symbol	Processing details	Reference
EXP	Outputs the exponent of an input value.	Page 1280 EXP(_E)
EXP_E		

■Calculating the sine/cosine/tangent

Function symbol	Processing details	Reference
SIN	Outputs the sine of an input value.	Page 1281 SIN(_E)
SIN_E		
COS	Outputs the cosine of an input value.	Page 1282 COS(_E)
COS_E		
TAN	Outputs the tangent of an input value.	Page 1283 TAN(_E)
TAN_E		

■Calculating the arc sine/arc cosine/arc tangent

Function symbol	Processing details	Reference
ASIN	Outputs the arc sine (SIN^{-1}) of an input value.	Page 1284 ASIN(_E)
ASIN_E		
ACOS	Outputs the arc cosine (COS^{-1}) of an input value.	Page 1285 ACOS(_E)
ACOS_E		
ATAN	Outputs the arc tangent (TAN^{-1}) of an input value.	Page 1286 ATAN(_E)
ATAN_E		

Arithmetic operation functions

■Addition

Function symbol	Processing details	Reference
ADD	Outputs the sum of input values ((s1)+(s2)+···+(s28)).	Page 1287 ADD(_E)
ADD_E		

■Multiplication

Function symbol	Processing details	Reference
MUL	Outputs the product of input values ((s1)×(s2)×···×(s28)).	Page 1290 MUL(_E)
MUL_E		

■Subtraction

Function symbol	Processing details	Reference
SUB	Outputs the difference between input values ((s1)-(s2)).	Page 1292 SUB(_E)
SUB_E		

■Division

Function symbol	Processing details	Reference
DIV	Outputs the quotient of input values ((s1)÷(s2)).	Page 1295 DIV(_E)
DIV_E		

■Remainder

Function symbol	Processing details	Reference
MOD	Outputs the remainder of input values ((s1)÷(s2)).	Page 1297 MOD(_E)
MOD_E		

■Exponentiation

Function symbol	Processing details	Reference
EXPT	Outputs the exponentiation of an input value.	Page 1299 EXPT(_E)
EXPT_E		

■Assignment (move operation)

Function symbol	Processing details	Reference
MOVE	Outputs the assignment value of an input value.	Page 1300 MOVE(_E)
MOVE_E		

Bit shift functions

■Shifting data to the left/right by n bit(s)

Function symbol	Processing details	Reference
SHL	Shifts the input value to the left by (n) bit(s), and outputs the operation result.	Page 1302 SHL(_E)
SHL_E		
SHR	Shifts the input value to the right by (n) bit(s), and outputs the operation result.	Page 1304 SHR(_E)
SHR_E		

■Rotating data to the left/right by n bit(s)

Function symbol	Processing details	Reference
ROL	Rotates the input value to the left by (n) bit(s), and outputs the operation result.	Page 1306 ROL(_E)
ROL_E		
ROR	Rotates the input value to the right by (n) bit(s), and outputs the operation result.	Page 1308 ROR(_E)
ROR_E		

Boolean functions

■AND operation, OR operation, XOR operation

Function symbol	Processing details	Reference
AND	Outputs the logical product of input values.	Page 1310 AND(_E), OR(_E), XOR(_E)
AND_E		
OR	Outputs the logical sum of input values.	
OR_E		
XOR	Outputs the exclusive logical sum of input values.	
XOR_E		

■NOT operation

Function symbol	Processing details	Reference
NOT	Outputs the logical NOT of input values.	Page 1313 NOT(_E)
NOT_E		

Selection functions

■Selecting a value

Function symbol	Processing details	Reference
SEL	Outputs the selected input value.	Page 1314 SEL(_E)
SEL_E		

■Selecting the maximum/minimum value

Function symbol	Processing details	Reference
MAX	Outputs the maximum input value.	Page 1316 MAX(_E), MIN(_E)
MAX_E		
MIN	Outputs the minimum input value.	
MIN_E		

■Controlling the upper/lower limit

Function symbol	Processing details	Reference
LIMIT	Outputs an input value that has been controlled in terms of the upper and lower limits.	Page 1318 LIMIT(_E)
LIMIT_E		

■Multiplexer

Function symbol	Processing details	Reference
MUX	Outputs one of the input values.	Page 1321 MUX(_E)
MUX_E		

Comparison functions

■Comparing data

Function symbol	Processing details	Reference
GT	Outputs the comparison result of input values.	Page 1323 GT(_E), GE(_E), EQ(_E), LE(_E), LT(_E)
GT_E		
GE		
GE_E		
EQ		
EQ_E		
LE		
LE_E		
LT		
LT_E		
NE		Page 1325 NE(_E)
NE_E		

String functions

■Detecting a string length

Function symbol	Processing details	Reference
LEN	Detects and outputs the length of the string input.	Page 1327 LEN(_E)
LEN_E		

■Extracting string data from the left/right

Function symbol	Processing details	Reference
LEFT	Extracts and outputs the specified number of characters, starting from the left end of the string input.	Page 1329 LEFT(_E), RIGHT(_E)
LEFT_E		
RIGHT	Extracts and outputs the specified number of characters, starting from the right end of the string input.	
RIGHT_E		

■Extracting string data

Function symbol	Processing details	Reference
MID	Extracts and outputs the specified number of characters, starting from the specified position of the string input.	Page 1332 MID(_E)
MID_E		

■Concatenating string data

Function symbol	Processing details	Reference
CONCAT	Concatenates strings, and outputs the operation result.	Page 1334 CONCAT(_E)
CONCAT_E		

■Inserting string data

Function symbol	Processing details	Reference
INSERT	Inserts a string into another string, and outputs the operation result.	Page 1336 INSERT(_E)
INSERT_E		

■Deleting string data

Function symbol	Processing details	Reference
DELETE	Deletes the specified range in a string, and outputs the operation result.	Page 1338 DELETE(_E)
DELETE_E		

■Replacing string data

Function symbol	Processing details	Reference
REPLACE	Replaces the specified range in a string, and outputs the operation result.	Page 1340 REPLACE(_E)
REPLACE_E		

■Searching string data

Function symbol	Processing details	Reference
FIND	Searches a string, and outputs the operation result.	Page 1343 FIND(_E)
FIND_E		

Time data type functions

■Addition

Function symbol	Processing details	Reference
ADD_TIME	Outputs the sum ((s1)+(s2)) of the TIME data type input values.	Page 1345 ADD_TIME(_E)
ADD_TIME_E		

■Subtraction

Function symbol	Processing details	Reference
SUB_TIME	Outputs the difference ((s1)-(s2)) between the TIME data type input values.	Page 1347 SUB_TIME(_E)
SUB_TIME_E		

■Multiplication

Function symbol	Processing details	Reference
MUL_TIME	Outputs the product ((s1)×(s2)) of the TIME data type input values.	Page 1349 MUL_TIME(_E)
MUL_TIME_E		

■Division

Function symbol	Processing details	Reference
DIV_TIME	Outputs the quotient ((s1)÷(s2))of the TIME data type input values.	Page 1351 DIV_TIME(_E)
DIV_TIME_E		

8.2 Standard Function Blocks

Bistable function blocks

■Bistable function block (set-dominant)

Function block symbol	Processing details	Reference
SR	Discriminates between two input values, and outputs1 (TRUE) or 0 (FALSE).	Page 1354 SR(_E)
SR_E		

■Bistable function block (reset-dominant)

Function block symbol	Processing details	Reference
RS	Discriminates between two input values, and outputs1 (TRUE) or 0 (FALSE).	Page 1356 RS(_E)
RS_E		

Edge detection function blocks

■Detecting a rising edge

Function block symbol	Processing details	Reference
R_TRIG	Detects a signal rising edge, and outputs the pulse signal.	Page 1358 R_TRIG(_E)
R_TRIG_E		

■Detecting a falling edge

Function block symbol	Processing details	Reference
F_TRIG	Detects a signal falling edge, and outputs the pulse signal.	Page 1360 F_TRIG(_E)
F_TRIG_E		

Counter function blocks

■Up counter

Function block symbol	Processing details	Reference
CTU	Counts up the number of rising edges of a signal.	Page 1362 CTU(_E)
CTU_E		

■Down counter

Function block symbol	Processing details	Reference
CTD	Counts down the number of rising edges of a signal.	Page 1364 CTD(_E)
CTD_E		

■Up/down counter

Function block symbol	Processing details	Reference
CTUD	Counts up or down the number of rising edges of a signal.	Page 1366 CTUD(_E)
CTUD_E		

■Counter function block

Function block symbol	Processing details	Reference
COUNTER_FB_M	Starts counting up when the execution condition is satisfied.	Page 1369 COUNTER_FB_M

Timer function blocks

■Pulse timer

Function block symbol	Processing details	Reference
TP	Keeps the signal on for the specified period of time.	Page 1371 TP(_E)
TP_E		

■On delay timer

Function block symbol	Processing details	Reference
TON	Turns on a signal after the specified period of time.	Page 1374 TON(_E)
TON_E		

■Off delay timer

Function block symbol	Processing details	Reference
TOF	Turns off a signal after the specified period of time.	Page 1377 TOF(_E)
TOF_E		

■Timer function block

Function block symbol	Processing details	Reference
TIMER_10_FB_M	Starts counting a timer when the execution condition is satisfied, and continues counting until the timer reaches the set value.	Page 1379 TIMER_□_M
TIMER_100_FB_M		
TIMER_HIGH_FB_M		
TIMER_LOW_FB_M		
TIMER_CONT_FB_M		
TIMER_CONTHFB_M		

9 MOTION CONTROL FUNCTION BLOCK LIST

This chapter describes how to read the list.

Item	Description
Function block symbol	Shows the function block name.
Processing details	Shows an overview of the function block.
Reference	Shows the reference of detailed information.

For an overview on the motion control function block, refer to the following.

☞ Page 1384 MOTION CONTROL FB OVERVIEW

9.1 Administrative Function Blocks

The list of administrative motion control function blocks is shown below.

Axes Group Enabled

Function block symbol	Processing details	Reference
MC_GroupEnable	Changes the specified Axes group status from "0: Axes group disabled (GroupDisabled)" to "4: Standby (GroupStandby)".	☞ Page 1482 Axes Group Enabled

Axes Group Disabled

Function block symbol	Processing details	Reference
MC_GroupDisable	Changes the specified axes group status to "0: Axes group disabled (GroupDisabled)".	☞ Page 1486 Axes Group Disabled

Operation Available

Function block symbol	Processing details	Reference
MC_Power	Switches a specified axis to the operation possible status.	☞ Page 1490 Operable

Current Position Change

Function block symbol	Processing details	Reference
MC_SetPosition	Changes the current position (commanded position, actual position) of the specified axis.	☞ Page 1494 Current Position Change

Torque Limit Value

Function block symbol	Processing details	Reference
MCv_SetTorqueLimit	Changes a torque limit value.	☞ Page 1502 Torque Limit Value

Override Value Setting

Function block symbol	Processing details	Reference
MC_SetOverride	Changes the target velocity, the target acceleration, and the target deceleration of the specified axis.	☞ Page 1510 Override Value Setting

Parameter Read

Function block symbol	Processing details	Reference
MC_ReadParameter	Reads objects of a device.	☞ Page 1514 Parameter Read

Parameter Write

Function block symbol	Processing details	Reference
MC_WriteParameter	Writes objects of a device.	☞ Page 1521 Parameter Write

Axis Error Reset

Function block symbol	Processing details	Reference
MC_Reset	Resets errors and warnings of the axis.	☞ Page 1527 Axis Error Reset

Axes Group Error Reset

Function block symbol	Processing details	Reference
MC_GroupReset	Resets errors and warnings of the axes group and each axis in the axes group.	☞ Page 1530 Axes Group Error Reset

Touch Probe Enabled

Function block symbol	Processing details	Reference
MC_TouchProbe	Records optional data when a trigger event occurs.	☞ Page 1534 Touch Probe Enabled

Touch Probe Disabled

Function block symbol	Processing details	Reference
MC_AbortTrigger	Disables the latch that is being executed.	☞ Page 1550 Touch Probe Disabled

Cam Table Selection

Function block symbol	Processing details	Reference
MC_CamTableSelect	Stores the specified operation profile data (cam data) into the open area.	☞ Page 1555 Cam Table Selection

Current Value Change per Cycle

Function block symbol	Processing details	Reference
MCv_ChangeCycle	Changes the current value per cycle of the specified operation profile data control FB.	☞ Page 1562 Current Value Change per Cycle

All Axes Operation Available

Function block symbol	Processing details	Reference
MCv_AllPower	Switches all axes to the operation possible status.	☞ Page 1570 All Axes Operation Available

Axes Group Override Value Setting

Function block symbol	Processing details	Reference
MC_GroupSetOverride	Changes the target velocity, the target acceleration, and the target deceleration of the specified axes group.	☞ Page 1573 Axes Group Override Value Setting

Motion Error Reset

Function block symbol	Processing details	Reference
MCv_MotionErrorReset	Resets all errors and warnings of the motion system.	☞ Page 1577 Motion Error Reset

Control Operation Cycle Synchronization

Function block symbol	Processing details	Reference
MCv_SyncOperationCycles	Synchronizes the control operation cycle of the axis with the specified cycle.	☞ Page 1581 Control Operation Cycle Synchronization

Advanced Synchronous Control Current Position per Cycle

Function block symbol	Processing details	Reference
MCv_AdvPositionPerCycleCalc	Calculates the current position per cycle based on the specified cam data.	☞ Page 1584 Advanced Synchronous Control One Cycle Current Position Calculation

Advanced Synchronous Control Cam Set Position Calculation

Function block symbol	Processing details	Reference
MCv_AdvCamSetPositionCalc	Calculates the cam set position based on the specified cam data.	☞ Page 1587 Cam Set Position Calculation in Advanced Synchronous Control

Digital Cam Switch Output

Function block symbol	Processing details	Reference
MC_DigitalCamSwitch	Outputs on/off signal according to the value of data.	☞ Page 1590 Digital Cam Switch Output

Single Axis Status Read

Function block symbol	Processing details	Reference
MC_ReadStatus	Acquires the state transition status of the specified axis.	☞ Page 1598 Axis Status Read of Single Axis

9.2 Motion Function Blocks

The list of motion related motion control function blocks is shown below.

OPR

Function block symbol	Processing details	Reference
MC_Home	Executes homing for the specified axis.	☞ Page 1601 Homing

Forced Stop

Function block symbol	Processing details	Reference
MC_Stop	Decelerates the specified axis to a stop.	☞ Page 1611 Forced Stop

Group Forced Stop

Function block symbol	Processing details	Reference
MC_GroupStop	Decelerates the specified axes group to a stop.	☞ Page 1616 Group Forced Stop

Absolute Value Positioning

Function block symbol	Processing details	Reference
MC_MoveAbsolute	Sets the target position of the absolute position and performs positioning.	☞ Page 1622 Absolute Value Positioning

Relative Value Positioning

Function block symbol	Processing details	Reference
MC_MoveRelative	Sets the movement amount of the relative position and performs positioning.	☞ Page 1635 Relative Value Positioning

JOG

Function block symbol	Processing details	Reference
MCv_Jog	Executes JOG operation according to the command velocity.	☞ Page 1644 JOG Operation

Speed Control

Function block symbol	Processing details	Reference
MC_MoveVelocity	Switches the drive unit to csv and performs velocity control according to the specified velocity.	☞ Page 1653 Speed Control

Torque Control

Function block symbol	Processing details	Reference
MC_TorqueControl	Switches the drive unit to cst and performs torque control according to the specified target torque.	☞ Page 1661 Torque Control

Pressure Control

Function block symbol	Processing details	Reference
MCv_PressureControl	Switches control mode of the drive unit to cyclic synchronous pressure mode (cspr) and performs pressure control according to the specified pressure.	☞ Page 1672 Pressure Control

Speed Control (Including Position Loop)

Function block symbol	Processing details	Reference
MCv_SpeedControl	Performs speed control including the position loop.	☞ Page 1682 Speed Control (Including Position Loop)

Absolute Value Linear Interpolation Control

Function block symbol	Processing details	Reference
MCv_MoveLinearInterpolateAbsolute	Specifies the target position of the absolute position of the set axes group and performs positioning by linear interpolation control.	☞ Page 1690 Absolute Value Linear Interpolation Control

Relative Value Linear Interpolation Control

Function block symbol	Processing details	Reference
MCv_MoveLinearInterpolateRelative	Specifies the movement amount of the relative position of the set axes group and performs positioning by linear interpolation control.	☞ Page 1705 Relative Value Linear Interpolation Control

Absolute Value Circular Interpolation Control

Function block symbol	Processing details	Reference
MCv_MoveCircularInterpolateAbsolute	Performs positioning with 2-axis circular interpolation using the structuring axes of the set axes group by setting the end point and sub point of the absolute position.	☞ Page 1719 Absolute Value Circular Interpolation Control

Relative Value Circular Interpolation Control

Function block symbol	Processing details	Reference
MCv_MoveCircularInterpolateRelative	Performs positioning with 2-axis circular interpolation using the structuring axes of the set axes group by setting the relative position from the current position at start to the end point and sub point.	☞ Page 1736 Relative Value Circular Interpolation Control

Cam Operation Start

Function block symbol	Processing details	Reference
MC_CamIn	Starts cam operation according to the specified cam data.	☞ Page 1754 Cam Operation Start

Gear Operation Start

Function block symbol	Processing details	Reference
MC_GearIn	Starts gear operation according to the specified gear ratio.	☞ Page 1787 Gear Operation Start

Addition/Subtraction Positioning

Function block symbol	Processing details	Reference
MC_CombineAxes	Adds or subtracts the movement amounts of the two specified master axes and performs positioning using said value as the commanded position.	☞ Page 1797 Addition/ Subtraction Positioning

Backlash Compensation Filter

Function block symbol	Processing details	Reference
MCv_BacklashCompensation Filter	Performs filter processing that compensates the backlash amount in the machine system according to the movement direction.	☞ Page 1808 Backlash Compensation Filter

Smoothing Filter

Function block symbol	Processing details	Reference
MCv_SmoothingFilter	Performs filter processing according to the specified frequency.	☞ Page 1821 Smoothing Filter

Moving Direction Restriction Filter

Function block symbol	Processing details	Reference
MCv_DirectionFilter	Performs filter processing to restrict traveling for the set movement direction.	☞ Page 1831 Moving Direction Restriction Filter

Speed Limit Filter

Function block symbol	Processing details	Reference
MCv_SpeedLimitFilter	Performs filter processing to restrict to the velocity of the set limit value.	☞ Page 1840 Speed Limit Filter

Axis Control Start Wait

Function block symbol	Processing details	Reference
MCv_MoveWait	Immediately performs the operation of the multiple-started motion FB for the specified axis by inputting an execution permission signal.	☞ Page 1848 Axis Control Start Wait

Axes Group Control Start Wait

Function block symbol	Processing details	Reference
MCv_GroupMoveWait	Immediately performs the operation of the multiple-started motion FB for the specified axes group by inputting an execution permission signal.	☞ Page 1855 Axes Group Control Start Wait

Advanced Synchronous Control

Function block symbol	Processing details	Reference
MCv_AdvancedSync	Starts synchronization control according to the specified advanced synchronous control setting.	☞ Page 1862 Advanced Synchronous Control

Multiple Axes Positioning Data Operation

Function block symbol	Processing details	Reference
MCv_MovePositioningData	Operates an axes group according to the specified positioning data.	☞ Page 1868 Multiple Axes Positioning Data Operation

Absolute Position Tracking Control

Function block symbol	Processing details	Reference
MCv_PositionControl	Moves the specified axis to the target position according to the limit velocity, acceleration, deceleration, and jerk. Even if the instruction reaches the target position, the position can be changed by restarting or continuous updating and the axis can be moved.	☞ Page 1878 Absolute Position Tracking Control

Motion Cyclic Position Control

Function block symbol	Processing details	Reference
MCv_CyclicPosition	Sends the target position to the drive unit according to the specified target position.	☞ Page 1891 Motion Cyclic Position Control

Motion Cyclic Speed Control

Function block symbol	Processing details	Reference
MCv_CyclicVelocity	Sends the target speed to the drive unit according to the specified target speed.	☞ Page 1899 Motion Cyclic Velocity Control

Motion Cyclic Torque Control

Function block symbol	Processing details	Reference
MCv_CyclicTorque	Sends the target torque to the drive unit according to the specified target position.	☞ Page 1903 Motion Cyclic Torque Control

9.3 Standard Function Blocks

The list of motion control function blocks used to perform axis control is shown below.

Profile Read

Function block symbol	Processing details	Reference
MCv_ReadProfileData	Reads the specified operation profile data from the open area or a file.	☞ Page 1909 Profile Read

Profile Write

Function block symbol	Processing details	Reference
MCv_WriteProfileData	Writes the specified operation profile data to the open area or a file.	☞ Page 1922 Profile Write

10 NETWORK FUNCTION BLOCK LIST

How to read the list is shown below.

Item	Description
Function block symbol	Indicates a function block name.
Processing details	Indicates an overview of the function blocks.
Reference	Indicates the reference of detailed information.

10.1 Function Blocks for Ethernet

Opening/disconnecting a connection

Function block symbol	Processing details	Reference
M+MX-SQ_ConnectionOpen	Opens (establishes) a connection.	☞ Page 1946 M+MX-SQ_ConnectionOpen
M+MX-SQ_ConnectionClose	Closes (disconnects) the connection.	☞ Page 1950 M+MX-SQ_ConnectionClose

Function blocks for socket communications

Function block symbol	Processing details	Reference
M+MX-SQ_Recv_Socket	Reads the data received from the external device through socket communications.	☞ Page 1952 M+MX-SQ_Recv_Socket
M+MX-SQ_Send_Socket	Sends data to the external device through socket communications.	☞ Page 1955 M+MX-SQ_Send_Socket

Function blocks for SLMP frames

Function block symbol	Processing details	Reference
M+MX-SQ_SLMP_DeviceRead_IP	Reads device data from the SLMP-compatible device specified by an IP address.	☞ Page 1958 M+MX-SQ_SLMP_DeviceRead_IP
M+MX-SQ_SLMP_DeviceWrite_IP	Writes device data to the SLMP-compatible device specified by an IP address.	☞ Page 1965 M+MX-SQ_SLMP_DeviceWrite_IP
M+MX-TSN_SLMP_DeviceRead_IP	Reads device data from the SLMP-compatible device specified by an IP address.	☞ Page 1972 M+MX-TSN_SLMP_DeviceRead_IP
M+MX-TSN_SLMP_DeviceWrite_IP	Writes device data to the SLMP-compatible device specified by an IP address.	☞ Page 1979 M+MX-TSN_SLMP_DeviceWrite_IP

10.2 Function Blocks for CC-Link IE TSN

Remote reset of SLMP frames

Function block symbol	Processing details	Reference
M+MX-TSN_RemoteReset2_IP	Sends the remote RESET of an SLMP request to the target station specified by an IP address.	☞ Page 1986 M+MX-TSN_RemoteReset2_IP

MEMO

PART 3 SEQUENCE INSTRUCTIONS/BASIC INSTRUCTIONS

This part consists of the following chapters.

11 SEQUENCE INSTRUCTIONS

12 BASIC INSTRUCTIONS

11 SEQUENCE INSTRUCTIONS

11.1 Contact Instructions

Operation start, series connection, parallel connection

LD, LDI, AND, ANI, OR, ORI

• LD: Normally open contact operation start, LDI: Normally closed contact operation start

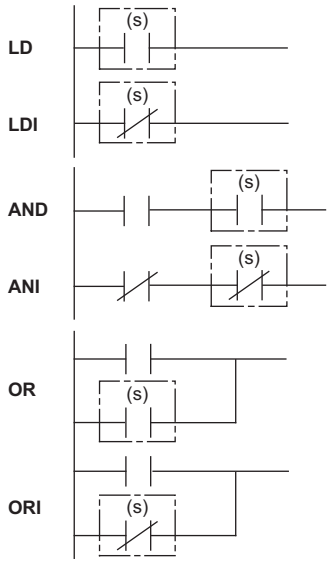
These instructions output the on/off information of the specified device as the operation result.

• AND: Normally open contact series connection, ANI: Normally closed contact series connection

These instructions perform an AND operation between the on/off information of the specified device and the previous operation result, and output the operation result.

• OR: Single normally open contact parallel connection, ORI: Single normally closed contact parallel connection

These instructions perform an OR operation between the on/off information of the specified device and the previous operation result, and output the operation result.

Ladder	ST
	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
LD LDI AND ANI OR ORI	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device used as a contact	—	Bit	ANY_BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DX)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	—	○	—	○	—	—	—	○

Processing details

■LD, LDI

- LD is a normally open contact operation start instruction and LDI is a normally closed contact operation start instruction. These instructions read the on/off information*1 of the specified bit device, and output it as the operation result.

*1 When a bit of a word device is specified, the instruction outputs on or off according to the status (1 or 0) of the specified bit.

■AND, ANI

- AND is a normally open contact series connection instruction and ANI is a normally closed contact series connection instruction. These instructions read the on/off information*1 of the specified bit device, perform an AND operation with the previous operation result, and output the operation result.

*1 When a bit of a word device is specified, the instruction outputs on or off according to the status (1 or 0) of the specified bit.

- Note the following when creating or displaying a program using the engineering tool (in ladder edit mode).
  - Write mode: When the AND and ANI instructions are connected in series, a ladder with up to 24 steps can be created.
  - Read mode: When the AND and ANI instructions are connected in series, a ladder with up to 24 steps can be displayed. If there are more than 24 steps, up to 24 steps are displayed.

■OR, ORI

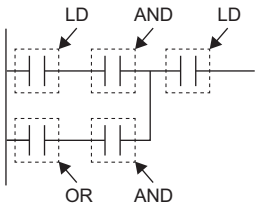
- OR is a single normally open contact parallel connection instruction and ORI is a single normally closed contact parallel connection instruction. These instructions read the on/off information*1 of the specified bit device, perform an OR operation with the previous operation result, and output the operation result.

*1 When a bit of a word device is specified, the instruction outputs on or off according to the status (1 or 0) of the specified bit.

- Note the following when creating or displaying a program using the engineering tool (in ladder edit mode).
  - Write mode: Up to 23 OR and ORI instructions can be connected consecutively.
  - Read mode: Up to 23 OR and ORI instructions connected consecutively can be displayed. A ladder with more than 23 instructions cannot be displayed correctly.

■Operation using LD, LDI, AND, ANI, OR, and ORI combined

An example of operation using LD, AND, and OR combined is shown below. The same operation is performed by using LDI, ANI, and ORI instead.



Point

- Specify a bit in a word device in hexadecimal. (For example, specify "D0.0B" for b11 in D0.)

Operation error

There is no operation error.

Pulse operation start, pulse series connection, pulse parallel connection

LDP, LDF, ANDP, ANDF, ORP, ORF

• LDP: Rising edge pulse operation start

This instruction turns on only at the rising edge (off to on) of the specified bit device.

• LDF: Falling edge pulse operation start

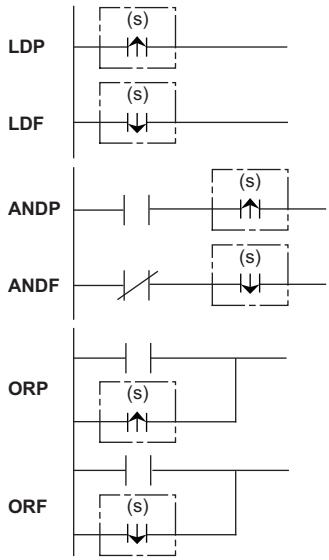
This instruction turns on only at the falling edge (on to off) of the specified bit device.

• ANDP: Rising edge pulse series connection, ANDF: Falling edge pulse series connection

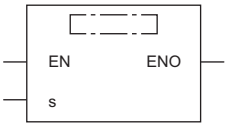
These instructions perform an AND operation with the previous operation result.

• ORP: Rising edge pulse parallel connection, ORF: Falling edge pulse parallel connection

These instructions perform an OR operation with the previous operation result.

Ladder	ST
	ENO:=LDP(EN,s); ENO:=LDF(EN,s); ENO:=ANDP(EN,s); ENO:=ANDF(EN,s); ENO:=ORP(EN,s); ENO:=ORF(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
LDP LDF ANDP ANDF ORP ORF	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device used as a contact	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

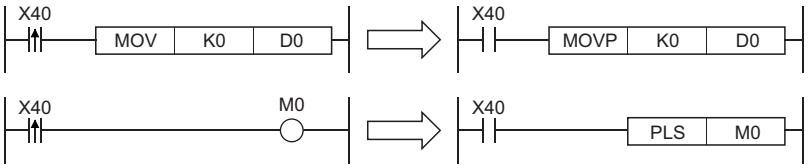
Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DX)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	—	○	—	○	—	—	—	○

Processing details

■LDP, LDF

- LDP is a rising edge pulse operation start instruction, and turns on only at the rising edge (off to on) of the specified bit device. When a bit-specified word device is used, this instruction turns on only when the specified bit changes from 0 to 1. In cases where there is an LDP instruction only, it acts identically to instructions for conversion to pulses that are executed during on (□P).

The following figure shows an example when a ladder using the LDP instruction is replaced with a ladder not using the LDP instruction.



- LDF is a falling edge pulse operation start instruction and turns on only at the falling edge (on to off) of the specified bit device. When a bit-specified word device is used, this instruction turns on only when the specified bit changes from 1 to 0.
- If the LDP instruction is used in the program written in ST language or FBD/LD, ENO turns on at the rising edge (off to on) of the specified bit device (s).
- If the LDF instruction is used in the program written in ST language or FBD/LD, ENO turns on at the falling edge (on to off) of the specified bit device (s).
- If the LDP or LDF instruction is used in the program written in ST language, set EN to be always on.
- If the LDP or LDF instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.

■ANDP, ANDF

- ANDP is a rising edge pulse series connection instruction and ANDF is a falling edge pulse series connection instruction. These instructions perform an AND operation with the previous operation result, and output the operation result. The following table lists the on/off information used by the ANDP and ANDF instructions.

Device specified by ANDP or ANDF		ANDP status	ANDF status
Bit device	Bit-specified word device		
Off→On	0→1	On	Off
Off	0	Off	Off
On	1	Off	Off
On→Off	1→0	Off	On

- If the ANDP instruction is used in the program written in ST or FBD/LD language, ENO turns on when the result of AND operation between EN and the rising edge of the specified bit device (s) is on. EN will not be an execution condition.
- If the ANDF instruction is used in the program written in ST or FBD/LD language, ENO turns on when the result of AND operation between EN and the falling edge of the specified bit device (s) is on. EN will not be an execution condition.

■ORP, ORF

- ORP is a rising edge pulse parallel connection instruction and ORF is a falling edge pulse parallel connection instruction. These instructions perform an OR operation with the previous operation result, and output the operation result. The following table lists the on/off information used by the ORP and ORF instructions.

Device specified by ORP or ORF		ORP status	ORF status
Bit device	Bit-specified word device		
Off→On	0→1	On	Off
Off	0	Off	Off
On	1	Off	Off
On→Off	1→0	Off	On

- If the ORP instruction is used in the program written in ST or FBD/LD language, ENO turns when the result of OR operation between EN and the rising edge of the specified bit device (s) is on. EN will not be an execution condition.
- If the ORF instruction is used in the program written in ST or FBD/LD language, ENO turns when the result of OR operation between EN and the falling edge of the specified bit device (s) is on. EN will not be an execution condition.

■Operation using LDP, LDF, ANDP, ANDF, ORP, and ORF combined

An example of operation using LDP, LDF, ANDP, ANDF, ORP, and ORF combined is same as that using LD, AND, and OR. (☞ Page 202 LD, LDI, AND, ANI, OR, ORI)

Operation error

There is no operation error.

Pulse NOT operation start, pulse NOT series connection, pulse NOT parallel connection

LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI

• LDPI: Rising edge pulse NOT operation start

This instruction turns on when the specified device is off, on, or at the falling edge (on to off).

• LDFI: Falling edge pulse NOT operation start

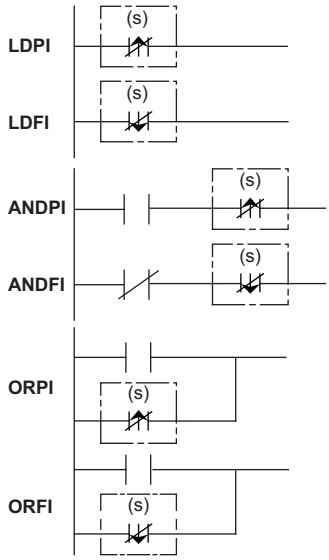
This instruction turns on when the specified bit device is at the rising edge (off to on), off, or on.

• ANDPI: Rising edge pulse NOT series connection, ANDFI: Falling edge pulse NOT series connection

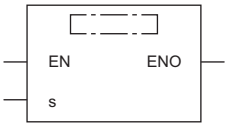
These instructions perform an AND operation with the previous operation result.

• ORPI: Rising edge pulse NOT parallel connection, ORFI: Falling edge pulse NOT parallel connection

These instructions perform an OR operation with the previous operation result.

Ladder	ST
	ENO:=LDPI(EN,s); ENO:=LDFI(EN,s); ENO:=ANDPI(EN,s); ENO:=ANDFI(EN,s); ENO:=ORPI(EN,s); ENO:=ORFI(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
LDPI LDFI ANDPI ANDFI ORPI ORFI	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device used as a contact	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DX)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	—	○	—	○	—	—	—	○

Processing details

■LDPI, LDFI

- LDPI is a rising edge pulse NOT operation start instruction, and turns on when the specified device is off, on, or at the falling edge (on to off). When a bit-specified word device is used, this instruction turns on when the specified bit is 0 or 1 or when the bit changes from 1 to 0.
- LDFI is a falling edge pulse NOT operation start instruction, and turns on when the specified bit device is at the rising edge (off to on), off, or on. When a bit-specified word device is used, this instruction turns on when the specified bit is 0 or 1 or when the bit changes from 0 to 1. The following table lists the on/off information used by the LDPI and LDFI instructions.

Device specified by LDPI or LDFI		LDPI status	LDFI status
Bit device	Bit-specified word device		
Off→On	0→1	Off	On
Off	0	On	On
On	1	On	On
On→Off	1→0	On	Off

- If the LDPI instruction is used in the program written in ST language or FBD/LD, ENO turns on at the timing except the rising edge (off to on) of the specified bit device (s).
- If the LDFI instruction is used in the program written in ST language or FBD/LD, ENO turns on at the timing except the falling edge (on to off) of the specified bit device (s).
- If the LDPI or LDFI instruction is used in the program written in ST language, set EN to be always on.
- If the LDPI or LDFI instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.

■ANDPI, ANDFI

- ANDPI is a rising edge pulse NOT series connection instruction and ANDFI is a falling edge pulse NOT series connection instruction. These instructions perform an AND operation with the previous operation result, and output the operation result. The following table lists the on/off information used by the ANDPI and ANDFI instructions.

Device specified by ANDPI or ANDFI		ANDPI status	ANDFI status
Bit device	Bit-specified word device		
Off→On	0→1	Off	On
Off	0	On	On
On	1	On	On
On→Off	1→0	On	Off

- If the ANDPI instruction is used in the program written in ST or FBD/LD language, ENO turns on when the result of AND operation between EN and the rising edge of the specified bit device (s) is not on. EN will not be an execution condition.
- If the ANDFI instruction is used in the program written in ST or FBD/LD language, ENO turns on when the result of AND operation between EN and the falling edge of the specified bit device (s) is not on. EN will not be an execution condition.

■ORPI, ORFI

- ORPI is a rising edge pulse NOT parallel connection instruction and ORFI is a falling edge pulse NOT parallel connection instruction. These instructions perform an OR operation with the previous operation result, and output the operation result. The following table lists the on/off information used by the ORPI and ORFI instructions.

Device specified by ORPI or ORFI		ORPI status	ORFI status
Bit device	Bit-specified word device		
Off→On	0→1	Off	On
Off	0	On	On
On	1	On	On
On→Off	1→0	On	Off

- If the ORPI instruction is used in the program written in ST or FBD/LD language, ENO turns when the result of OR operation between EN and the rising edge of the specified bit device (s) is not on. EN will not be an execution condition.
- If the ORFI instruction is used in the program written in ST or FBD/LD language, ENO turns when the result of OR operation between EN and the falling edge of the specified bit device (s) is not on. EN will not be an execution condition.

■Operation using LDPI, LDFI, ANDPI, ANDFI, ORPI, and ORFI combined

An example of operation using LDPI, LDFI, ANDPI, ANDFI, ORPI, and ORFI combined is same as that using LD, AND, and OR. (☞ Page 202 LD, LDI, AND, ANI, OR, ORI)

Operation error

There is no operation error.

11.2 Association Instructions

Ladder block series/parallel connection

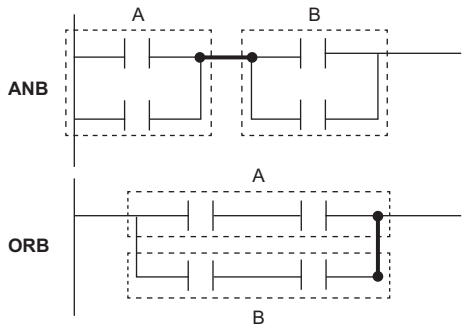
ANB, ORB

• ANB: Ladder block series connection

This instruction performs an AND operation between block A and block B.

• ORB: Ladder block parallel connection

This instruction performs an OR operation between block A and block B.

Ladder	ST
 A: A block B: B block	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
ANB ORB	Every scan

Processing details

■ANB

• This instruction performs an AND operation between block A and block B, and outputs the operation result.

• The symbol of the ANB instruction is not a contact but a connection.

■ORB

• This instruction performs an OR operation between block A and block B, and outputs the operation result.

• Ladder blocks, each having two or more contacts, are connected in parallel. Use the OR or ORI instruction for connection of blocks, each having only one contact. The ORB instruction is not required in this case.

• The symbol of the ORB instruction is not a contact but a connection.

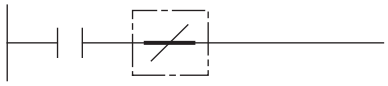
Operation error

There is no operation error.

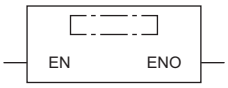
Inverting the operation result

INV

This instruction inverts the operation result up to just before the INV instruction.

Ladder	ST
	ENO:=INV(EN);

FBD/LD



■Execution condition

Instruction	Execution condition
INV	Every scan

Processing details

- This instruction inverts the operation result up to just before the INV instruction.

Operation result up to just before the INV instruction	Operation result after execution of the INV instruction
Off	On
On	Off

Operation error

There is no operation error.

Point

- The INV instruction operates based on the results of calculation made until the INV instruction is given. Accordingly, use it in the same position as that of the AND (☞ Page 202 Operation start, series connection, parallel connection). The INV instruction cannot be used at the LD and OR (☞ Page 202 Operation start, series connection, parallel connection) positions.
- When a ladder block is used, the operation result is inverted within the range of the ladder block. When the INV instruction and the ANB instruction are used together in the same ladder, pay attention to the inversion range.



Broken line part: Inversion range

For details on the ANB instruction, refer to the following.

☞ Page 210 ANB, ORB

Converting the operation result into a pulse

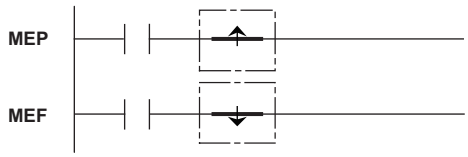
MEP, MEF

• MEP: Converting the operation result into a pulse (rising edge)

This instruction turns on at the rising edge (off to on) of the operation result up to the MEP instruction.

• MEF: Converting the operation result into a pulse (falling edge)

This instruction turns on at the falling edge (on to off) of the operation result up to the MEF instruction.

Ladder	ST
	ENO:=MEP(EN); ENO:=MEF(EN);

FBD/LD



■Execution condition

Instruction	Execution condition
MEP MEF	Every scan

Processing details

■MEP

• This instruction turns on (continuity state) at the rising edge (off to on) of the operation result up to the MEP instruction. The instruction turns off (non-continuity state) when the operation result is in another state (not rising edge).

• Use of the MEP instruction eases pulse conversion processing when multiple contacts are connected in series.

■MEF

• This instruction turns on (continuity state) at the falling edge (on to off) of the operation result up to the MEF instruction. The instruction turns off (non-continuity state) when the operation result is in another state (not falling edge).

• Use of the MEF instruction eases pulse conversion processing when multiple contacts are connected in series.

Operation error

There is no operation error.

Point

• The MEP or MEF instruction may not operate correctly if pulse conversion is performed for an index-modified contact in the subroutine program or in the area between the FOR and NEXT instructions. To perform pulse conversion for an index-modified contact in the subroutine program or in the area between the FOR and NEXT instructions, refer to the following.

☞ Page 213 EGP, EGF

• The MEP or MEF instruction operates based on the result of operation performed from the LD instruction immediately before the MEP or MEF instruction until the MEP or MEF instruction is given. Therefore, use them at the same position as that of the AND (☞ Page 202 Operation start, series connection, parallel connection). The MEP or MEF instruction cannot be used at the LD and OR (☞ Page 202 Operation start, series connection, parallel connection) positions.

Converting the edge relay operation result into a pulse

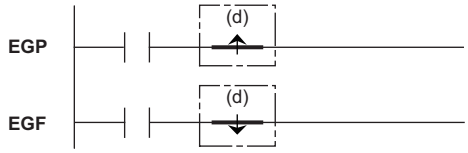
EGP, EGF

• EGP: Converting the edge relay operation result into a pulse (rising edge)

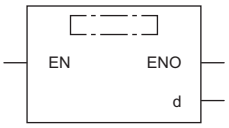
This instruction stores the operation result up to the EGP instruction in the edge relay (V). The instruction turns on at the rising edge (off to on) of the operation result.

• EGF: Converting the edge relay operation result into a pulse (falling edge)

This instruction stores the operation result up to the EGF instruction in the edge relay (V). The instruction turns on at the falling edge (on to off) of the operation result.

Ladder	ST
	ENO:=EGP(EN,d); ENO:=EGF(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EGP	
EGF	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Edge relay number for storing operation result	—	Bit	ANY_BOOL*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only bit type labels assigned to the device (V) can be used.

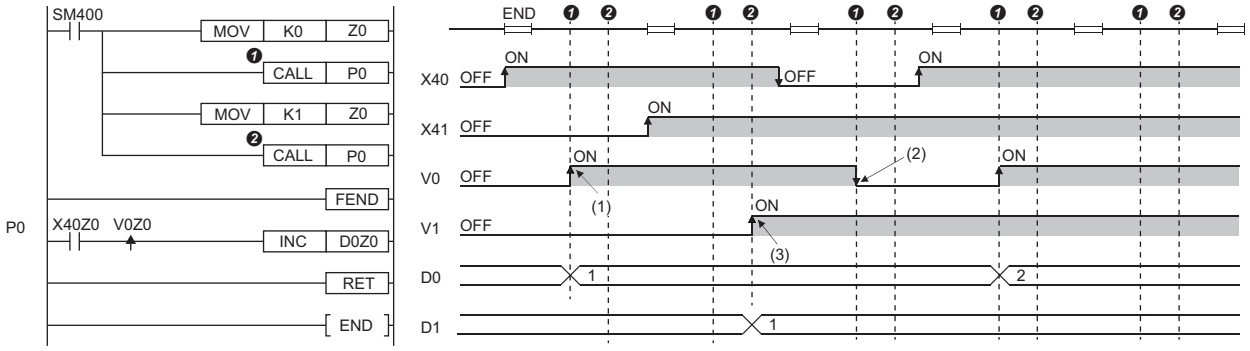
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (V)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	—	—	—	—	—	—	—	—	—	○

Processing details

■EGP

- This instruction stores the operation result up to the EGP instruction in the edge relay (V).
- The instruction turns on (continuity state) at the rising edge (off to on) of the operation result up to the EGP instruction. The instruction turns off (non-continuity state) when the operation result is in another state (staying on, falling edge (on to off), or staying off).
- The instruction is used to perform pulse conversion for index-modified programs in the subroutine program or in the area between the FOR and NEXT instructions.
- The instruction can be used in the same way as the AND instruction.
- The following figure shows the operation performed when the instruction is used in the subroutine program.



(1) The device turns on at the rising edge of X40.
(2) The device turns off at the falling edge of X40.
(3) The device turns on at the rising edge of X41.

■EGF

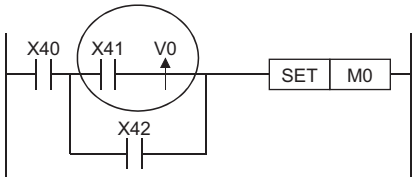
- This instruction stores the operation result up to the EGF instruction in the edge relay (V).
- The instruction turns on (continuity state) at the falling edge (on to off) of the operation result up to the EGF instruction. The instruction turns off (non-continuity state) when the operation result is in another state (staying on, rising edge (off to on), or staying off).
- The instruction is used to perform pulse conversion for index-modified programs in the subroutine program or in the area between the FOR and NEXT instructions.
- The instruction can be used in the same way as the AND instruction.

Operation error

There is no operation error.

Point

- The EGP or EGF instruction operates based on the result of operation performed from the LD instruction immediately before the EGP or EGF instruction until the EGP or EGF instruction is given. Therefore, use them at the same position as that of the AND (☞ Page 202 Operation start, series connection, parallel connection). The EGP or EGF instruction cannot be used at the LD and OR (☞ Page 202 Operation start, series connection, parallel connection) positions.
- The instructions cannot be used at the ladder block position shown below.

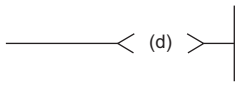


11.3 Output Instructions

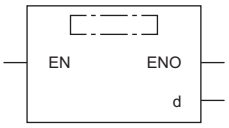
Out (excluding the timer, counter, and annunciator)

OUT

This instruction outputs the operation result to the specified device.

Ladder	ST
	ENO:=OUT(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
OUT	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	On/off target device number	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DY)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	○	○*2	○	—	—	—	○	—	—	—	○

*1 When F is used, refer to the following.
☞ Page 227 OUT F

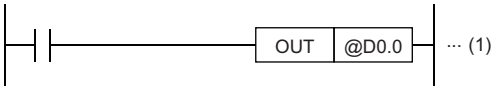
*2 When T or ST is used, refer to the following.
☞ Page 217 OUT T, OUTH T, OUT ST, OUTH ST
When C is used, refer to the following.
☞ Page 223 OUT C

Processing details

- This instruction outputs the operation result up to the OUT instruction to the specified device.

Condition	Operation result	Coil/Specified bit
When a bit device is used	Off	Off
	On	On
When a bit-specified word device is used	Off	0
	On	1

- When indirect specification is used, specify the bit as shown below.



(1) The operation result is output to bit 0 of the indirect address stored in D0.

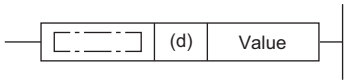
Operation error

There is no operation error.

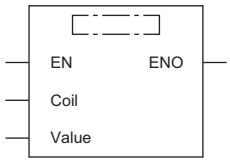
OUT T, OUTH T, OUT ST, OUTH ST

- OUT T: Low-speed timer instruction
- OUTH T: High-speed timer instruction
- OUT ST: Low-speed retentive timer instruction
- OUTH ST: High-speed retentive timer instruction

These instructions start time measurement when the operation result up to the OUT instruction is on. When time is up, the normally open contact turns on (continuity state) and the normally closed contact turns off (non-continuity state).

Ladder	ST
 Value: Set value	ENO:=OUT_T(EN,Coil,Value); ENO:=OUTH(EN,Coil,Value);

FBD/LD



Value: Set value
(☐ is to be replaced by either of the following: OUT_T, OUTH.)

■Execution condition

Instruction	Execution condition
OUT T OUTH T OUT ST OUTH ST	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Timer device or timer type label	—	Bit	ANY_BOOL
Coil				
Value	Value set for the timer	0 to 65535	16-bit unsigned binary*1	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 If the program is written in ST language or FBD/LD, the data type will be ANY_INT.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DY)
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○*1	—	—	—	—	—	—	—	—	—
Coil												
Value	—	—	○*2	○	—	—	—	—	○*3	—	—	—

*1 Only T and ST can be used.
*2 T, ST, and C cannot be used.
*3 Only K (decimal constant) can be used.

Processing details

- These instructions start time measurement, triggered by the coil specified by (d) (in ST language or FBD/LD, displayed as Coil), when the operation result up to the OUT instruction is on. When time is up (current value ≥ set value), the normally open contact turns on (continuity state) and the normally closed contact turns off (non-continuity state).
- When the operation result up to the OUT instruction turns off, the contact responds as shown below.

Type	Timer coil	Current value	Before time is up		After time is up	
			Normally open contact	Normally closed contact	Normally open contact	Normally closed contact
Low-speed timer	Off	0	Non-continuity	Continuity	Non-continuity	Continuity
High-speed timer						
Low-speed retentive timer	Off	Current value retained	Non-continuity	Continuity	Continuity	Non-continuity
High-speed retentive timer						

- To clear the current value of the retentive timer and turn off the contact after time is up, use the RST instruction.
- When the timer set value is 0, the time will be up at execution of the OUT instruction.
- The following operations are performed at execution of the OUT instruction.
  - The coil used as a trigger of the OUT T, OUTH T, OUT ST, or OUTH ST instruction turns on or off.
  - The contact used as a trigger of the OUT T, OUTH T, OUT ST, or OUTH ST instruction turns on or off.
  - The current value of the OUT T, OUTH T, OUT ST, or OUTH ST instruction is changed.
- If the OUT T instruction is skipped by using such as the JMP instruction while the OUT T, OUTH T, OUT ST, or OUTH ST instruction is on, the current value is not updated or the contact is not turned on or off.
- If the same OUT T, OUTH T, OUT ST, or OUTH ST instruction is executed two times or more in a single scan, the current value is updated by the number of times the instruction is executed.

Point

The timer limit value is set in parameter using the engineering tool.

- Low-speed timer/low-speed retentive timer: 1 to 10000ms (in increments of 1ms) (Default: 100ms)
- High-speed timer/high-speed retentive timer: 0.01 to 100.0ms (in increments of 0.01ms) (Default: 10.0ms)

For the counting method, refer to the following.

User's manual for the controller used

Precautions

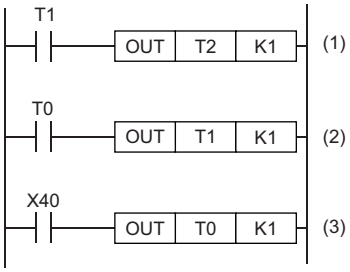
To create a program in which the operation of a timer contact triggers the operation of another timer, program the timers in order from the one that operates last.

When the set value of the timer is shorter than the scan time, if a program is created in order of timer measurements, all timers turn on in the same scan.

- The set value is smaller than the scan time.
- The set value is 1.

Ex.

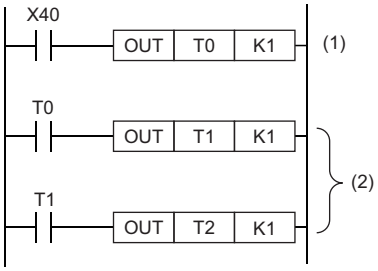
When timers T0 to T2 are programmed in order from the one that measures last



(3) Timer T0 starts measurement when X40 turns on.
(2) Timer T1 starts measurement from the next scan after the contact of timer T0 turns on.
(1) Timer T2 starts measurement from the next scan after the contact of timer T1 turns on.

Ex.

When timers T0 to T2 are programmed in order of measurement



(1) Timer T0 starts measurement when X40 turns on.
(2) When the contact of timer T0 turns on, the contacts of timers T1 and T2 also turn on.

Operation error

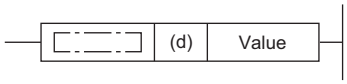
There is no operation error.

Long timer

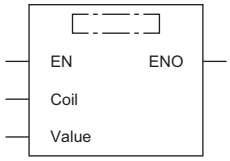
OUT LT, OUT LST

- OUT LT: Low-speed long timer instruction
- OUT LST: Low-speed long retentive timer instruction

These instructions start time measurement when the operation result up to the OUT instruction is on. When time is up, the normally open contact turns on (continuity state) and the normally closed contact turns off (non-continuity state).

Ladder	ST
 Value: Set value	ENO:=OUT_T(EN,Coil,Value);

FBD/LD



Value: Set value
(☐ is to be replaced by OUT_T.)

■Execution condition

Instruction	Execution condition
OUT LT OUT LST	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Long timer device or long timer type label	—	Bit	ANY_BOOL
Coil				
Value	Value set for the long timer	0 to 4294967295	32-bit unsigned binary*1	ANY32*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 If the program is written in ST language or FBD/LD, the data type will be ANY_INT.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	—	—	—	○*1	—	—	—	—	—	—
Coil												
Value	—	—	○*2	○	—	—	—	—	○*3	—	—	—

*1 Only LT and LST can be used.
*2 T, ST, and C cannot be used.
*3 Only K (decimal constant) can be used.

Processing details

- These instructions start time measurement, triggered by the coil specified by (d) (in SD language or FBD/LD, displayed as Coil), when the operation result up to the OUT instruction is on. When time is up (current value ≥ set value), the normally open contact turns on (continuity state) and the normally closed contact turns off (non-continuity state).
- When the operation result up to the OUT instruction turns off, the contact responds as shown below.

Type	Timer coil	Current value	Before time is up		After time is up	
			Normally open contact	Normally closed contact	Normally open contact	Normally closed contact
Long timer	Off	0	Non-continuity	Continuity	Non-continuity	Continuity
Long retentive timer	Off	Current value retained	Non-continuity	Continuity	Continuity	Non-continuity

- To clear the current value of the long retentive timer and turn off the contact after time is up, use the RST instruction.
- When the timer set value is 0, the time will be up at execution of the OUT instruction.
- The following operations are performed at execution of the OUT instruction.
  - The coil used as a trigger of the OUT LT or OUT LST instruction turns on or off.
  - The contact used as a trigger of the OUT LT or OUT LST instruction turns on or off.
  - The current value of the OUT LT or OUT LST instruction is changed.
- If the OUT LT instruction is skipped by using such as the JMP instruction while the OUT LT or OUT LST instruction is on, the current value is not updated or the contact is not turned on or off.
- If the same OUT LT or OUT LST instruction is executed two times or more in a single scan, the current value is updated by the number of times the instruction is executed.

Point

The timer limit value is set in parameter using the engineering tool.

- Long timer/long retentive timer: 0.001 to 1000ms (in increments of 0.001ms) (Default: 0.001ms)

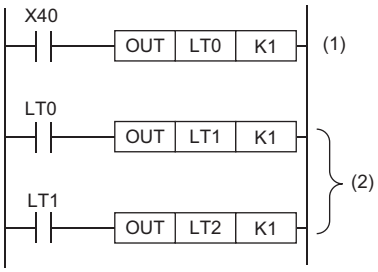
For the counting method, refer to the following.

User's manual for the controller used

Precautions

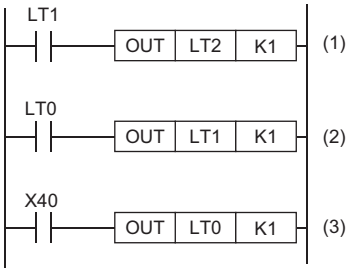
To create a program in which the operation of a long timer contact triggers the operation of another long timer, program the long timers in order from the one that operates first.
When long timers are programmed in order from the one that operates last, the timer measurement error may increase.
When the setting value of the long timer is shorter than the scan time, the operations are as follows.

Ex.
When long timers LT0 to LT2 are programmed in order of measurement



(1) Timer LT0 starts measurement when X40 turns on.
(2) When the contact of timer LT0 turns on, timer LT1 and then timer LT2 start measurement.

Ex.
When timers LT0 to LT2 are programmed in order from the one that measures last



(3) Timer LT0 starts measurement when X40 turns on.
(2) Long timer LT1 starts measurement from the next scan after the contact of long timer LT0 turns on.
(1) Long timer LT2 starts measurement from the next scan after the contact of long timer LT1 turns on.

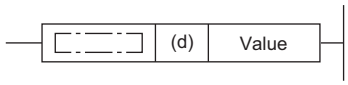
Operation error

There is no operation error.

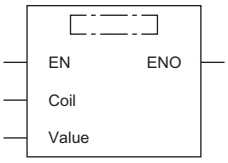
Counter

OUT C

This instruction increments the current counter value (count value) by one when the operation result up to the OUT instruction turns on. When the count value reaches the set value, the normally open contact of the counter turns on (continuity state) and the normally closed contact turns off (non-continuity state).

Ladder	ST
 Value: Set value	ENO:=OUT_C(EN,Coil,Value);

FBD/LD



Value: Set value

■Execution condition

Instruction	Execution condition
OUT C	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Counter number	—	Bit	ANY_BOOL*1
Coil				
Value	Value set for the counter	0 to 65535	16-bit unsigned binary*2	ANY16*2
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only counter type labels can be used.
*2 If the program is written in ST language or FBD/LD, the data type will be ANY_INT.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○*1	—	—	—	—	—	—	—	—	—
Coil												
Value	—	—	○*2	○	—	—	—	—	○*3	—	—	—

*1 Only C can be used.
*2 T, ST, and C cannot be used.
*3 Only K (decimal constant) can be used.

Processing details

- This instruction increments the current counter value (count value) in the device specified by (d) (in ST language or FBD/LD, displayed as Coil) by one on the rising edge (off to on) of the operation result up to the OUT instruction. When the count value reaches the set value (current value ≥ set value), the normally open contact turns on (continuity state) and the normally closed contact turns off (non-continuity state).
- When the set value is 0, the same processing is performed as when it is set to the above.
- Counting is disabled while the operation result remains on. (Count input does not need to be converted into pulses.)
- After counting-up, the count value and contact status remain unchanged until the RST instruction is executed.

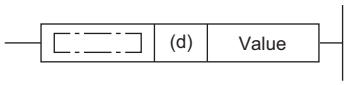
Operation error

There is no operation error.

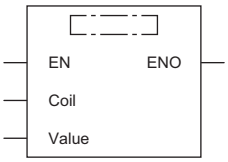
Long counter

OUT LC

This instruction increments the current long counter value (count value) by one on the rising edge (off to on) of the operation result up to the OUT instruction. When the count value reaches the set value, the normally open contact of the long counter turns on (continuity state) and the normally closed contact turns off (non-continuity state).

Ladder	ST
 Value: Set value	ENO:=OUT_C(EN,Coil,Value);

FBD/LD



Value: Set value
(☐ is to be replaced by OUT_C.)

■Execution condition

Instruction	Execution condition
OUT LC	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Long counter number	—	Bit	ANY_BOOL*1
Coil				
Value	Set value for the long counter	0 to 4294967295	32-bit unsigned binary*2	ANY32*2
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only long counter type labels can be used.
*2 If the program is written in ST language or FBD/LD, the data type will be ANY_INT.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	—	—	—	○*1	—	—	—	—	—	—
Coil												
Value	—	—	○*2	○	—	—	—	—	○*3	—	—	—

*1 Only LC can be used.
*2 T, ST, and C cannot be used.
*3 Only K (decimal constant) can be used.

Processing details

- This instruction increments the current long counter value (count value) in the device specified by (d) (in ST language or FBD/LD, displayed as Coil) by one on the rising edge (off to on) of the operation result up to the OUT instruction. When the count value reaches the set value (current value ≥ set value), the normally open contact turns on (continuity state) and the normally closed contact turns off (non-continuity state).
- When the set value is 0, the same processing is performed as when it is set to the above.
- Counting is disabled while the operation result remains on. (Count input does not need to be converted into pulses.)
- After counting-up, the count value and contact status remain unchanged until the RST instruction is executed.

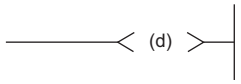
Operation error

There is no operation error.

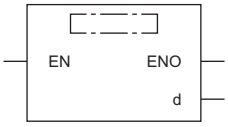
Annunciator

OUT F

This instruction outputs the operation result up to the OUT F instruction to the specified annunciator.

Ladder	ST
	ENO:=OUT(EN,d);

FBD/LD



(☐ is to be replaced by OUT.)

■Execution condition

Instruction	Execution condition
OUT F	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Target annunciator number	—	Bit	ANY_BOOL*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to the annunciator can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	—	—	—	—	—	—	—	—	—	—	—

*1 Only F can be used.

Processing details

- This instruction outputs the operation result up to the OUT F instruction to the specified annunciator.
- When the annunciator (F) is turned on by this instruction, the following are performed.
  - The USER LED of the controller turns on.
  - The annunciator number (F number) to be turned on is stored in the special register (SD64 to SD79).
  - The value in SD63 is incremented by one.
- If the value in SD63 is 16 (meaning 16 annunciators are already on), the annunciator number will not be stored in the special register (SD64 to SD79) even when a new annunciator turns on.
- When the annunciator (F) is turned off by this instruction, the following are performed.
  - The coil turns off, but the USER LED status and the data in SD63 to SD79 remain unchanged.
  - To turn off the USER LED or delete the annunciator number that has been turned off by this instruction from SD63 to SD79, use the RST F instruction.

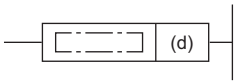
Operation error

There is no operation error.

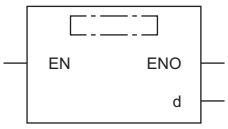
Setting devices (excluding annunciator)

SET

This instruction turns on the specified bit.

Ladder	ST
	ENO:=SET(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SET	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Set target bit device number or bit specification of word device	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DY)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	○	—	—	—	—	—	○	—	—	—	○

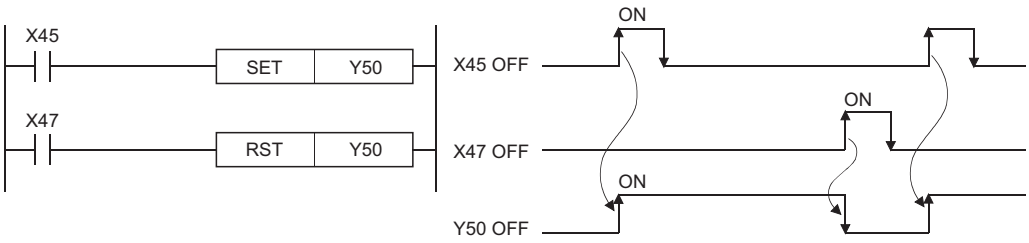
*1 When F is used, refer to the following.
☞ Page 232 SET F

Processing details

- This instruction changes the device status as follows when the execution command turns on.

Device	Status
Bit device	Turns on the coil or contact.
Bit-specified word device	Sets the specified bit to 1.

- The device that has been turned on remains on even after the execution command turns off. The device that has been turned on can be turned off by using the RST instruction.



- When the execution command is off, the device status does not change.

Operation error

There is no operation error.

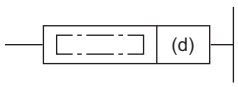
Point

When X is used, specify a device number that is not used in actual input. If the number that is used in actual input is specified, the data of actual input is written over the input device (X) specified by the SET instruction.

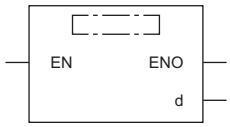
Resetting devices (excluding annunciator)

RST

This instruction turns off the specified device. For the timer and counter, the instruction clears the current value to 0 and turns off the contact or coil.

Ladder	ST
	ENO:=RST(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
RST	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Reset target bit device number, bit specification of word device, or reset target word device number	—	Bit/Word/Double word	ANY_ELEMENTARY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	(DY)
(d)	○*1	○	○	○	○	○	○	○	—	—	—	○

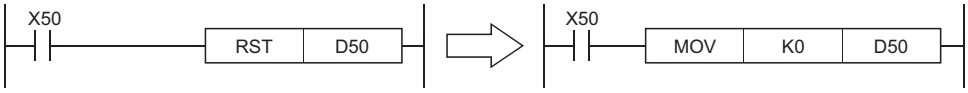
*1 When F is used, refer to the following.
☞ Page 234 RST F

Processing details

- This instruction changes the device status as follows when the execution command turns on.

Device	Status
Bit device	Turns off the coil or contact.
Timer, counter	Clears the current value to 0 and turns off the coil or contact.
Bit-specified word device	Sets the specified bit to 0.
Word device other than timer and counter	Clears the data to 0.

- When the execution command is off, the device status does not change.
- The RST instruction specifying a word device operates in the same way as the following ladder.



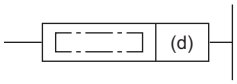
Operation error

There is no operation error.

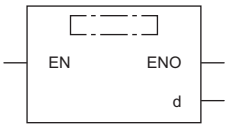
Setting annunciator

SET F

This instruction turns on the specified annunciator.

Ladder	ST
	ENO:=SET(EN,d);

FBD/LD



(☐ is to be replaced by SET.)

■Execution condition

Instruction	Execution condition
SET F	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Set target annunciator number (F number)	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

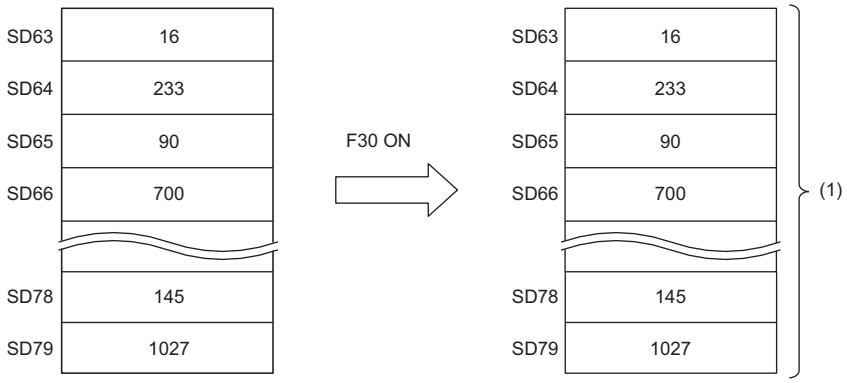
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	—	—	—	—	—	—	—	—	—	—	—

*1 Only F can be used.

Processing details

- This instruction turns on the annunciator specified by (d) when the execution command turns on.
- When the annunciator (F) is turned on, the following are performed.
  - The USER LED turns on.
  - The annunciator number (F number) turned on is stored in the special register (SD64 to SD79).
  - The value in SD63 is incremented by one.
- If the value in SD63 is 16 (meaning 16 annunciators are already on), the annunciator number will not be stored in the special register (SD64 to SD79) even when a new annunciator turns on.



(1) The data remain the same.

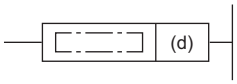
Operation error

There is no operation error.

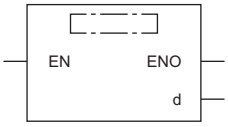
Resetting annunciator

RST F

This instruction turns off the specified annunciator.

Ladder	ST
	ENO:=RST(EN,d);

FBD/LD



(☐ is to be replaced by RST.)

■Execution condition

Instruction	Execution condition
RST F	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Reset target annunciator number (F number)	—	Bit*1	ANY_BOOL*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 If the program is written in ST language or FBD/LD, the data type will be ANY_ELEMENTARY.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	—	—	—	—	—	—	—	—	—	—	—

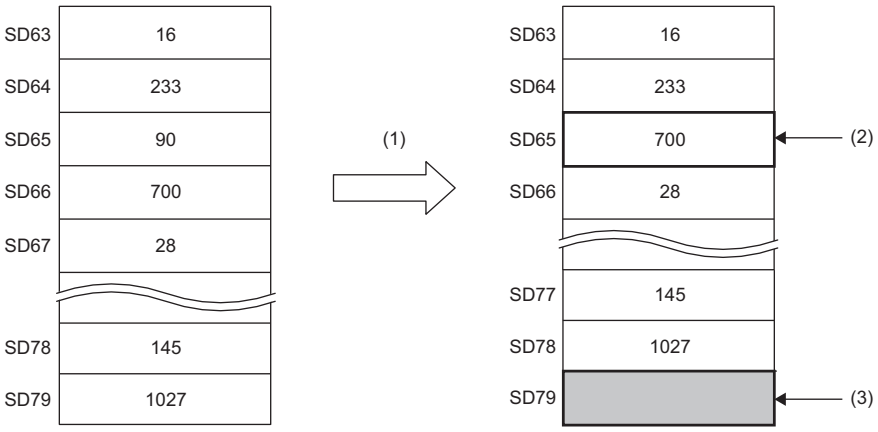
*1 Only F can be used.

Processing details

- This instruction turns off the annunciator specified by (d) when the execution command turns on.
- The annunciator number (F number) turned off is deleted from the special register (SD64 to SD79), and the value in SD63 is decremented by one.
- If the value in SD63 is 16, the corresponding annunciator number is deleted from SD64 to SD79 by the RST instruction. If an annunciator with a number not registered in SD64 to SD79 has been turned on, the number is newly registered. If all annunciator numbers in SD64 to SD79 are reset (turned off), the USER LED of the controller turns off.

Ex.

When the value in SD63 is 16 and there is an annunciator number that is not registered



(1) Reset F90.

(2) The F number in SD66 is shifted to this area.

(3) New F number is stored.

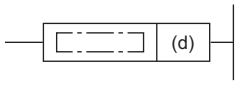
Operation error

There is no operation error.

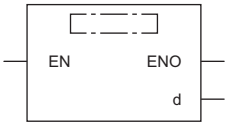
Rising edge output

PLS

This instruction turns on the specified device for one scan on the rising edge (off to on) of the execution command.

Ladder	ST
	ENO:=PLS(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
PLS	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Pulse conversion target device number	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

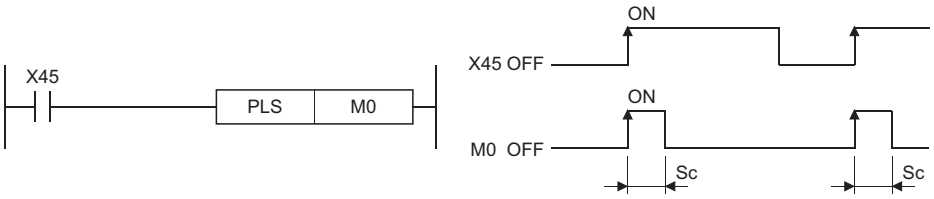
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DY)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	—	—	—	○	—	—	—	○

Processing details

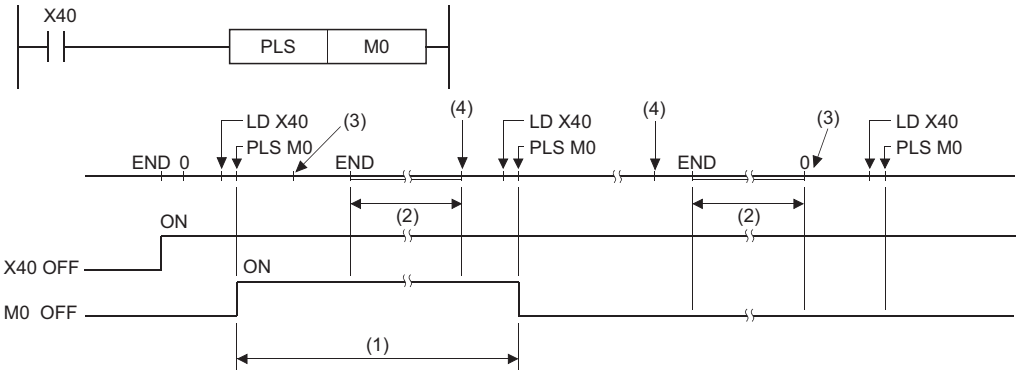
- This instruction turns on the specified device on the rising edge (off to on) of the execution command. When the execution command is in another state (staying on, falling edge (on to off), or staying off), the instruction turns off the specified device. If only one PLS instruction in the device specified by (d) is executed in a single scan, the specified device turns on for one scan. For the operation to be performed if more than one PLS instruction is executed during one scan, refer to the following.

☞ Page 52 Operations arising when the OUT, SET/RST, and PLS/PLF instructions of the same device are used



Sc: 1 scan

- Once after execution of the PLS instruction, even if the switch of the controller is moved to the STOP position and then the RUN position again, the PLS instruction is not executed.



(1) M0 turns on for one scan.
(2) Controller operation stop time
(3) Change the RUN/STOP/RESET switch of the controller from RUN→STOP.
(4) Change the RUN/STOP/RESET switch of the controller from STOP→RUN.

- If the latch relay (L) is specified as the execution command and the system is powered on while the latch relay is on, the execution command turns on in the first scan, triggering execution of the PLS instruction and turning on the specified device. The device that has been turned on in the first scan after power-on can be turned off by the next PLS instruction.
- The PLS instruction performs OFF processing at the execution of the next instruction after the instruction execution.

Point

- Note that if the PLS instruction is jumped by using the CJ instruction or the executed subroutine program is not called by using the CALL(P) instruction, the device specified by (d) may be on for more than one scan.

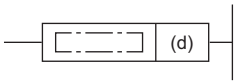
Operation error

There is no operation error.

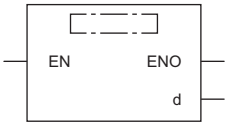
Falling edge output

PLF

This instruction turns on the specified device for one scan on the falling edge (on to off) of the execution command.

Ladder	ST
	ENO:=PLF(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
PLF	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Pulse conversion target device number	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

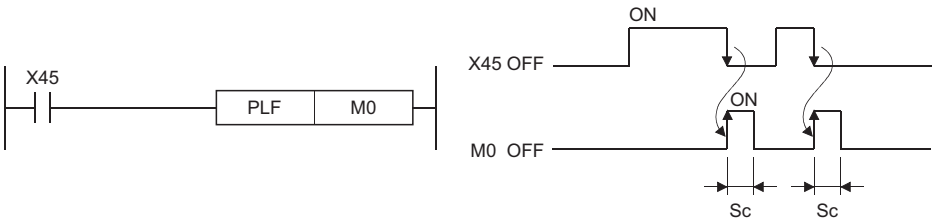
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	(DY)
(d)	○	○	○	○	—	—	—	○	—	—	—	○

Processing details

- This instruction turns on the specified device on the falling edge (on to off) of the execution command. When the execution command is in another state (staying off, rising edge (off to on), or staying on), the instruction turns off the specified device. If only one PLF instruction in the device specified by (d) is executed during one scan, the specified device turns on for one scan. For the operation to be performed if more than one PLF instruction is executed during one scan, refer to the following.

☞ Page 52 Operations arising when the OUT, SET/RST, and PLS/PLF instructions of the same device are used



Sc: 1 scan

- Once after execution of the PLF instruction, even if the switch of the controller is moved to the STOP position and then the RUN position again, the PLF instruction is not executed.
- The PLF instruction performs OFF processing at the execution of the next instruction after the instruction execution.

Point

- Note that if the PLF instruction is jumped by using the CJ instruction or the executed subroutine program is not called by using the CALL(P) instruction, the device specified by (d) may be on for more than one scan.

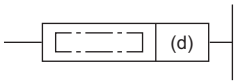
Operation error

There is no operation error.

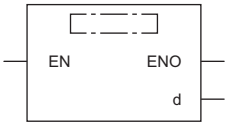
Inverting the bit device output

FF

This instruction inverts the status of the specified device.

Ladder	ST
	ENO:=FF(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FF	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Inversion target device number	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DY)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	—	—	—	○	—	—	—	○

Processing details

- This instruction inverts the status of the device specified by (d) on the rising edge of the execution command.

Device	Device status	
	Before execution of the FF instruction	After execution of the FF instruction
Bit device	Off	On
	On	Off
Bit-specified word device	0	1
	1	0

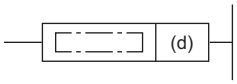
Operation error

There is no operation error.

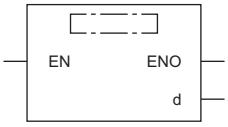
Converting the direct access output into a pulse

DELTA(P)

These instructions convert the specified direct access output (DY) into pulse output.

Ladder	ST
	ENO:=DELTA(EN,d); ENO:=DELTAP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DELTA	
DELTAP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Pulse conversion target device number	—	Bit	ANY_BOOL*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to the device (DY) can be used.

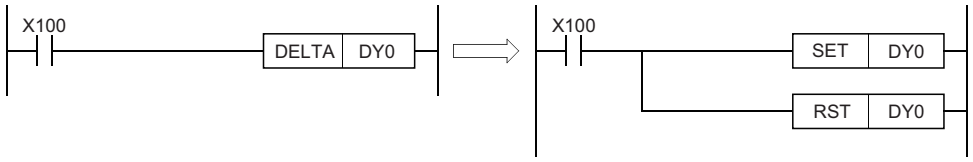
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	(DY)
(d)	—	—	—	—	—	—	—	—	—	—	—	○

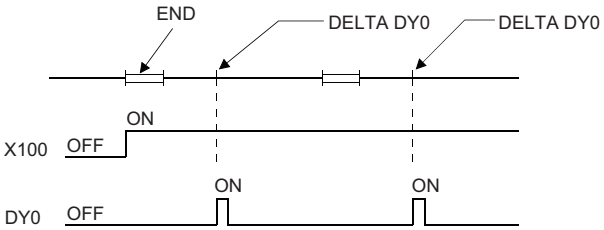
Processing details

- These instructions convert the direct access output (DY) specified by (d) into pulse output. If DY0 is specified by (d), the program operates in the same way as the one that uses the SET and RST instructions.

The following figure shows an example when a ladder using the DELTA instruction is replaced with a ladder using the SET/RST instructions.



The following figure shows the operation of the instruction.



Precautions

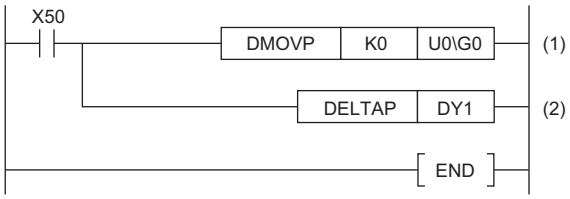
These instructions are used as an execution command (rising edge execution) for intelligent function modules.
These instructions cannot be used as an actual output command for output modules.

Operation error

There is no operation error.

Point

The DELTA(P) instruction is used to set a preset value of the high-speed counter module.
[Example]
A program that presets the CH1 of the high-speed counter module (RD62P2) mounted on slot 0 of the base unit when X50 turns on



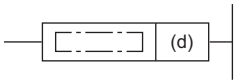
(1) Store the preset value (0) in the buffer memory areas 0 and 1 of the RD62P2.
(2) Output the preset command.

11.4 Shift Instructions

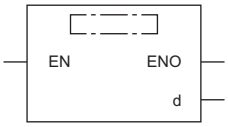
Shifting bit devices

SFT(P)

These instructions shift the on/off state of the device area just before the one specified to the specified device area, and turn off the shift source device.

Ladder	ST
	ENO:=SFT(EN,d); ENO:=SFTP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFT	
SFTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (DY)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	—	—	—	○	—	—	—	○

Processing details

■Bit device

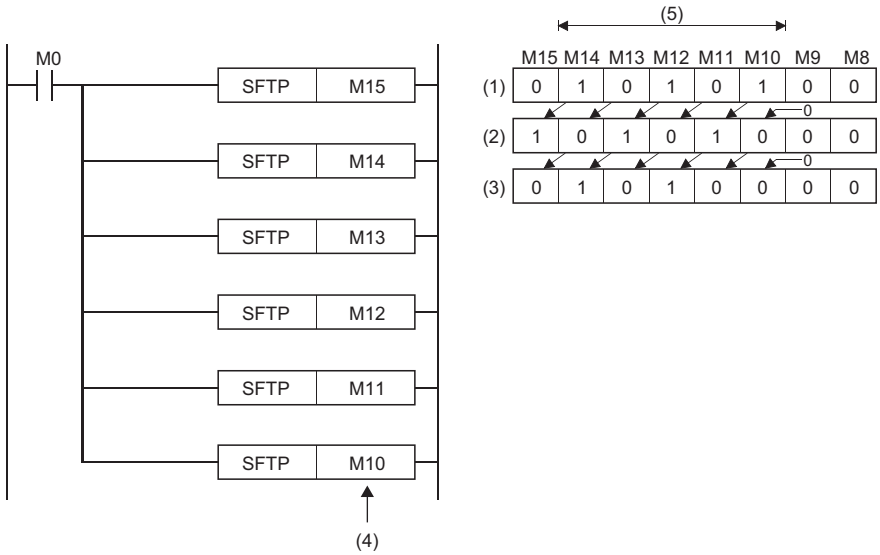
- These instructions shift the on/off state of the device area just before the one specified by (d) to the device area specified by (d). After the data is shifted, the data of the shift source device area is turned off.

Point

When the SFT(P) instruction is used consecutively, program devices in descending order of the device numbers.

Ex.

The following figure shows an example when the SFTP instruction shifts the device from M10 to M11.



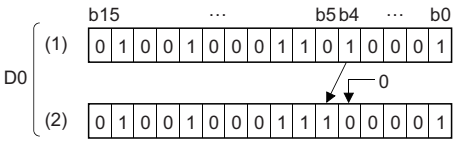
(1) Initial state
(2) After the 1st shift input
(3) After the 2nd shift input
(5) Shift range

■Bit-specified word device

- These instructions shift the 1/0 state of the bit just before the one specified by (d) to the bit specified by (d). After the data is shifted, the data of the shift source bit is set to 0.

Ex.

The SFT(P) instruction that specifies D0.5 (b5 in D0) is executed, it shifts the 1/0 state of b4 in D0 to b5, and sets b4 to 0.



(1) Before shifting the bit
(2) After shifting the bit

Operation error

There is no operation error.

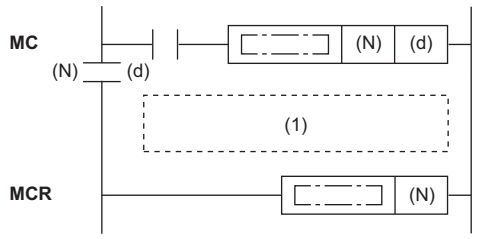
11.5 Master Control Instructions

Setting/resetting a master control

MC, MCR

- MC: This instruction starts a master control.
- MCR: This instruction ends a master control.

Ladder



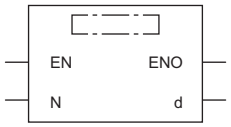
(1) Master control ladder

ST

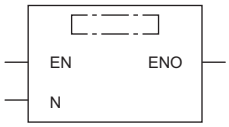
```
ENO:=MC(EN,N,d);
ENO:=MCR(EN,N);
```

FBD/LD

MC



MCR



■Execution condition

Instruction	Execution condition
MC MCR	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(N)	Nesting	N0 to N14	Device name	ANY16_S*1
(d)	Number of the device to be turned on	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to the device (N) or to which constants are assigned can be used.

■Applicable devices

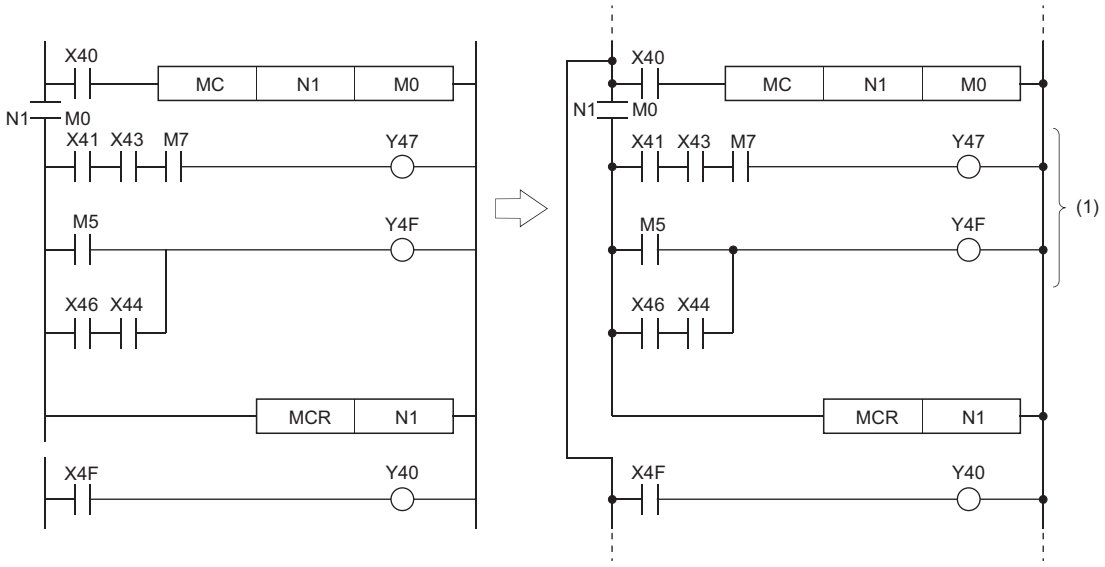
Operand	Bit		Word			Double word		Indirect specification	Constant			Others	
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	N	DY
(N)	—	—	—	—	—	—	—	—	—	—	—	○	—
(d)	○	○	○	—	—	—	—	○	—	—	—	—	○

Processing details

These instructions are used to create an efficient ladder switching program by opening and closing the common rails of the ladder.

The following is the program example using the master control instructions.

(Left: Display on the engineering tool, Right: Actual operation)



(1) Executed only when X40 is on.

■MC

- If the execution command of the MC instruction is on when a master control starts, the operation result between the MC and MCR instructions will be the one as programmed. If the execution command is off, the operation result between the MC and MCR instructions will be as follows.

Device	Status
High-speed timer Low-speed timer	The count value is set to 0, and both the coil and contact are turned off.
High-speed retentive timer Low-speed retentive timer Counter	The coil is turned off, but both the count value and contact maintain the current status.
Device used by the OUT instruction	Forcibly turned off.
Device used by the SET and RST instructions Device used by the SFT(P) instruction Device used by basic instructions and application instructions	Maintains the current status.

- Even if the MC instruction is off, the instructions between the MC and MCR instructions are executed and therefore the scan time is not shortened.

Point

- When a ladder performing a master control includes an instruction which does not require a contact instruction (such as the FOR to NEXT instruction), the controller executes the instruction regardless of the execution command of the MC instruction.
- To create an easy-to-understand program, use the MC and MCR instructions within a single program block.

- The MC instruction can use the same nesting (N) number as many times as needed by specifying different devices in (d).
- When the MC instruction is on, the coil of the device specified by (d) turns on. Using the same device for the OUT instruction causes double coils. Do not use the device specified by (d) in other instructions.

■MCR

- This instruction is a master control reset instruction which indicates the end of the master control area.
- Do not place any contact instruction before the MCR instruction.
- Use the MC and MCR instructions with the same nesting number as a set. Note that if the MCR instructions are nested in one place, all master controls can be terminated by specifying the lowest nesting (N) number. (Refer to "Precautions".)

Operation error

There is no operation error.

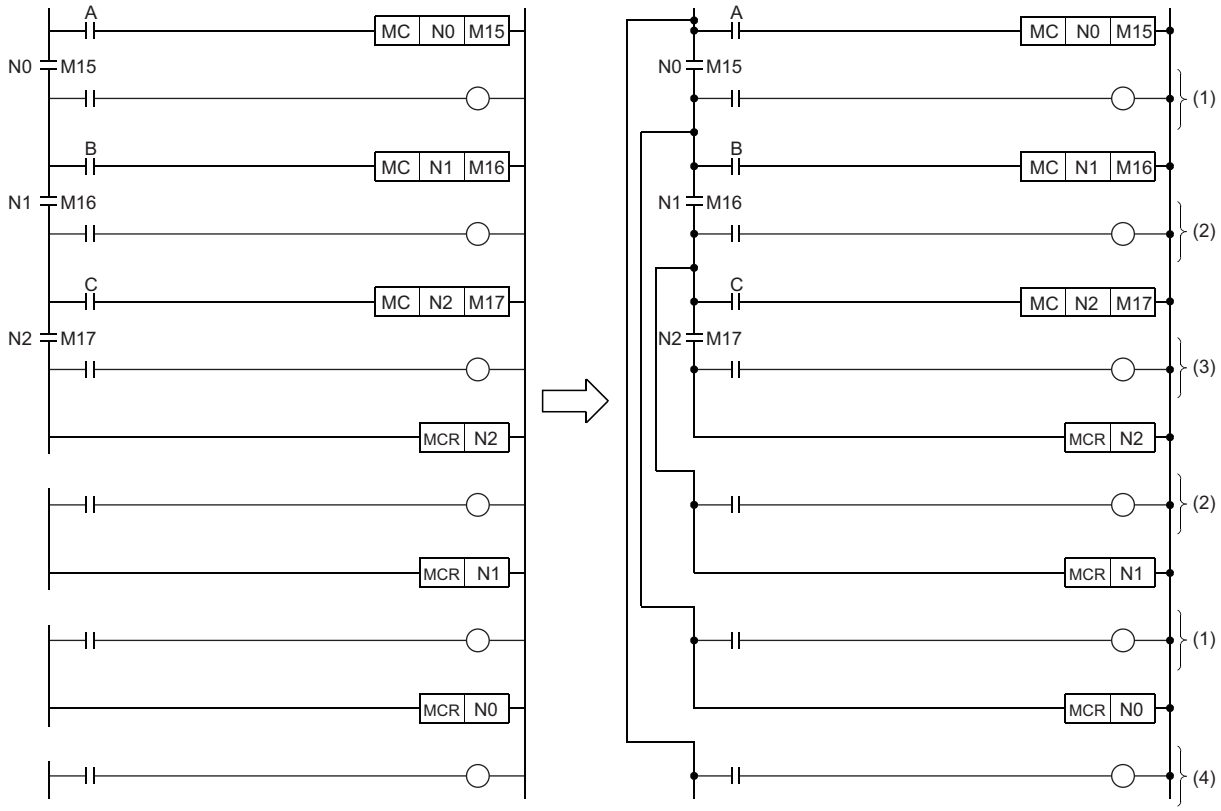
Point

The master control instructions can be nested. Individual master control areas are distinguished by nesting (N) numbers. Nesting can be set from N0 to N14.

Using the nesting structure enables the creation of a ladder which can sequentially constrain the program execution conditions.

The following figure shows a ladder program example using the nesting structure.

(Left: Display on the engineering tool, Right: Actual operation)



(1) Executed when A is on.

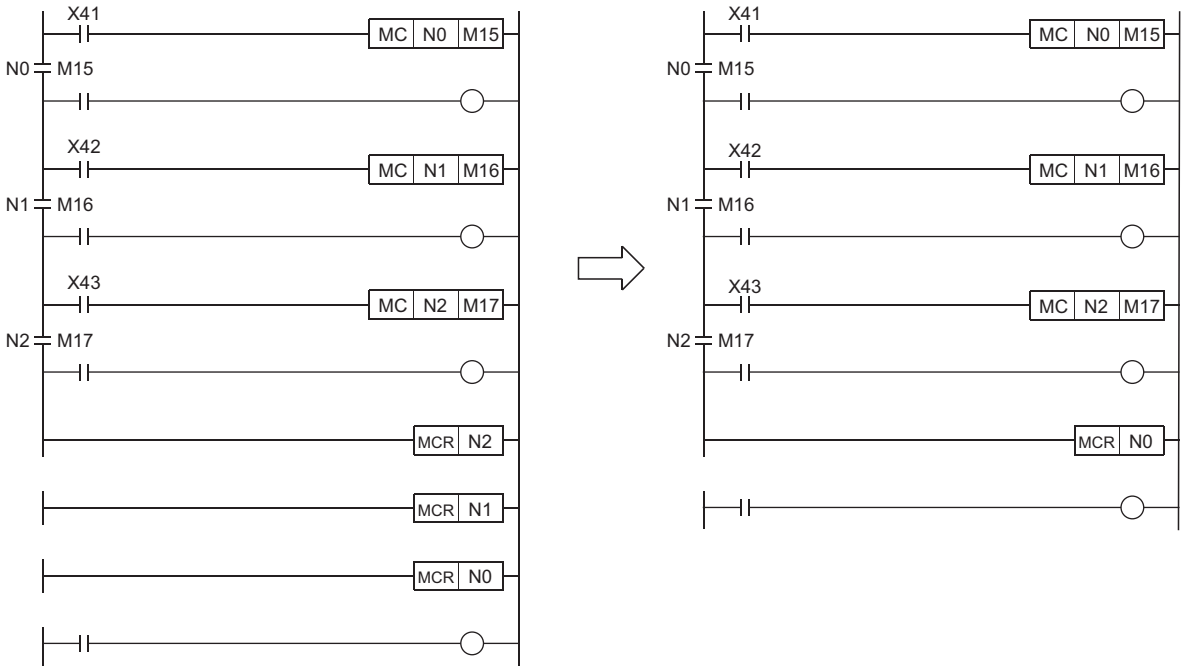
(2) Executed when A and B are on.

(3) Executed when A, B, and C are on.

(4) Executed regardless of the status of A, B, and C

Precautions

- Up to 15 nests (N0 to N14) are allowed. When nesting is performed, the MC instruction should use nesting (N) numbers in order from lower numbers and the MCR instruction should use them in order from higher numbers.
- If the MCR instructions are nested in one place, all master controls can be terminated by specifying the lowest nesting (N) number.



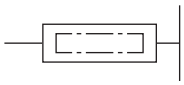
- If the MCR instruction is used in the structured text and FBD/LD languages, specifying a variable or constant which is not always on (TRUE/1/SM400) in EN (execution condition) will cause a conversion error.

11.6 Termination Instructions

Ending the main routine program

FEND

This instruction is used to separate the main routine program from subroutine programs and interrupt programs in a program file.

Ladder	ST
	Not supported

FBD/LD
Not supported

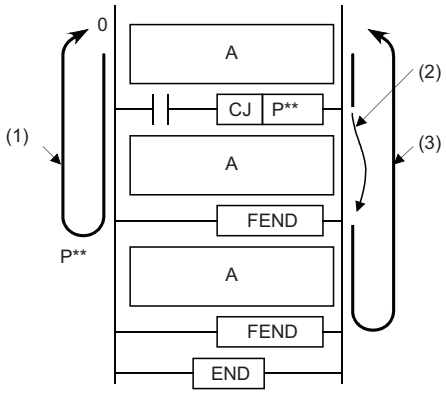
■Execution condition

Instruction	Execution condition
FEND	Every scan

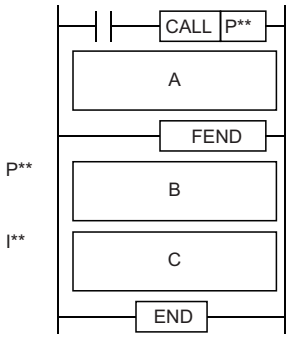
Processing details

- This instruction is used to divide sequence program operations by using a program branch instruction such as the CJ instruction or to separate the main routine program from subroutine programs and interrupt programs specified by the interrupt pointer (I).
- When the instruction is executed, the controller terminates the running program.
- Sequence programs following the FEND instruction can be displayed on the engineering tool (ladder mode).

When the CJ instruction is used



When there are subroutine and interrupt programs



A: Main routine program
B: Subroutine program
C: Interrupt program
(1) Operation performed when the CJ instruction is not executed
(2) Jump caused by the CJ instruction
(3) Operation performed when the CJ instruction is executed

Operation error

Error code (SD0)	Description
3341H	After execution of the CALL(P), FCALL(P), ECALL(P), or EFCALL(P) instruction, the FEND instruction is executed before the RET instruction.
3351H	Within the interrupt program specified by the interrupt pointer (I), the FEND instruction is executed before the IRET instruction.

Ending the sequence program

END

This instruction indicates the end of a program.

Ladder	ST
	Not supported

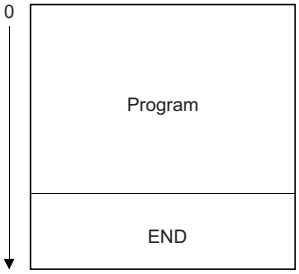
FBD/LD
Not supported

■Execution condition

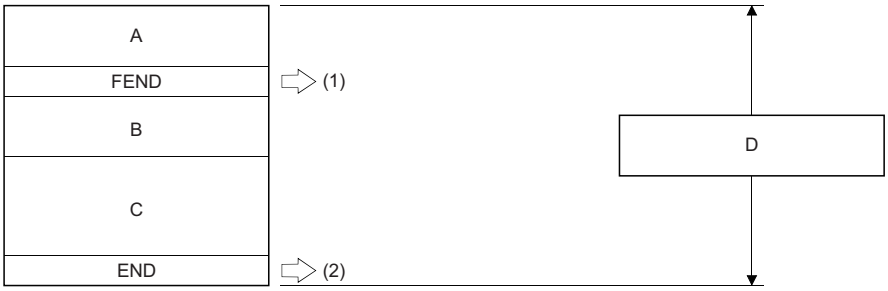
Instruction	Execution condition
END	Every scan

Processing details

- This instruction indicates the end of a program including a main routine program, subroutine programs, and interrupt programs.
- When the instruction is executed, the controller terminates the running program.



- If END processing is required in the middle of a program, use the FEND instruction.
- If the program is created using the engineering tool (in ladder edit mode), the END instruction is automatically input and cannot be edited.
- The following figure shows how to use the termination instructions when a main routine program, subroutine program, and interrupt program exist.



A: Main routine program
B: Subroutine program
C: Interrupt program
D: Main sequence program area
(1): The FEND instruction is required.
(2): The END instruction is required.

Point

When a program is divided into multiple program blocks, the END instruction indicates the end of a program block.
The END instruction within the program registered at the end of the program setting performs END processing.

Operation error

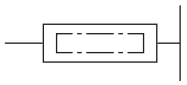
Error code (SD0)	Description
3341H	After execution of the CALL(P), FCALL(P), ECALL(P), or EFCALL(P) instruction, the END instruction is executed before the RET instruction.
3351H	Within the interrupt program specified by the interrupt pointer (I), the END instruction is executed before the IRET instruction.

11.7 Stop Instruction

Stopping the sequence program

STOP

Stops the operation of the controller. (The operation of this instruction is the same as setting the switch of the controller to the STOP position.)

Ladder	ST
	ENO:=STOP(EN);

FBD/LD



■Execution condition

Instruction	Execution condition
STOP	

Processing details

- This instruction resets the output (Y) and stops the operation of the controller when the execution command turns on. (The operation of this instruction is the same as setting the switch of the controller to the STOP position.)
- To restart the operation of the controller after execution of the STOP instruction, set the switch back to STOP, and then set it to RUN again.

Operation error

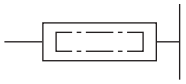
Error code (SD0)	Description
3341H	After execution of the CALL(P), FCALL(P), ECALL(P), EFCALL(P), or XCALL instruction, the STOP instruction is executed before the RET instruction.
3351H	Within the interrupt program specified by the interrupt pointer (I), the STOP instruction is executed before the IRET instruction.
3353H	The STOP instruction is executed within a fixed scan execution type program.
3354H	The STOP instruction is executed in an event execution type program.

11.8 No Operation Instruction

No operation (NOPLF)

NOPLF

This instruction is a no-operation instruction and has no impact on the previous operations.

Ladder	ST
	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
NOPLF	Every scan

Processing details

- This instruction is a no-operation instruction and has no impact on the previous operations.

Operation error

There is no operation error.

12 BASIC INSTRUCTIONS

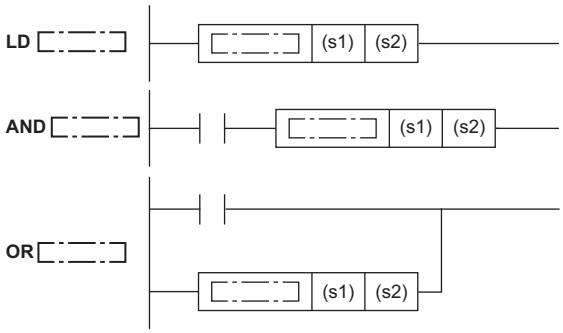
12.1 Comparison Operation Instructions

Comparing 16-bit binary data

LD□(_U), AND□(_U), OR□(_U)

These instructions compare the two sets of 16-bit binary data specified. (Devices are used as normally open contacts.)

Ladder



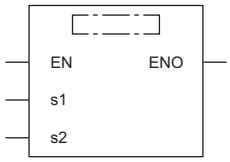
(□ is to be replaced by any of the following: =(_U), <>(_U), >(_U), <=(_U), <(_U), >=(_U).)

ST

ENO:=LD_□(EN,s1,s2); ENO:=AND_□(EN,s1,s2); ENO:=OR_□(EN,s1,s2);	ENO:=LD_□_U(EN,s1,s2); ENO:=AND_□_U(EN,s1,s2); ENO:=OR_□_U(EN,s1,s2);

(□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)*1

FBD/LD



(□ is to be replaced by combination of any of the following: LD_, AND_, OR_ and EQ(_U), NE(_U), GT(_U), LE(_U), LT(_U), GE(_U).)*1

*1 EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
LD□(_U), AND□(_U), OR□(_U)	Every scan

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	LD□, AND□, OR□	Comparison data or the device where comparison data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	LD□_U, AND□_U, OR□_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	LD□, AND□, OR□	Comparison data or the device where comparison data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	LD□_U, AND□_U, OR□_U		0 to 65535	16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions perform a comparison operation between the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2). (Devices are used as normally open contacts.)
- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
=(_U), EQ(_U)	(s1)=(s2)	Continuity state (ENO is on.)
<>(_U), NE(_U)	(s1)≠(s2)	
>(_U), GT(_U)	(s1)>(s2)	
<=(_U), LE(_U)	(s1)≤(s2)	
<(_U), LT(_U)	(s1)<(s2)	
>=(_U), GE(_U)	(s1)≥(s2)	
=(_U), EQ(_U)	(s1)≠(s2)	Non-continuity state (ENO is off.)
<>(_U), NE(_U)	(s1)=(s2)	
>(_U), GT(_U)	(s1)≤(s2)	
<=(_U), LE(_U)	(s1)>(s2)	
<(_U), LT(_U)	(s1)≥(s2)	
>=(_U), GE(_U)	(s1)<(s2)	

- When hexadecimal constants are used for (s1) and (s2) and the numerical value (8 to F) whose most significant bit (b15) is 1 is specified as a constant, the value is considered as a negative binary value in comparison operation.
- If the LD□ instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.
- If the OR□ instruction is used in the program written in FBD/LD and EN is set to TRUE, ENO turns on. EN will not be an execution condition.

Operation error

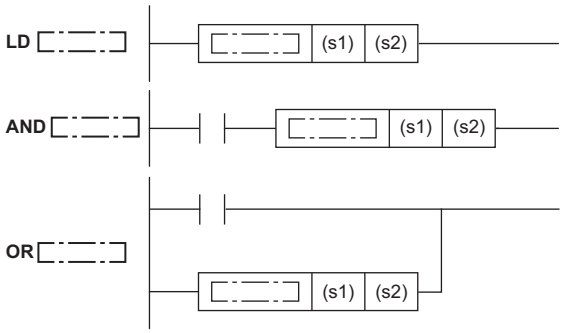
There is no operation error.

Comparing 32-bit binary data

LDD□(_U), ANDD□(_U), ORD□(_U)

These instructions compare the two sets of 32-bit binary data specified. (Devices are used as normally open contacts.)

Ladder



(□ is to be replaced by any of the following: D=(_U), D<>(_U), D>(_U), D<=(_U), D<(_U), D>=(_U).)

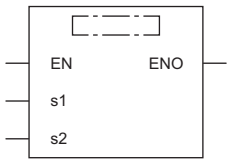
ST

ENO:=LDD_□(EN,s1,s2);
ENO:=ANDD_□(EN,s1,s2);
ENO:=ORD_□(EN,s1,s2);

ENO:=LDD_□_U(EN,s1,s2);
ENO:=ANDD_□_U(EN,s1,s2);
ENO:=ORD_□_U(EN,s1,s2);

(□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)*1

FBD/LD



(□ is to be replaced by combination of any of the following: LDD_, ANDD_, ORD_ and EQ(_U), NE(_U), GT(_U), LE(_U), LT(_U), GE(_U).)*1

*1 EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
LDD□(_U), ANDD□(_U), ORD□(_U)	Every scan

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	LDD□, ANDD□, ORD□	Comparison data or the start device where the comparison data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	LDD□_U, ANDD□_U, ORD□_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	LDD□, ANDD□, ORD□	Comparison data or the start device where the comparison data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	LDD□_U, ANDD□_U, ORD□_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—

Processing details

- These instructions perform a comparison operation between the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2). (Devices are used as normally open contacts.)
- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
D=(_U), EQ(_U)	(s1)=(s2)	Continuity state (ENO is on.)
D<>(_U), NE(_U)	(s1)≠(s2)	
D>(_U), GT(_U)	(s1)>(s2)	
D<=(_U), LE(_U)	(s1)≤(s2)	
D<(_U), LT(_U)	(s1)<(s2)	
D>=(_U), GE(_U)	(s1)≥(s2)	
D=(_U), EQ(_U)	(s1)≠(s2)	Non-continuity state (ENO is off.)
D<>(_U), NE(_U)	(s1)=(s2)	
D>(_U), GT(_U)	(s1)≤(s2)	
D<=(_U), LE(_U)	(s1)>(s2)	
D<(_U), LT(_U)	(s1)≥(s2)	
D>=(_U), GE(_U)	(s1)<(s2)	

- When hexadecimal constants are specified for (s1) and (s2) and the numerical value (8 to F) whose most significant bit (b31) is 1 is specified as a constant, the value is considered as a negative binary value in comparison operation.
- To specify the compare target data, use an instruction which handles 32-bit data, such as the DMOV(P) instruction. If an instruction which handles 16-bit data, such as the MOV(P) instruction, is used, comparison cannot be performed normally.
- If the LDD□ instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.
- If the ORD□ instruction is used in the program written in FBD/LD and EN is set to TRUE, ENO turns on. EN will not be an execution condition.

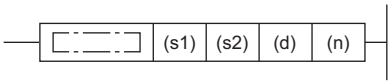
Operation error

There is no operation error.

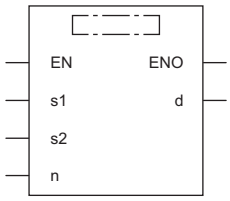
Comparing 16-bit binary block data

BKCMP□(P)(_U)

These instructions compare the two sets of 16-bit binary block data specified.

Ladder	ST	
	ENO:=BKCMP_□(EN,s1,s2,n,d); ENO:=BKCMP_□P(EN,s1,s2,n,d);	ENO:=BKCMP_□_U(EN,s1,s2,n,d); ENO:=BKCMP_□P_U(EN,s1,s2,n,d) ;
(□ is replaced by any of the following: BKCMP=(P)(_U), BKCMP<>(P)(_U), BKCMP>(P)(_U), BKCMP<=(P)(_U), BKCMP<(P)(_U), BKCMP>=(P)(_U).)	(□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)*1	

FBD/LD



(□ is to be replaced by combination of any of the following: BKCMP_ and EQ(P)(_U), NE(P)(_U), GT(P)(_U), LE(P)(_U), LT(P)(_U), GE(P)(_U).)*1

*1 EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
BKCMP□(_U)	
BKCMP□P(_U)	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	BKCMP□(P)	Comparison data or the start device where the comparison data is stored	-32768 to 32767	16-bit signed binary	ANY16_S*1
	BKCMP□(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U*1
(s2)	BKCMP□(P)	Start device where the comparison data is stored	—	16-bit signed binary	ANY16_S*1
	BKCMP□(P)_U			16-bit unsigned binary	ANY16_U*1
(d)		Start device for storing the comparison operation result	—	Bit	ANY_BOOL*1
(n)		Number of data points to be compared	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

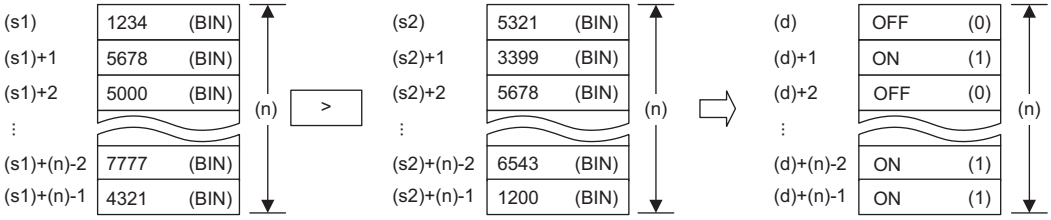
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

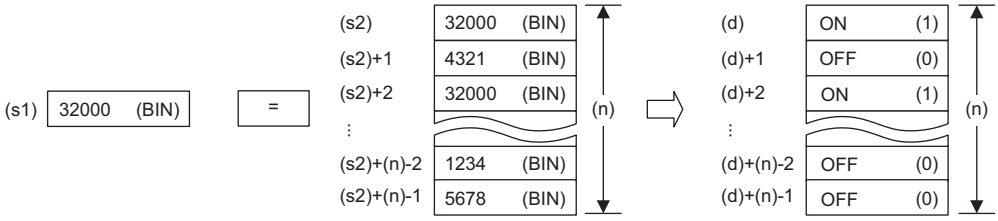
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions compare the (n) points of 16-bit binary data from the device specified by (s1) with the (n) points of 16-bit binary data from the device specified by (s2), and stores the operation result in the device specified by (d) and later.
- If the comparison condition is satisfied, the relevant device specified by (d) turns on; otherwise, the device turns off.



- Specify data in units of 16 bits.
- A constant can be specified for (s1).



- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
BKCMP=(P)(_U), BKCMP_EQ(P)(_U)	(s1)=(s2)	On (1)
BKCMP<>(P)(_U), BKCMP_NE(P)(_U)	(s1)≠(s2)	
BKCMP>(P)(_U), BKCMP_GT(P)(_U)	(s1)>(s2)	
BKCMP<=(P)(_U), BKCMP_LE(P)(_U)	(s1)≤(s2)	
BKCMP<(P)(_U), BKCMP_LT(P)(_U)	(s1)<(s2)	
BKCMP>=(P)(_U), BKCMP_GE(P)(_U)	(s1)≥(s2)	
BKCMP=(P)(_U), BKCMP_EQ(P)(_U)	(s1)≠(s2)	Off (0)
BKCMP<>(P)(_U), BKCMP_NE(P)(_U)	(s1)=(s2)	
BKCMP>(P)(_U), BKCMP_GT(P)(_U)	(s1)≤(s2)	
BKCMP<=(P)(_U), BKCMP_LE(P)(_U)	(s1)>(s2)	
BKCMP<(P)(_U), BKCMP_LT(P)(_U)	(s1)≥(s2)	
BKCMP>=(P)(_U), BKCMP_GE(P)(_U)	(s1)<(s2)	

- When the comparison operation results stored in (n) points from the device specified by (d) are all on (1), SM704 turns on.

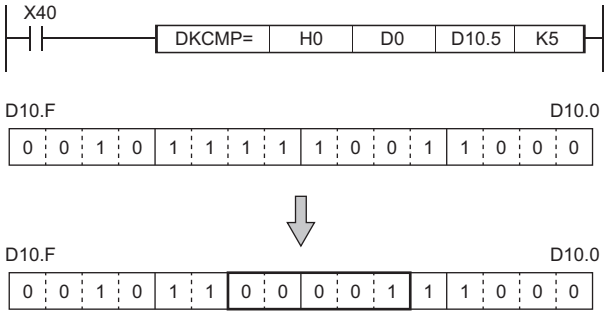
Operation error

Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping.
	The device ranges starting from the ones specified by (s2) and (d) are overlapping.

Point

When bits of a word device are specified, the bits other than the specified ones for storing the operation result do not change as shown below.

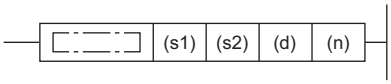
- Number of data points to be compared (n): K5
- Start number of the device for storing the comparison operation result (d): D10.5



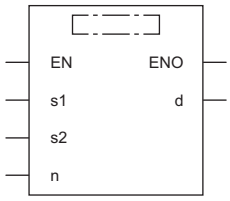
Comparing 32-bit binary block data

DBKCMP□(P)(_U)

These instructions compare the two sets of 32-bit binary block data specified.

Ladder	ST	
	ENO:=DBKCMP_□(EN,s1,s2,n,d); ENO:=DBKCMP_□P(EN,s1,s2,n,d);	ENO:=DBKCMP_□_U(EN,s1,s2,n,d); ENO:=DBKCMP_□P_U(EN,s1,s2,n,d);
(□ is replaced by any of the following: DBKCMP=(P)(_U), DBKCMP<>(P)(_U), DBKCMP>(P)(_U), DBKCMP<=(P)(_U), DBKCMP<(P)(_U), DBKCMP>=(P)(_U).)	(□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)*1	

FBD/LD



(□ is to be replaced by combination of any of the following: DBKCMP_ and EQ(P)(_U), NE(P)(_U), GT(P)(_U), LE(P)(_U), LT(P)(_U), GE(P)(_U).)*1

*1 EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
DBKCMP□(_U)	
DBKCMP□P(_U)	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DBKCMP□(P)	Comparison data or the start device where the comparison data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S*1
	DBKCMP□(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U*1
(s2)	DBKCMP□(P)	Start device where the comparison data is stored	—	32-bit signed binary	ANY32_S*1
	DBKCMP□(P)_U			32-bit unsigned binary	ANY32_U*1
(d)		Start device for storing the comparison operation result	—	Bit	ANY_BOOL*1
(n)		Number of data points to be compared	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

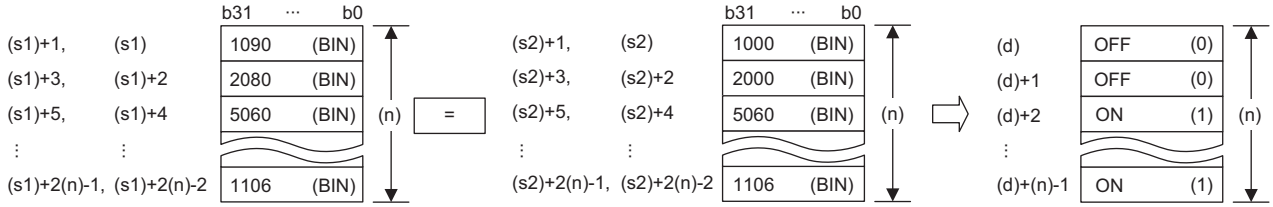
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

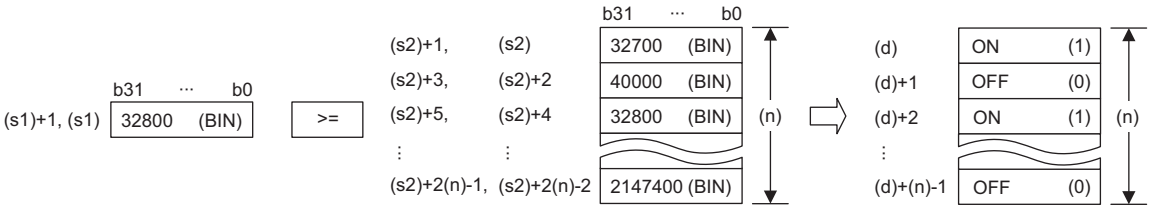
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions compare the (n) points of 32-bit binary data from the device specified by (s1) with the (n) points of 32-bit binary data from the device specified by (s2), and stores the operation result in the device specified by (d) and later.
- If the comparison condition is satisfied, the relevant device specified by (d) turns on; otherwise, the device turns off.



- Comparison operation is performed in units of 32 bits.
- A constant can be specified for (s1).



- Specify (d) outside the device ranges for (n) points from the device specified by (s1) and those from the device specified by (s2).
- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
DBKCMP=(P)(_U), DBKCMP_EQ(P)(_U)	(s1)=(s2)	On (1)
DBKCMP<>(P)(_U), DBKCMP_NE(P)(_U)	(s1)≠(s2)	
DBKCMP>(P)(_U), DBKCMP_GT(P)(_U)	(s1)>(s2)	
DBKCMP<=(P)(_U), DBKCMP_LE(P)(_U)	(s1)≤(s2)	
DBKCMP<(P)(_U), DBKCMP_LT(P)(_U)	(s1)<(s2)	
DBKCMP>=(P)(_U), DBKCMP_GE(P)(_U)	(s1)≥(s2)	
DBKCMP=(P)(_U), DBKCMP_EQ(P)(_U)	(s1)≠(s2)	Off (0)
DBKCMP<>(P)(_U), DBKCMP_NE(P)(_U)	(s1)=(s2)	
DBKCMP>(P)(_U), DBKCMP_GT(P)(_U)	(s1)≤(s2)	
DBKCMP<=(P)(_U), DBKCMP_LE(P)(_U)	(s1)>(s2)	
DBKCMP<(P)(_U), DBKCMP_LT(P)(_U)	(s1)≥(s2)	
DBKCMP>=(P)(_U), DBKCMP_GE(P)(_U)	(s1)<(s2)	

- When the comparison operation results stored in (n) points from the device specified by (d) are all on (1), SM704 turns on.
- If (n) is 0, no processing is performed.

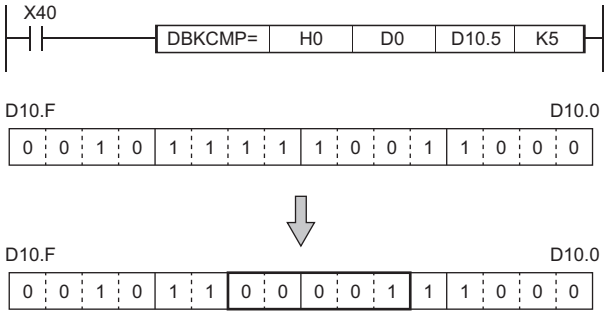
Operation error

Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping.
	The device ranges starting from the ones specified by (s2) and (d) are overlapping.

Point

When bits of a word device are specified, the bits other than the specified ones for storing the operation result do not change as shown below.

- Number of data points to be compared (n): K5
- Start number of the device for storing the comparison operation result (d): D10.5

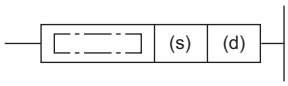


12.2 Arithmetic Operation Instructions

Adding 16-bit binary data

+(P)(_U) [when two operands are set]

These instructions add the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 266 +(P)(_U) [when three operands are set])

FBD/LD
Not supported (☞ Page 266 +(P)(_U) [when three operands are set])

■Execution condition

Instruction	Execution condition
+ +_U	
+P +P_U	

Setting data

■Description, range, data type

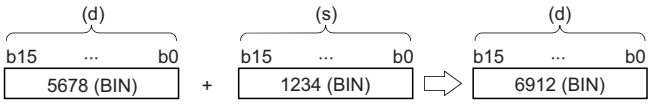
Operand		Description	Range	Data type	Data type (label)
(s)	+(P)	Second addend data or the device where the second addend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	+(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	+(P)	Device where the first addend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	+(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

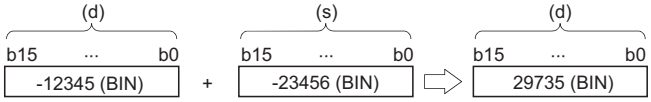
Processing details

- These instructions add the 16-bit binary data in the device specified by (d) and the 16-bit binary data in the device specified by (s), and store the operation result in the device specified by (d).



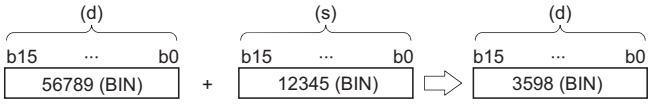
- If an overflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[+(P) instruction]





[+(P)_U instruction]

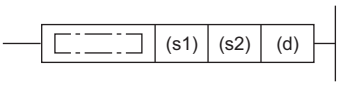


Operation error

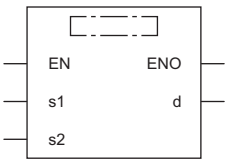
There is no operation error.

+(P)(_U) [when three operands are set]

These instructions add the two sets of 16-bit binary data specified.

Ladder	ST
	ENO:=PLUS(EN,s1,s2,d); ENO:=PLUSP(EN,s1,s2,d); ENO:=PLUS_U(EN,s1,s2,d); ENO:=PLUSP_U(EN,s1,s2,d);

FBD/LD



(☐ is to be replaced by any of the following: PLUS, PLUSP, PLUS_U, PLUSP_U.)

■Execution condition

Instruction	Execution condition
+ +_U	
+P +P_U	

Setting data

■Description, range, data type

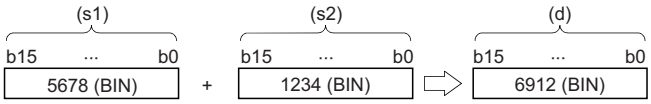
Operand		Description	Range	Data type	Data type (label)
(s1)	+(P)	First addend data or the device where the first addend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	+(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	+(P)	Second addend data or the device where the second addend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	+(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	+(P)	Device for storing the operation result	—	16-bit signed binary	ANY16_S
	+(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions add the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



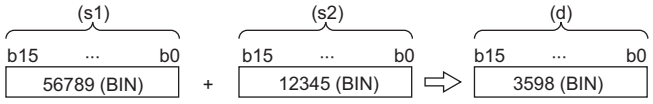
- If an overflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[+(P) instruction]





[+(P)_U instruction]



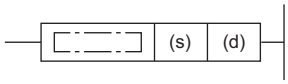
Operation error

There is no operation error.

Subtracting 16-bit binary data

-(P)(_U) [when two operands are set]

These instructions perform subtraction between the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 270 -(P)(_U) [when three operands are set])

FBD/LD
Not supported (☞ Page 270 -(P)(_U) [when three operands are set])

■Execution condition

Instruction	Execution condition
- -_U	
-P -P_U	

Setting data

■Description, range, data type

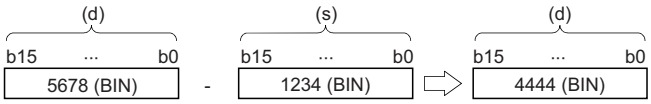
Operand		Description	Range	Data type	Data type (label)
(s)	-(P)	Subtrahend data or the device where subtrahend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	-(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	-(P)	Device where minuend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	-(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

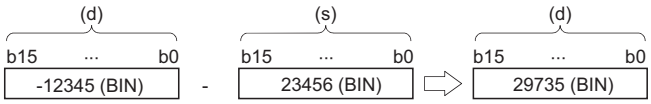
Processing details

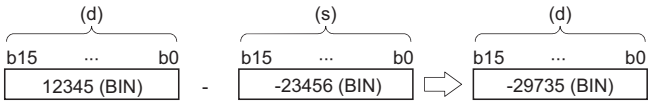
- These instructions subtract the 16-bit binary data in the device specified by (s) from the 16-bit binary data in the device specified by (d), and store the operation result in the device specified by (d).



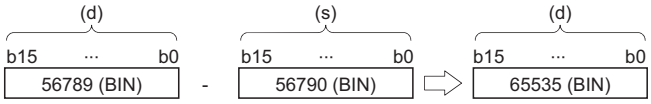
- If an underflow occurs in the result, the borrow bit is ignored. In this case, SM700 does not turn on.

[-(P) instruction]





[-(P)_U instruction]

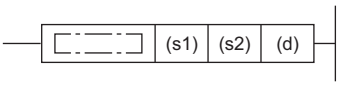


Operation error

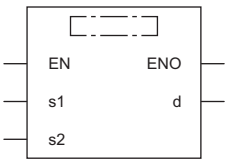
There is no operation error.

-(P)(_U) [when three operands are set]

These instructions perform subtraction between the two sets of 16-bit binary data specified.

Ladder	ST
	ENO:=MINUS(EN,s1,s2,d); ENO:=MINUSP(EN,s1,s2,d); ENO:=MINUS_U(EN,s1,s2,d); ENO:=MINUSP_U(EN,s1,s2,d);

FBD/LD



(☐ is to be replaced by any of the following: MINUS, MINUSP, MINUS_U, MINUSP_U.)

■Execution condition

Instruction	Execution condition
- -_U	
-P -P_U	

Setting data

■Description, range, data type

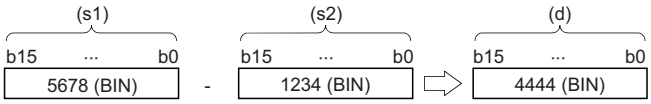
Operand		Description	Range	Data type	Data type (label)
(s1)	-(P)	Minuend data or the device where minuend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	-(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	-(P)	Subtrahend data or the device where subtrahend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	-(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	-(P)	Device for storing the operation result	—	16-bit signed binary	ANY16_S
	-(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

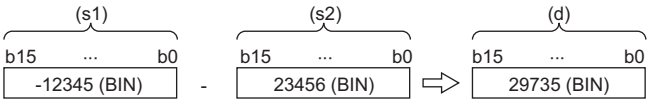
Processing details

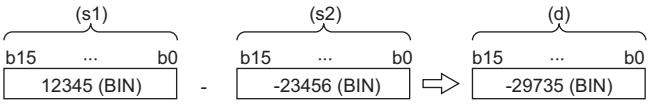
- These instructions subtract the 16-bit binary data in the device specified by (s2) from the 16-bit binary data in the device specified by (s1), and store the operation result in the device specified by (d).



- If an underflow occurs in the result, the borrow bit is ignored. In this case, SM700 does not turn on.

[-(P) instruction]





[-(P)_U instruction]



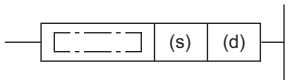
Operation error

There is no operation error.

Adding 32-bit binary data

D+(P)(_U) [when two operands are set]

These instructions add the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 274 D+(P)(_U) [when three operands are set])

FBD/LD
Not supported (☞ Page 274 D+(P)(_U) [when three operands are set])

■Execution condition

Instruction	Execution condition
D+ D+_U	
D+P D+P_U	

Setting data

■Description, range, data type

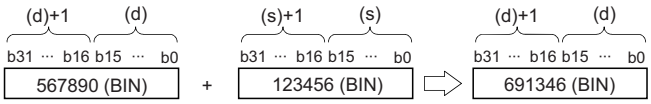
Operand		Description	Range	Data type	Data type (label)
(s)	D+(P)	Second addend data or the start device where the second addend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D+(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	D+(P)	Start device where the first addend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D+(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

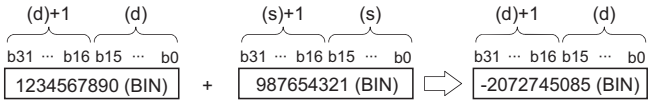
Processing details

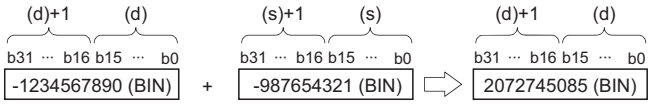
- These instructions add the 32-bit binary data in the device specified by (d) and the 32-bit binary data in the device specified by (s), and store the operation result in the device specified by (d).



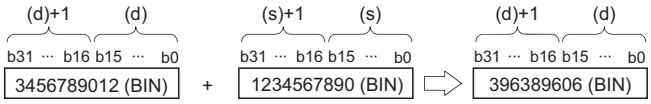
- If an overflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[D+(P) instruction]





[D+(P)_U instruction]

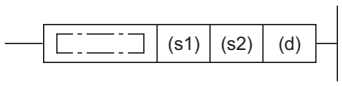


Operation error

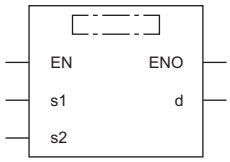
There is no operation error.

D+(P)(_U) [when three operands are set]

These instructions add the two sets of 32-bit binary data specified.

Ladder	ST
	ENO:=DPLUS(EN,s1,s2,d); ENO:=DPLUSP(EN,s1,s2,d); ENO:=DPLUS_U(EN,s1,s2,d); ENO:=DPLUSP_U(EN,s1,s2,d);

FBD/LD



(☐ is to be replaced by any of the following: DPLUS, DPLUSP, DPLUS_U, DPLUSP_U.)

■Execution condition

Instruction	Execution condition
D+ D+_U	
D+P D+P_U	

Setting data

■Description, range, data type

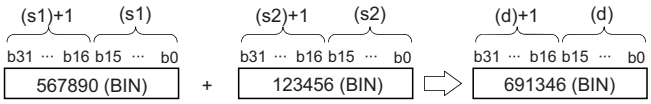
Operand		Description	Range	Data type	Data type (label)
(s1)	D+(P)	First addend data or the start device where the first addend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D+(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	D+(P)	Second addend data or the start device where the second addend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D+(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	D+(P)	Start device for storing the operation result	—	32-bit signed binary	ANY32_S
	D+(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

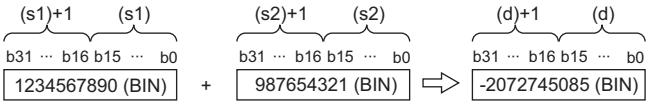
Processing details

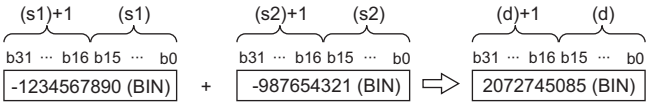
- These instructions add the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



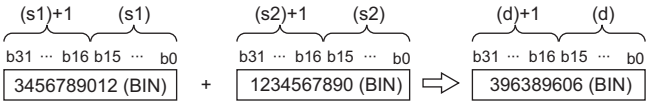
- If an overflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[D+(P) instruction]





[D+(P)_U instruction]



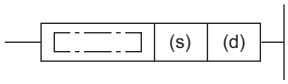
Operation error

There is no operation error.

Subtracting 32-bit binary data

D-(P)(_U) [when two operands are set]

These instructions perform subtraction between the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 278 D-(P)(_U) [when three operands are set])

FBD/LD
Not supported (☞ Page 278 D-(P)(_U) [when three operands are set])

■Execution condition

Instruction	Execution condition
D- D-_U	
D-P D-P_U	

Setting data

■Description, range, data type

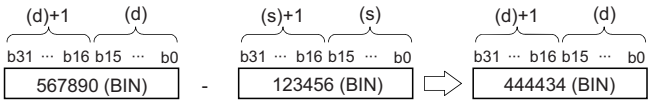
Operand		Description	Range	Data type	Data type (label)
(s)	D-(P)	Subtrahend data or the start device where subtrahend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D-(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	D-(P)	Start device where minuend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D-(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

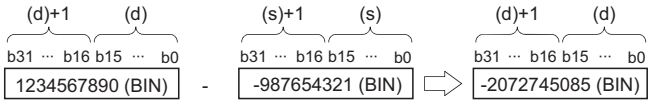
Processing details

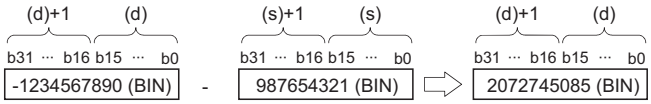
- These instructions subtract the 32-bit binary data in the device specified by (s) from the 32-bit binary data in the device specified by (d) and, and store the operation result in the device specified by (d).



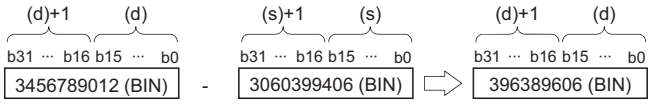
- If an underflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[D-(P) instruction]





[D-(P)_U instruction]

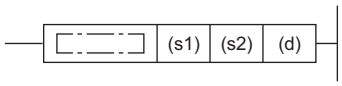


Operation error

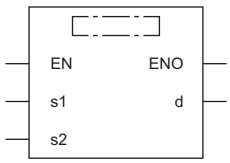
There is no operation error.

D-(P)(_U) [when three operands are set]

These instructions perform subtraction between the two sets of 32-bit binary data specified.

Ladder	ST
	ENO:=DMINUS(EN,s1,s2,d); ENO:=DMINUSP(EN,s1,s2,d); ENO:=DMINUS_U(EN,s1,s2,d); ENO:=DMINUSP_U(EN,s1,s2,d);

FBD/LD



(☐ is to be replaced by any of the following: DMINUS, DMINUSP, DMINUS_U, DMINUSP_U.)

■Execution condition

Instruction	Execution condition
D- D-_U	
D-P D-P_U	

Setting data

■Description, range, data type

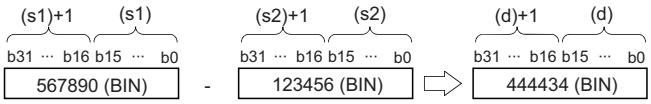
Operand		Description	Range	Data type	Data type (label)
(s1)	D-(P)	Minuend data or the start device where minuend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D-(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	D-(P)	Subtrahend data or the start device where subtrahend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D-(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	D-(P)	Start device for storing the operation result	—	32-bit signed binary	ANY32_S
	D-(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

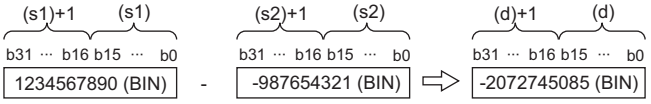
Processing details

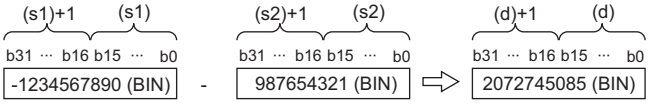
- These instructions subtracts the 32-bit binary data in the device specified by (s2) from the 32-bit binary data in the device specified by (s1), and store the operation result in the device specified by (d).



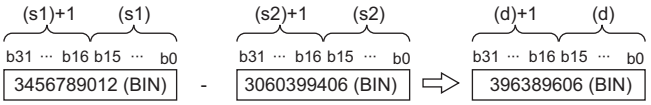
- If an underflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[D-(P) instruction]





[D-(P)_U instruction]



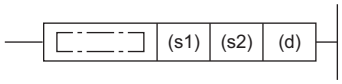
Operation error

There is no operation error.

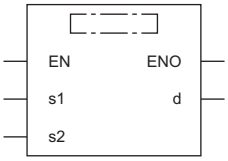
Multiplying 16-bit binary data

*(P)(_U)

These instructions multiply the two sets of 16-bit binary data specified.

Ladder	ST	
	ENO:=MULTI(EN,s1,s2,d); ENO:=MULTIP(EN,s1,s2,d);	ENO:=MULTI_U(EN,s1,s2,d); ENO:=MULTIP_U(EN,s1,s2,d);

FBD/LD



(☐ is to be replaced by any of the following: MULTI, MULTIP, MULTI_U, MULTIP_U.)

■Execution condition

Instruction	Execution condition
* *_U	
*P *P_U	

Setting data

■Description, range, data type

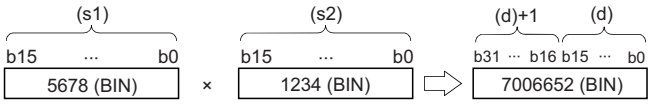
Operand		Description	Range	Data type	Data type (label)
(s1)	*(P)	Multiplicand data or the device where multiplicand data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	*(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	*(P)	Multiplier data or the device where multiplier data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	*(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	*(P)	Start device for storing the operation result	—	32-bit signed binary	ANY32_S
	*(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions multiply the 16-bit binary data in the device specified by (s1) by the 16-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



- When (d) is a bit device, data should be specified in order from lower bits.

Ex.

Operation result when (d) is a bit device

- K1···Lower 4 bits (b0 to b3)
- K4···Lower 16 bits (b0 to b15)
- K8···Lower 32 bits (b0 to b31)

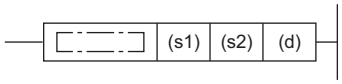
Operation error

There is no operation error.

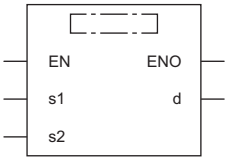
Dividing 16-bit binary data

/(P)(_U)

These instructions perform division between the two sets of 16-bit binary data specified.

Ladder	ST	
	ENO:=DIVISION(EN,s1,s2,d); ENO:=DIVISIONP(EN,s1,s2,d);	ENO:=DIVISION_U(EN,s1,s2,d); ENO:=DIVISIONP_U(EN,s1,s2,d);

FBD/LD



(☐ is to be replaced by any of the following: DIVISION, DIVISIONP, DIVISION_U, DIVISIONP_U.)

■Execution condition

Instruction	Execution condition
/ /_U	
/P /P_U	

Setting data

■Description, range, data type

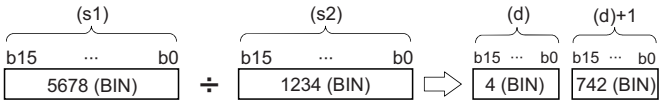
Operand		Description	Range	Data type	Data type (label)
(s1)	/(P)	Dividend data or the device where dividend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	/(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	/(P)	Divisor data or the device where divisor data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	/(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	/(P)	Start device for storing the operation result	—	32-bit signed binary	ANY16_S_ARRAY (Number of elements: 2)
	/(P)_U			32-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions divide the 16-bit binary data in the device specified by (s1) by the 16-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



(d): Quotient
(d)+1: Remainder

- As the operation result, the quotient and remainder are stored in 32 bits. When a bit device is specified, the number of digit-specified bits is used to store the quotient and remainder.
  - Quotient···Stored in lower 16 bits.
  - Remainder···Stored in upper 16 bits.

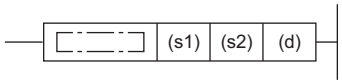
Operation error

Error code (SD0)	Description
3280H	The value (divisor) in the device specified by (s2) is 0.

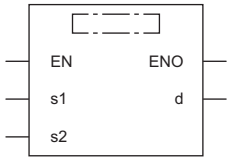
Multiplying 32-bit binary data

D*(P)(_U)

These instructions multiply the two sets of 32-bit binary data specified.

Ladder	ST	
	ENO:=DMULTI(EN,s1,s2,d); ENO:=DMULTIP(EN,s1,s2,d);	ENO:=DMULTI_U(EN,s1,s2,d); ENO:=DMULTIP_U(EN,s1,s2,d);

FBD/LD



(☐ is to be replaced by any of the following: DMULTI, DMULTIP, DMULTI_U, DMULTIP_U.)

■Execution condition

Instruction	Execution condition
D* D*_U	
D*P D*P_U	

Setting data

■Description, range, data type

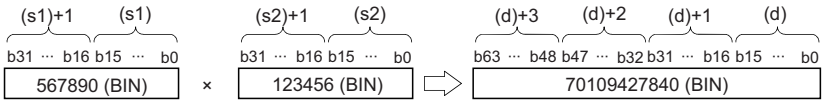
Operand		Description	Range	Data type	Data type (label)
(s1)	D*(P)	Multiplicand data or the start device where multiplicand data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D*(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	D*(P)	Multiplier data or the start device where multiplier data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D*(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	D*(P)	Start device for storing the operation result	—	64-bit signed binary	ANY32_S_ARRAY (Number of elements: 2)
	D*(P)_U			64-bit unsigned binary	ANY32_U_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions multiply the 32-bit binary data in the device specified by (s1) by the 32-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



- When (d) is a bit device, only the lower 32 bits of the operation result are stored. If the upper 32 bits of the operation result are required, temporarily store the result in a word device, and transfer the data stored in (d)+2 and (d)+3 to the specified bit devices.

Ex.

Operation result when (d) is a bit device

- K1···Lower 4 bits (b0 to b3)
- K4···Lower 16 bits (b0 to b15)
- K8···Lower 32 bits (b0 to b31)

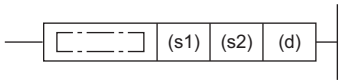
Operation error

There is no operation error.

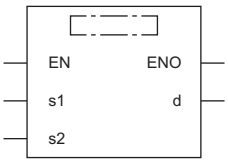
Dividing 32-bit binary data

D/(P)(_U)

These instructions perform division between the two sets of 32-bit binary data specified.

Ladder	ST	
	ENO:=DDIVISION(EN,s1,s2,d); ENO:=DDIVISIONP(EN,s1,s2,d);	ENO:=DDIVISION_U(EN,s1,s2,d); ENO:=DDIVISIONP_U(EN,s1,s2,d);

FBD/LD



(☐ is to be replaced by any of the following: DDIVISION, DDIVISIONP, DDIVISION_U, DDIVISIONP_U.)

■Execution condition

Instruction	Execution condition
D/ D/_U	
D/P D/P_U	

Setting data

■Description, range, data type

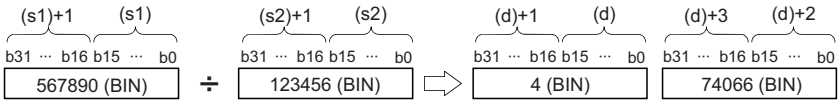
Operand		Description	Range	Data type	Data type (label)
(s1)	D/(P)	Dividend data or the start device where dividend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D/(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	D/(P)	Divisor data or the start device where divisor data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	D/(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	D/(P)	Start device for storing the operation result	—	64-bit signed binary	ANY32_S_ARRAY (Number of elements: 2)
	D/(P)_U			64-bit unsigned binary	ANY32_U_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions divide the 32-bit binary data in the device specified by (s1) by the 32-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



- As the operation result when a word device is specified, the quotient and remainder are stored in 64 bits. The quotient is stored in lower 32 bits, and the remainder is stored in upper 32 bits. When a bit device is specified, only quotient is stored in 32 bits.

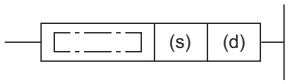
Operation error

Error code (SD0)	Description
3280H	The value (divisor) in the device specified by (s2) is 0.

Adding BCD 4-digit data

B+(P) [when two operands are set]

These instructions add the two sets of BCD 4-digit data specified.

Ladder	ST
	Not supported (☞ Page 289 B+(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 289 B+(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
B+	
B+P	

Setting data

■Description, range, data type

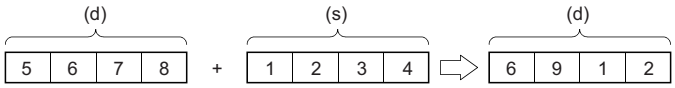
Operand	Description	Range	Data type	Data type (label)
(s)	Second addend data or the device where the second addend data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Device where the first addend data is stored	0 to 9999	BCD 4-digit	ANY16

■Applicable devices

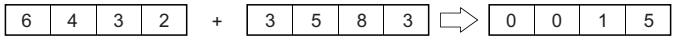
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions add the BCD 4-digit data in the device specified by (d) and the BCD 4-digit data in the device specified by (s), and store the operation result in the device specified by (d).



- If the result exceeds 9999, the carry bit is ignored. In this case, SM700 does not turn on.

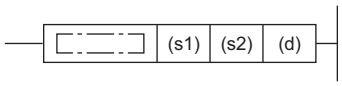


Operation error

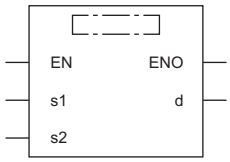
Error code (SD0)	Description
3285H	The BCD data in the device specified by (s) is out of the range, 0 to 9999.
	The BCD data in the device specified by (d) is out of the range, 0 to 9999.

B+(P) [when three operands are set]

These instructions add the two sets of BCD 4-digit data specified.

Ladder	ST
	ENO:=BPLUS(EN,s1,s2,d); ENO:=BPLUSP(EN,s1,s2,d);

FBD/LD



(☐ is to be replaced by either of the following: BPLUS, BPLUSP.)

■Execution condition

Instruction	Execution condition
B+	
B+P	

Setting data

■Description, range, data type

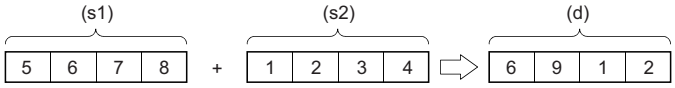
Operand	Description	Range	Data type	Data type (label)
(s1)	First addend data or the device where the first addend data is stored	0 to 9999	BCD 4-digit	ANY16
(s2)	Second addend data or the device where the second addend data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Device for storing the operation result	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

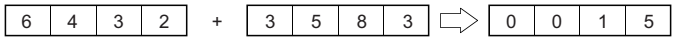
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions add the BCD 4-digit data in the device specified by (s1) and the BCD 4-digit data in the device specified by (s2), and store the operation result in the device specified by (d).



- If the result exceeds 9999, the carry bit is ignored. In this case, SM700 does not turn on.



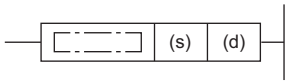
Operation error

Error code (SD0)	Description
3285H	The BCD data in the device specified by (s1) is out of the range, 0 to 9999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 9999.

Subtracting BCD 4-digit data

B-(P) [when two operands are set]

These instructions perform subtraction between the two sets of BCD 4-digit data specified.

Ladder	ST
	Not supported (☞ Page 292 B-(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 292 B-(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
B-	
B-P	

Setting data

■Description, range, data type

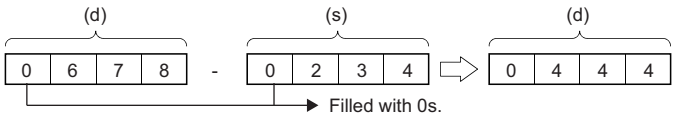
Operand	Description	Range	Data type	Data type (label)
(s)	Subtrahend data or the device where subtrahend data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Device where minuend data is stored	0 to 9999	BCD 4-digit	ANY16

■Applicable devices

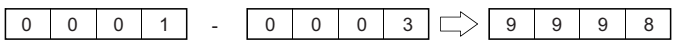
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions subtract the BCD 4-digit data in the device specified by (s) from the 32-bit binary data in the device specified by (d), and store the operation result in the device specified by (d).



- If an underflow occurs, the result will be as follows. In this case, SM700 does not turn on.

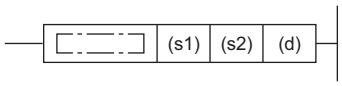


Operation error

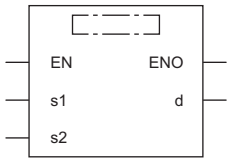
Error code (SD0)	Description
3285H	The BCD data in the device specified by (s) is out of the range, 0 to 9999.
	The BCD data in the device specified by (d) is out of the range, 0 to 9999.

B-(P) [when three operands are set]

These instructions perform subtraction between the two sets of BCD 4-digit data specified.

Ladder	ST
	ENO:=BMINUS(EN,s1,s2,d); ENO:=BMINUSP(EN,s1,s2,d);

FBD/LD



(☐ is to be replaced by either of the following: BMINUS, BMINUSP.)

■Execution condition

Instruction	Execution condition
B-	
B-P	

Setting data

■Description, range, data type

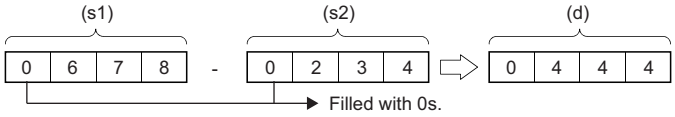
Operand	Description	Range	Data type	Data type (label)
(s1)	Minuend data or the device where minuend data is stored	0 to 9999	BCD 4-digit	ANY16
(s2)	Subtrahend data or the device where subtrahend data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Device for storing the operation result	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

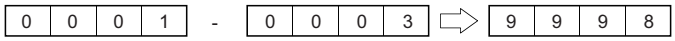
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions subtract the BCD 4-digit data in the device specified by (s2) from the BCD 4-digit data in the device specified by (s1), and store the operation result in the device specified by (d).



- If an underflow occurs, the result will be as follows. In this case, SM700 does not turn on.



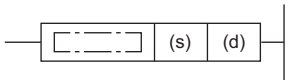
Operation error

Error code (SD0)	Description
3285H	The BCD data in the device specified by (s1) is out of the range, 0 to 9999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 9999.

Adding BCD 8-digit data

DB+(P) [when two operands are set]

These instructions add the two sets of BCD 8-digit data specified.

Ladder	ST
	Not supported (☞ Page 295 DB+(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 295 DB+(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
DB+	
DB+P	

Setting data

■Description, range, data type

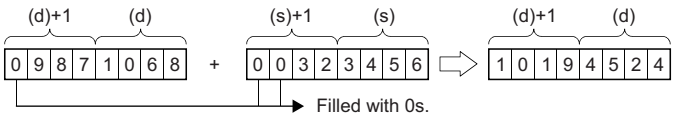
Operand	Description	Range	Data type	Data type (label)
(s)	Second addend data or the start device where the second addend data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device where the first addend data is stored	0 to 99999999	BCD 8-digit	ANY32

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

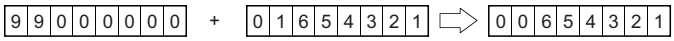
Processing details

- These instructions add the BCD 8-digit data in the device specified by (d) and the BCD 8-digit data in the device specified by (s), and store the operation result in the device specified by (d).



(d)+1, (s)+1: Upper 4 digits
(d), (s): Lower 4 digits

- If the result exceeds 99999999, the carry bit is ignored. In this case, SM700 does not turn on.

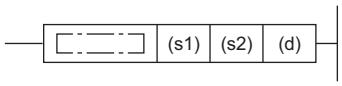


Operation error

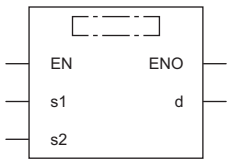
Error code (SD0)	Description
3285H	The BCD data in the device specified by (s) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (d) is out of the range, 0 to 99999999.

DB+(P) [when three operands are set]

These instructions add the two sets of BCD 8-digit data specified.

Ladder	ST
	ENO:=DBPLUS(EN,s1,s2,d); ENO:=DBPLUSP(EN,s1,s2,d);

FBD/LD



(☐ is to be replaced by either of the following: DBPLUS, DBPLUSP.)

■Execution condition

Instruction	Execution condition
DB+	
DB+P	

Setting data

■Description, range, data type

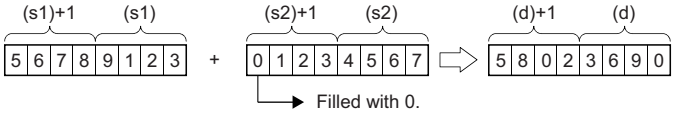
Operand	Description	Range	Data type	Data type (label)
(s1)	First addend data or the start device where the first addend data is stored	0 to 99999999	BCD 8-digit	ANY32
(s2)	Second addend data or the start device where the second addend data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the operation result	—	BCD 8-digit	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

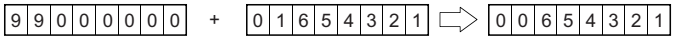
Processing details

- These instructions add the BCD 8-digit data in the device specified by (s1) and the BCD 8-digit data in the device specified by (s2), and store the operation result in the device specified by (d).



(d)+1, (s1)+1, (s2)+1: Upper 4 digits
(d), (s1), (s2): Lower 4 digits

- If the result exceeds 99999999, the carry bit is ignored. In this case, SM700 does not turn on.



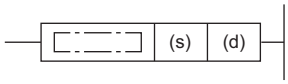
Operation error

Error code (SD0)	Description
3285H	The BCD data in the device specified by (s1) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 99999999.

Subtracting BCD 8-digit data

DB-(P) [when two operands are set]

These instructions perform subtraction between the two sets of BCD 8-digit data specified.

Ladder	ST
	Not supported (☞ Page 298 DB-(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 298 DB-(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
DB-	
DB-P	

Setting data

■Description, range, data type

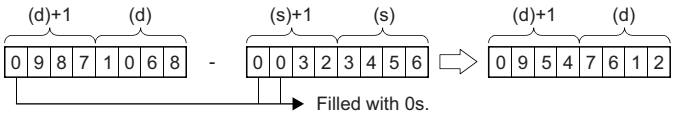
Operand	Description	Range	Data type	Data type (label)
(s)	Subtrahend data or the start device where subtrahend data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Minuend data or the start device where minuend data is stored	0 to 99999999	BCD 8-digit	ANY32

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

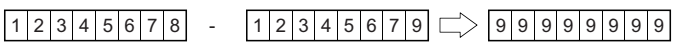
Processing details

- These instructions subtract the BCD 8-digit data in the device specified by (s) from the BCD 8-digit data in the device specified by (d), and store the operation result in the device specified by (d).



(d)+1, (s)+1: Upper 4 digits
(d), (s): Lower 4 digits

- If an underflow occurs, the result will be as follows. In this case, SM700 does not turn on.

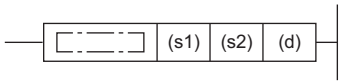


Operation error

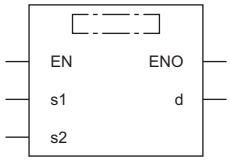
Error code (SD0)	Description
3285H	The BCD data in the device specified by (s) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (d) is out of the range, 0 to 99999999.

DB-(P) [when three operands are set]

These instructions perform subtraction between the two sets of BCD 8-digit data specified.

Ladder	ST
	ENO:=DBMINUS(EN,s1,s2,d); ENO:=DBMINUSP(EN,s1,s2,d);

FBD/LD



(☐ is to be replaced by either of the following: DBMINUS, DBMINUSP.)

■Execution condition

Instruction	Execution condition
DB-	
DB-P	

Setting data

■Description, range, data type

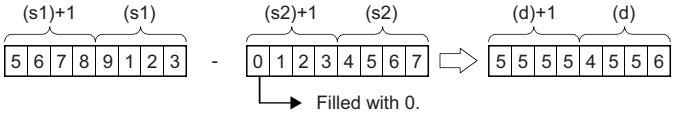
Operand	Description	Range	Data type	Data type (label)
(s1)	Minuend data or the start device where minuend data is stored	0 to 99999999	BCD 8-digit	ANY32
(s2)	Subtrahend data or the start device where subtrahend data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the operation result	—	BCD 8-digit	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

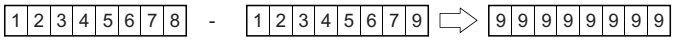
Processing details

- These instructions subtract the BCD 8-digit data in the device specified by (s2) from the BCD 8-digit data in the device specified by (s1), and store the operation result in the device specified by (d).



(d)+1, (s1)+1, (s2)+1: Upper 4 digits
(d), (s1), (s2): Lower 4 digits

- If an underflow occurs, the result will be as follows. In this case, SM700 does not turn on.



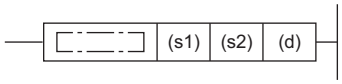
Operation error

Error code (SD0)	Description
3285H	The BCD data in the device specified by (s1) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 99999999.

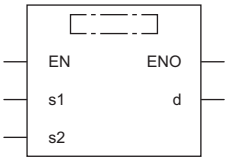
Multiplying BCD 4-digit data

B*(P)

These instructions multiply the two sets of BCD 4-digit data specified.

Ladder	ST
	Not supported

FBD/LD



(☐ is to be replaced by either of the following: BMULTI, BMULTIP.)

■Execution condition

Instruction	Execution condition
B*	
B*P	

Setting data

■Description, range, data type

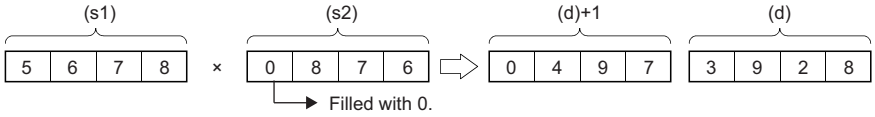
Operand	Description	Range	Data type	Data type (label)
(s1)	Multiplicand data or the device where multiplicand data is stored	0 to 9999	BCD 4-digit	ANY16
(s2)	Multiplier data or the device where multiplier data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the operation result	—	BCD 8-digit	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions multiply the BCD 4-digit data in the device specified by (s1) by the BCD 4-digit data in the device specified by (s2), and store the operation result in the device specified by (d). ((d)+1: Upper 4 digits, (d): Lower 4 digits)



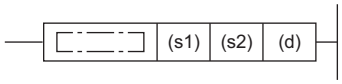
Operation error

Error code (SD0)	Description
3285H	The BCD data in the device specified by (s1) is out of the range, 0 to 9999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 9999.

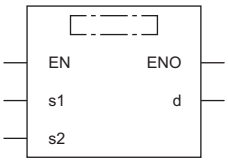
Dividing BCD 4-digit data

B/(P)

These instructions perform division between the two sets of BCD 4-digit data specified.

Ladder	ST
	Not supported

FBD/LD



(☐ is to be replaced by either of the following: BDIVISION, BDIVISIONP.)

■Execution condition

Instruction	Execution condition
B/	
B/P	

Setting data

■Description, range, data type

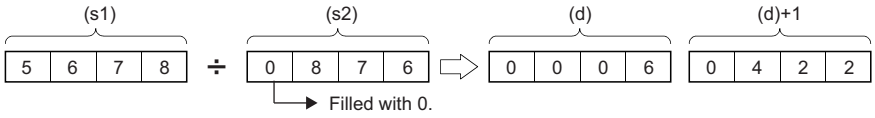
Operand	Description	Range	Data type	Data type (label)
(s1)	Dividend data or the device where dividend data is stored	0 to 9999	BCD 4-digit	ANY16
(s2)	Divisor data or the device where divisor data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the operation result	—	BCD 8-digit	ANY16_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions divide the BCD 4-digit data in the device specified by (s1) by the BCD 4-digit data in the device specified by (s2), and store the operation result in the device specified by (d).



(d): Quotient

(d)+1: Remainder

- As the operation result, the quotient and remainder are stored in 32 bits.
  - Quotient (BCD 4 digits)···Stored in lower 16 bits.
  - Remainder (BCD 4 digits)···Stored in upper 16 bits.

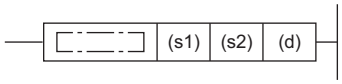
Operation error

Error code (SD0)	Description
3280H	The value (divisor) in the device specified by (s2) is 0.
3285H	The BCD data in the device specified by (s1) is out of the range, 0 to 9999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 9999.

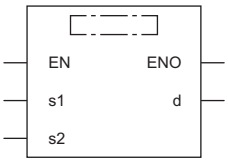
Multiplying BCD 8-digit data

DB*(P)

These instructions multiply the two sets of BCD 8-digit data specified.

Ladder	ST
	Not supported

FBD/LD



(☐ is to be replaced by either of the following: DBMULTI, DBMULTIP.)

■Execution condition

Instruction	Execution condition
DB*	
DB*P	

Setting data

■Description, range, data type

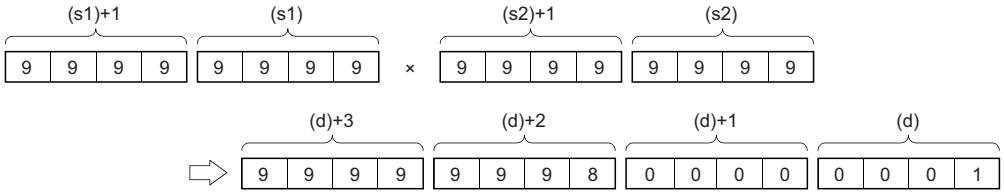
Operand	Description	Range	Data type	Data type (label)
(s1)	Multiplicand data or the start device where multiplicand data is stored	0 to 99999999	BCD 8-digit	ANY32
(s2)	Multiplier data or the start device where multiplier data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the operation result	—	BCD 16-digit	ANY32_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions multiply the BCD 8-digit data in the device specified by (s1) by the BCD 8-digit data in the device specified by (s2), and store the operation result in the device specified by (d).



- When (d) is a bit device, only the lower 8 digits (lower 32 bits) of the operation result are stored.

Ex.

Operation result when (d) is a bit device

- K1···Lower 1 digit (b0 to b3)
- K4···Lower 4 digits (b0 to b15)
- K8···Lower 8 digits (b0 to b31)

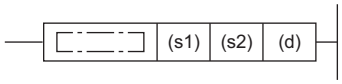
Operation error

Error code (SD0)	Description
3285H	The BCD data in the device specified by (s1) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 99999999.

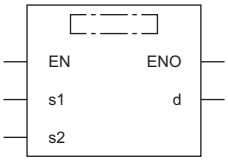
Dividing BCD 8-digit data

DB/(P)

These instructions perform division between the two sets of BCD 8-digit data specified.

Ladder	ST
	Not supported

FBD/LD



(☐ is to be replaced by either of the following: DBDIVISION, DBDIVISIONP.)

■Execution condition

Instruction	Execution condition
DB/	
DB/P	

Setting data

■Description, range, data type

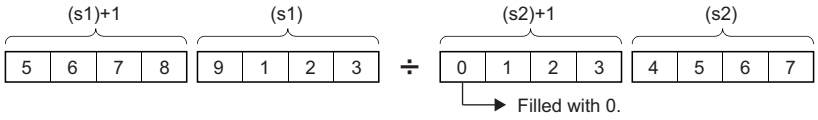
Operand	Description	Range	Data type	Data type (label)
(s1)	Dividend data or the start device where dividend data is stored	0 to 99999999	BCD 8-digit	ANY32
(s2)	Divisor data or the start device where divisor data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the operation result	—	BCD 16-digit	ANY32_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

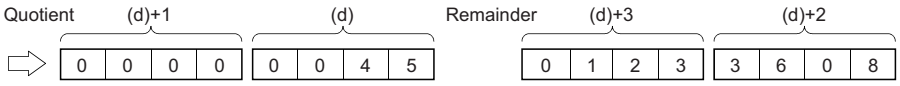
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions divide the BCD 8-digit data in the device specified by (s1) by the BCD 8-digit data in the device specified by (s2), and store the operation result in the device specified by (d).





(d)+1, (d)+3: Upper 4 digits
(d), (d)+2: Lower 4 digits

- As the operation result, the quotient and remainder are stored in 64 bits.
  - Quotient (BCD 8 digits)···Stored in lower 32 bits.
  - Remainder (BCD 8 digits)···Stored in upper 32 bits.
- When (d) is a bit device, the remainder is not stored.

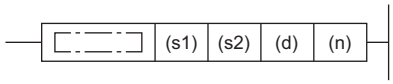
Operation error

Error code (SD0)	Description
3280H	The value (divisor) in the device specified by (s2) is 0.
3285H	The BCD data in the device specified by (s1) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 99999999.

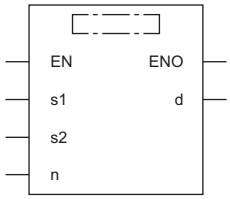
Adding 16-bit binary block data

BK+(P)(_U)

These instructions add the two 16-bit binary data blocks specified.

Ladder	ST	
	ENO:=BKPLUS(EN,s1,s2,n,d); ENO:=BKPLUSP(EN,s1,s2,n,d);	ENO:=BKPLUS_U(EN,s1,s2,n,d); ENO:=BKPLUSP_U(EN,s1,s2,n,d);

FBD/LD



(☐ is to be replaced by any of the following: BKPLUS, BKPLUSP, BKPLUS_U, BKPLUSP_U.)

■Execution condition

Instruction	Execution condition
BK+ BK+_U	
BK+P BK+P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	BK+(P)	First addend data or the start device where the first addend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S*1
	BK+(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U*1
(s2)	BK+(P)	Second addend data or the start device where the second addend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S*1
	BK+(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U*1
(d)	BK+(P)	Start device for storing the operation result	—	16-bit signed binary	ANY16_S*1
	BK+(P)_U			16-bit unsigned binary	ANY16_U*1
(n)		Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

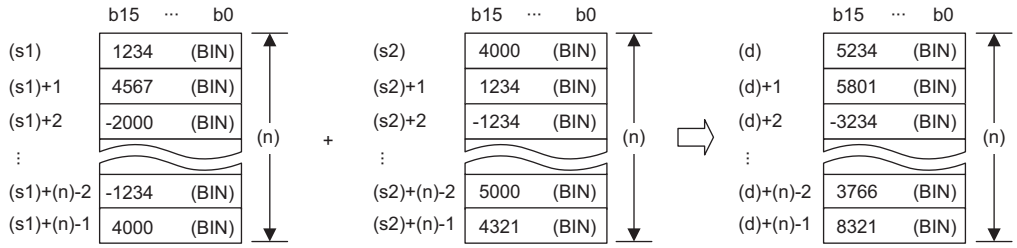
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

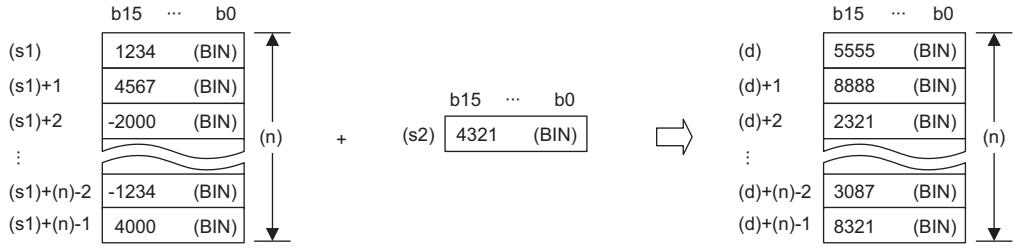
- These instructions add the (n) points of 16-bit binary data from the device specified by (s1) and the (n) points of 16-bit binary data from the device specified by (s2) or the constant, and store the operation result in the device specified by (d) and later.
- Specify data in units of 16 bits.

Ex.

When a device is specified by (s2) (signed value specification)



When a constant is specified by (s2) (signed value specification)



- If an overflow occurs, the result will be as follows. In this case, SM700 does not turn on.

When a signed value is specified	When an unsigned value is specified
K32767 (H7FFF) + K2 (H0002) ⇨ K-32767 (H8001) K-32767 (H8001) + K-2 (HFFFE) ⇨ K32767 (H7FFF)	K65535 (HFFFF) + K1 (H0001) ⇨ K0 (H0000)

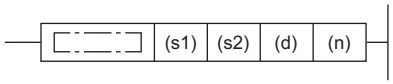
Operation error

Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are overlapping (except when the same device is specified for (s2) and (d)).

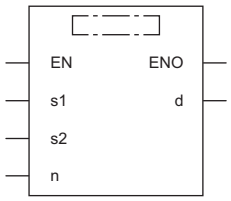
Subtracting 16-bit binary block data

BK-(P)(_U)

These instructions perform subtraction between the two 16-bit binary data blocks specified.

Ladder	ST	
	ENO:=BKMINUS(EN,s1,s2,n,d); ENO:=BKMINUSP(EN,s1,s2,n,d);	ENO:=BKMINUS_U(EN,s1,s2,n,d); ENO:=BKMINUSP_U(EN,s1,s2,n,d);

FBD/LD



(☐ is to be replaced by any of the following: BKMINUS, BKMINUSP, BKMINUS_U, BKMINUSP_U.)

■Execution condition

Instruction	Execution condition
BK- BK-_U	
BK-P BK-P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	BK-(P)	Minuend data or the start device where minuend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S*1
	BK-(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U*1
(s2)	BK-(P)	Subtrahend data or the start device where subtrahend data is stored	-32768 to 32767	16-bit signed binary	ANY16_S*1
	BK-(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U*1
(d)	BK-(P)	Start device for storing the operation result	—	16-bit signed binary	ANY16_S*1
	BK-(P)_U			16-bit unsigned binary	ANY16_U*1
(n)		Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

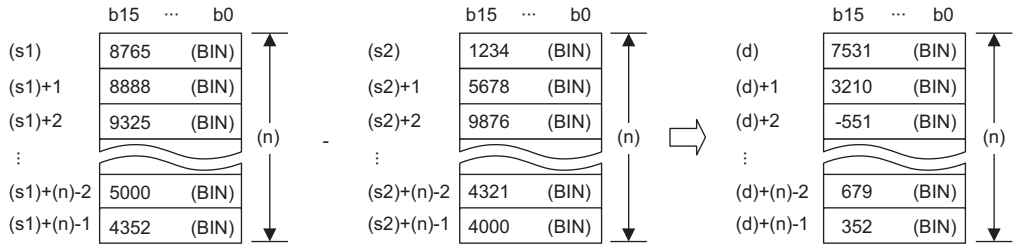
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

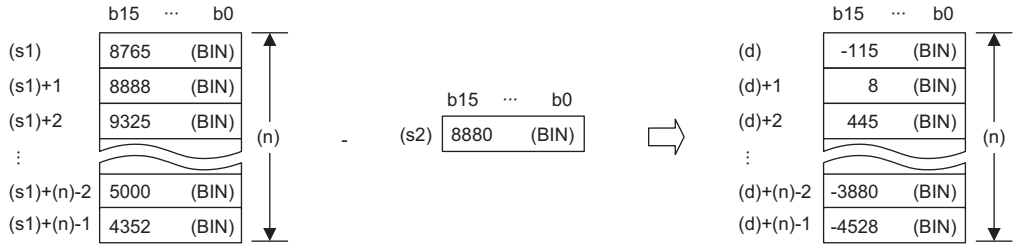
- These instructions subtract the (n) points of 16-bit binary data from the device specified by (s2) or the constant from the (n) points of 16-bit binary data from the device specified by (s1), and store the operation result in the device specified by (d) and later.
- Specify data in units of 16 bits.

Ex.

When a device is specified by (s2)



When a constant is specified by (s2)



- If an overflow occurs, the result will be as follows. In this case, SM700 does not turn on.

When a signed value is specified	When an unsigned value is specified
K-32768 (H8000) - K2 (H0002) ⇨ K32766 (H7FFE) K32767 (H7FFF) - K-2 (HFFFE) ⇨ K-32767 (H8001)	K0 (H0000) - K1 (H0001) ⇨ K65535 (HFFFF)

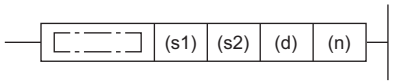
Operation error

Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are overlapping (except when the same device is specified for (s2) and (d)).

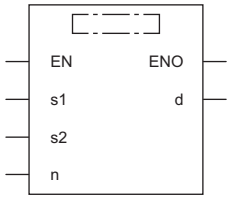
Adding 32-bit binary block data

DBK+(P)(_U)

These instructions add the two 32-bit binary data blocks specified.

Ladder	ST	
	ENO:=DBKPLUS(EN,s1,s2,n,d); ENO:=DBKPLUSP(EN,s1,s2,n,d);	ENO:=DBKPLUS_U(EN,s1,s2,n,d); ENO:=DBKPLUSP_U(EN,s1,s2,n,d);

FBD/LD



(☐ is to be replaced by any of the following: DBKPLUS, DBKPLUSP, DBKPLUS_U, DBKPLUSP_U.)

■Execution condition

Instruction	Execution condition
DBK+ DBK+_U	
DBK+P DBK+P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DBK+(P)	First addend data or the start device where the first addend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S*1
	DBK+(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U*1
(s2)	DBK+(P)	Second addend data or the start device where the second addend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S*1
	DBK+(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U*1
(d)	DBK+(P)	Start device for storing the operation result	—	32-bit signed binary	ANY32_S*1
	DBK+(P)_U			32-bit unsigned binary	ANY32_U*1
(n)		Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

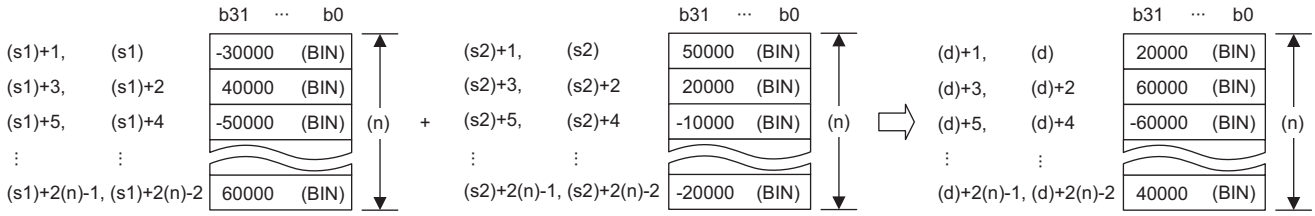
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

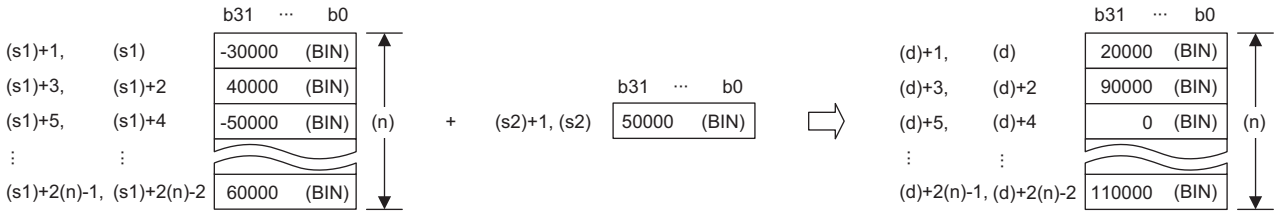
- These instructions add the (n) points of 32-bit binary data from the device specified by (s1) and the (n) points of 32-bit binary data from the device specified by (s2) or the constant, and store the operation result in the device specified by (d) and later.
- Specify data in units of 32 bits.

Ex.

When a device is specified by (s2) (signed value specification)



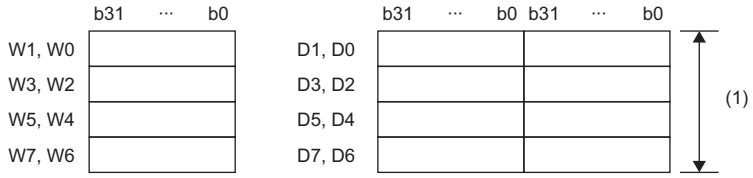
When a constant is specified by (s2) (signed value specification)



- Operation is possible when the same device is specified by (s1) or (s2) and (d). However, if the device range of (n) points from (s1) or (s2) and the device range of (n) points from (d) are partly overlapped, an error results.

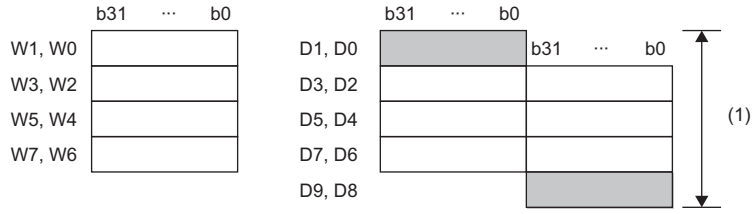
Ex.

When the four points of device from that specified by (s2) and (d) exactly match



(1) Operation is possible because they exactly match.

When four points of device from that specified by (s2) and (d) are partly overlapped



(1) An operation error results because they partly match.

- If (n) is 0, no processing is performed.
- If an overflow occurs, the result will be as follows. In this case, SM700 does not turn on.

When a signed value is specified	When an unsigned value is specified
K2147483647 (H7FFFFFFF) + K2 (H00000002) ⇨ K-2147483647 (H80000001) K-2147483647 (H80000001) + K-2 (HFFFFFFFE) ⇨ K2147483647 (H7FFFFFFF)	K4294967295 (HFFFFFFFF) + K1 (H00000001) ⇨ K0 (H00000000)

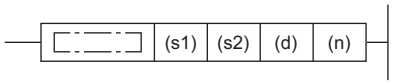
Operation error

Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are overlapping (except when the same device is specified for (s2) and (d)).

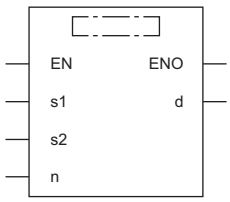
Subtracting 32-bit binary block data

DBK-(P)(_U)

These instructions perform subtraction between the two 32-bit binary data blocks specified.

Ladder	ST	
	ENO:=DBKMINUS(EN,s1,s2,n,d); ENO:=DBKMINUSP(EN,s1,s2,n,d);	ENO:=DBKMINUS_U(EN,s1,s2,n,d); ENO:=DBKMINUSP_U(EN,s1,s2,n,d);

FBD/LD



(☐ is to be replaced by any of the following: DBKMINUS, DBKMINUSP, DBKMINUS_U, DBKMINUSP_U.)

■Execution condition

Instruction	Execution condition
DBK- DBK-_U	
DBK-P DBK-P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DBK-(P)	Minuend data or the start device where minuend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S*1
	DBK-(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U*1
(s2)	DBK-(P)	Subtrahend data or the start device where subtrahend data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S*1
	DBK-(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U*1
(d)	DBK-(P)	Start device for storing the operation result	—	32-bit signed binary	ANY32_S*1
	DBK-(P)_U			32-bit unsigned binary	ANY32_U*1
(n)		Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

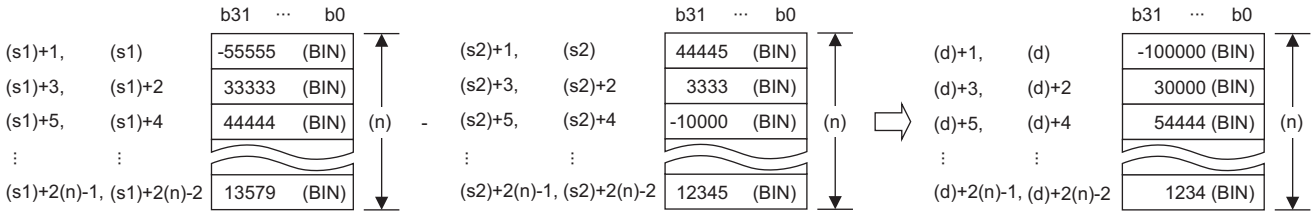
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

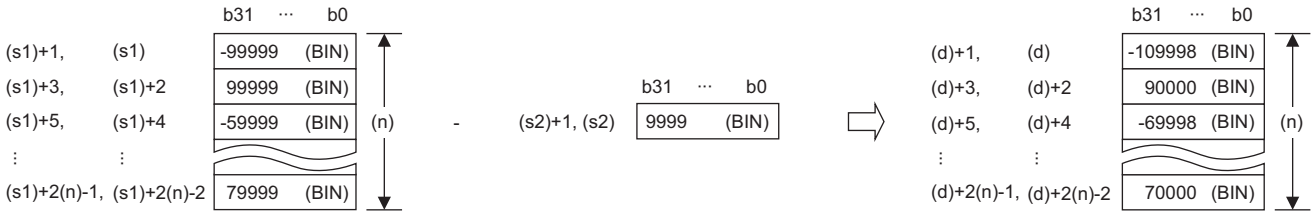
- These instructions subtract the (n) points of 32-bit binary data from the device specified by (s2) or the constant from the (n) points of 32-bit binary data from the device specified by (s1), and store the operation result in the device specified by (d) and later.
- Specify data in units of 32 bits.

Ex.

When a device is specified by (s2) (signed value specification)



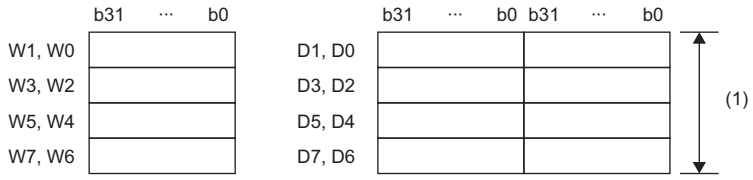
When a constant is specified by (s2) (signed value specification)



- Operation is possible when the same device is specified by (s1) or (s2) and (d). However, if the device range of (n) points from (s1) or (s2) and the device range of (n) points from (d) are partly overlapped, an error results.

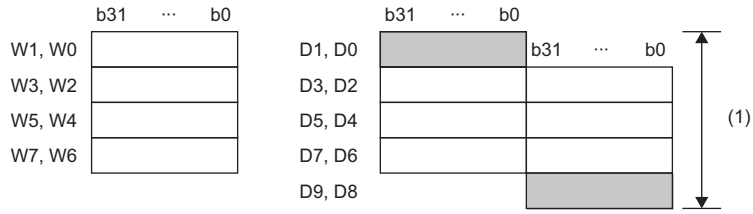
Ex.

When the four points of device from that specified by (s2) and (d) exactly match



(1) Operation is possible because they exactly match.

When four points of device from that specified by (s2) and (d) are partly overlapped



(1) An operation error results because they partly match.

- If (n) is 0, no processing is performed.
- If an overflow occurs, the result will be as follows. In this case, SM700 does not turn on.

When a signed value is specified	When an unsigned value is specified
K2147483647 (H7FFFFFFF) - K-2 (HFFFFFFFE) → K-2147483647 (H80000001) K-2147483647 (H80000001) - K2 (H00000002) → K2147483647 (H7FFFFFFF)	K0 (H00000000) - K1 (H00000001) → K4294967295 (HFFFFFFFF)

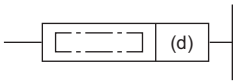
Operation error

Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are overlapping (except when the same device is specified for (s2) and (d)).

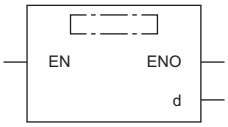
Incrementing 16-bit binary data

INC(P)(_U)

These instructions increment the specified 16-bit binary data by one.

Ladder	ST	
	ENO:=INC(EN,d); ENO:=INCP(EN,d);	ENO:=INC_U(EN,d); ENO:=INCP_U(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
INC INC_U	
INCP INCP_U	

Setting data

■Description, range, data type

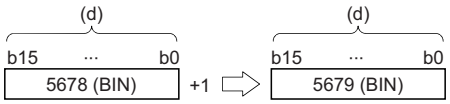
Operand		Description	Range	Data type	Data type (label)
(d)	INC(P)	Increment target device	-32768 to 32767	16-bit signed binary	ANY16_S
	INC(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions increment the 16-bit binary data in the device specified by (d) by one.



- When the INC(P) instruction is executed while the data in the device specified by (d) is 32767, -32768 is stored in the device specified by (d). (When a signed value is specified)
- When the INC(P)_U instruction is executed while the data in the device specified by (d) is 65535, 0 is stored in the device specified by (d). (When an unsigned value is specified)

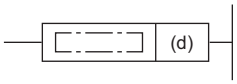
Operation error

There is no operation error.

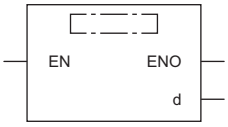
Decrementing 16-bit binary data

DEC(P)(_U)

These instructions decrement the specified 16-bit binary data by one.

Ladder	ST	
	ENO:=DEC(EN,d); ENO:=DECP(EN,d);	ENO:=DEC_U(EN,d); ENO:=DECP_U(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DEC DEC_U	
DECP DECP_U	

Setting data

■Description, range, data type

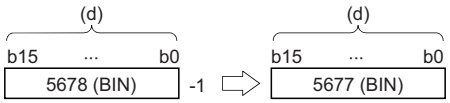
Operand		Description	Range	Data type	Data type (label)
(d)	DEC(P)	Decrement target device	-32768 to 32767	16-bit signed binary	ANY16_S
	DEC(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions decrement the 16-bit binary data in the device specified by (d) by one.



- When the DEC(P) instruction is executed while the data in the device specified by (d) is -32768, 32767 is stored in the device specified by (d). (When a signed value is specified)
- When the DEC(P)_U instruction is executed while the data in the device specified by (d) is 0, 65535 is stored in the device specified by (d). (When an unsigned value is specified)

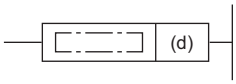
Operation error

There is no operation error.

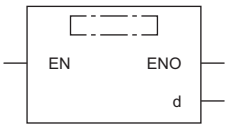
Incrementing 32-bit binary data

DINC(P)(_U)

These instructions increment the specified 32-bit binary data by one.

Ladder	ST	
	ENO:=DINC(EN,d); ENO:=DINCP(EN,d);	ENO:=DINC_U(EN,d); ENO:=DINCP_U(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DINC DINC_U	
DINCP DINCP_U	

Setting data

■Description, range, data type

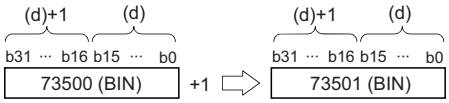
Operand		Description	Range	Data type	Data type (label)
(d)	DINC(P)	Increment target start device	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DINC(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions increment the 32-bit binary data in the device specified by (d) by one.



- When the DINC(P)(_U) instruction is executed while the data in the device specified by (d) is 2147483647, -2147483648 is stored in the device specified by (d). (When a signed value is specified)
- When the DINC(P)(_U) instruction is executed while the data in the device specified by (d) is 4294967295, 0 is stored in the device specified by (d). (When an unsigned value is specified)

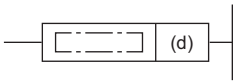
Operation error

There is no operation error.

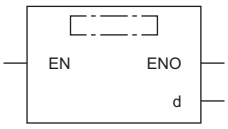
Decrementing 32-bit binary data

DDEC(P)(_U)

These instructions decrement the specified 32-bit binary data by one.

Ladder	ST	
	ENO:=DDEC(EN,d); ENO:=DDECP(EN,d);	ENO:=DDEC_U(EN,d); ENO:=DDECP_U(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DDEC DDEC_U	
DDECP DDECP_U	

Setting data

■Description, range, data type

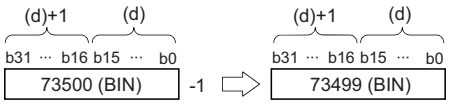
Operand		Description	Range	Data type	Data type (label)
(d)	DDEC(P)	Decrement target start device	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DDEC(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions decrement the 32-bit binary data in the device specified by (d) by one.



- When the DDEC(P)(_U) instruction is executed while the data in the device specified by (d) is -2147483648, 2147483647 is stored in the device specified by (d). (When a signed value is specified)
- When the DDEC(P)(_U) instruction is executed while the data in the device specified by (d) is 0, -1 is stored in the device specified by (d). (When a signed value is specified)
- When the DDEC(P)(_U) instruction is executed while the data in the device specified by (d) is 0, 4294967295 is stored in the device specified by (d). (When an unsigned value is specified)

Operation error

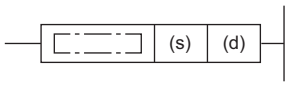
There is no operation error.

12.3 Logical Operation Instructions

Performing an AND operation on 16-bit data

WAND(P) [when two operands are set]

These instructions perform an AND operation on the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 325 WAND(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 325 WAND(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
WAND	
WANDP	

Setting data

■Description, range, data type

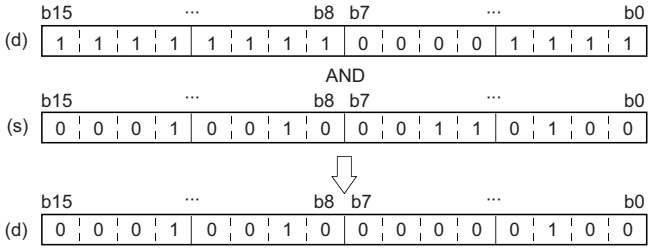
Operand	Description	Range	Data type	Data type (label)
(s)	Logical AND data or the device where logical AND data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	-32768 to 32767	16-bit signed binary	ANY16

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions perform an AND operation (bit-by-bit) on the 16-bit binary data in the device specified by (d) and the 16-bit binary data in the device specified by (s), and store the operation result in the device specified by (d).



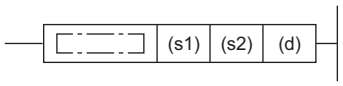
- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

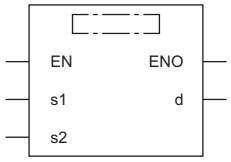
There is no operation error.

WAND(P) [when three operands are set]

These instructions perform an AND operation on the two sets of 16-bit binary data specified.

Ladder	ST
	ENO:=WAND(EN,s1,s2,d); ENO:=WANDP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
WAND	
WANDP	

Setting data

■Description, range, data type

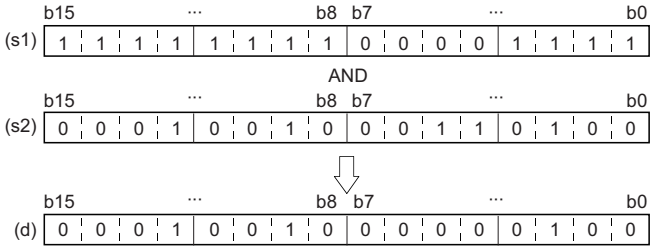
Operand	Description	Range	Data type	Data type (label)
(s1)	Logical AND data or the device where logical AND data is stored	-32768 to 32767	16-bit signed binary	ANY16
(s2)	Logical AND data or the device where logical AND data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions perform an AND operation (bit-by-bit) on the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

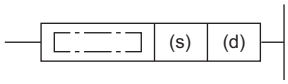
Operation error

There is no operation error.

Performing an AND operation on 32-bit data

DAND(P) [when two operands are set]

These instructions perform an AND operation on the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 329 DAND(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 329 DAND(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
DAND	
DANDP	

Setting data

■Description, range, data type

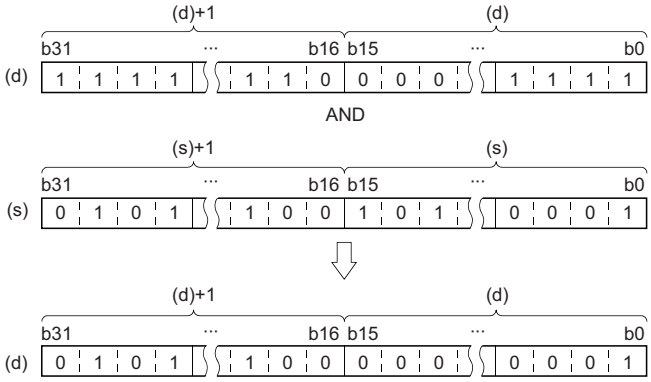
Operand	Description	Range	Data type	Data type (label)
(s)	Logical AND data or the start device where logical AND data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	-2147483648 to 2147483647	32-bit signed binary	ANY32

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions perform an AND operation (bit-by-bit) on the 32-bit binary data in the device specified by (d) and the 32-bit binary data in the device specified by (s), and store the operation result in the device specified by (d).



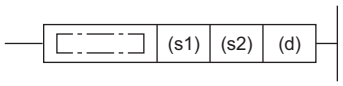
- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

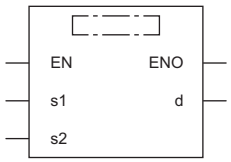
There is no operation error.

DAND(P) [when three operands are set]

These instructions perform an AND operation on the two sets of 32-bit binary data specified.

Ladder	ST
	ENO:=DAND(EN,s1,s2,d); ENO:=DANDP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DAND	
DANDP	

Setting data

■Description, range, data type

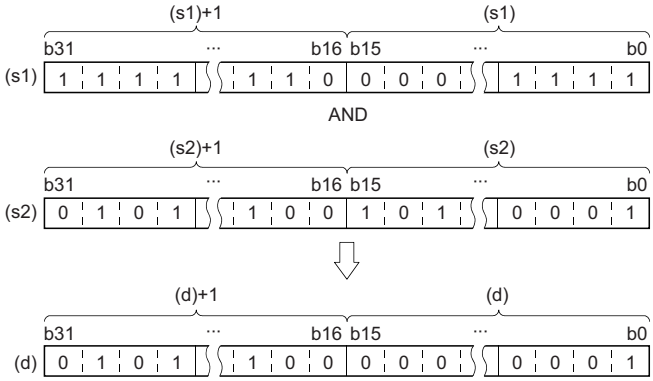
Operand	Description	Range	Data type	Data type (label)
(s1)	Logical AND data or the start device where logical AND data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(s2)	Logical AND data or the start device where logical AND data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions perform an AND operation (bit-by-bit) on the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

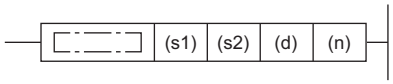
Operation error

There is no operation error.

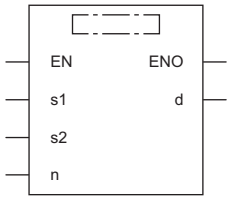
Performing an AND operation on 16-bit block data

BKAND(P)

These instructions perform an AND operation on the two 16-bit binary data blocks specified.

Ladder	ST
	ENO:=BKAND(EN,s1,s2,n,d); ENO:=BKANDP(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BKAND	
BKANDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Logical AND data or the start device where logical AND data is stored	—	16-bit signed binary	ANY16*1
(s2)	Logical AND data or the start device where logical AND data is stored	-32768 to 32767	16-bit signed binary	ANY16*1
(d)	Start device for storing the operation result	—	16-bit signed binary	ANY16*1
(n)	Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

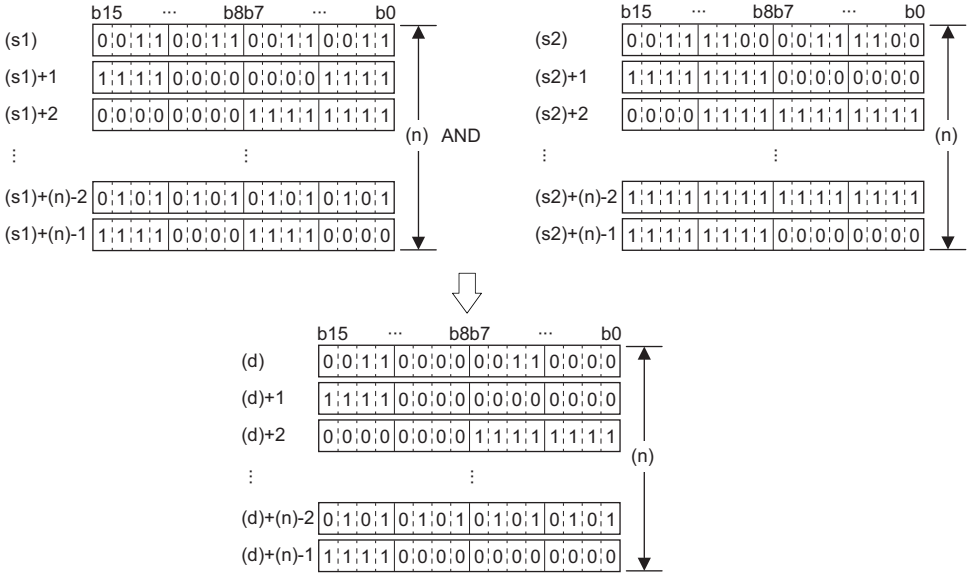
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)*1	—	—	○	—	—	—	—	○	—	—	—	—
(s2)*1	—	—	○	—	—	—	—	○	○	—	—	—
(d)*1	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

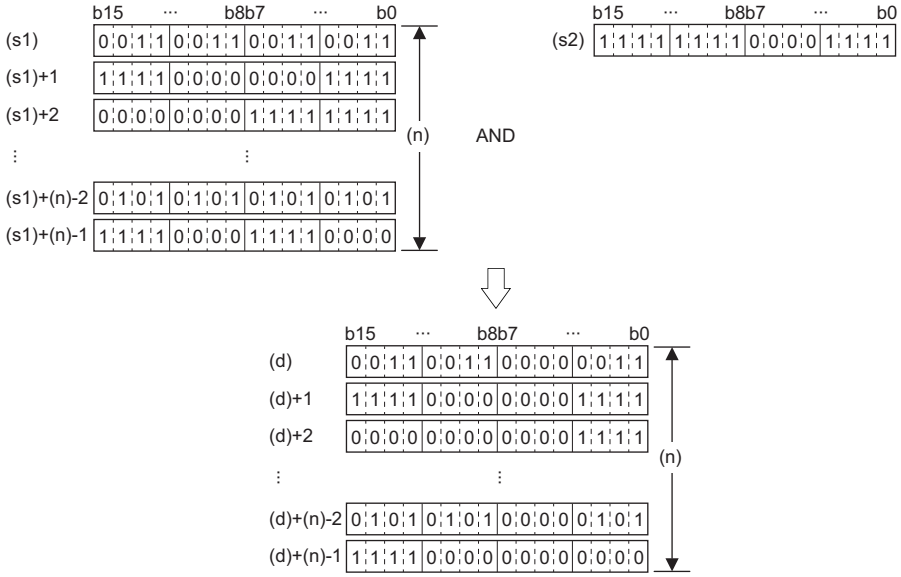
*1 The same device number can be specified for (s1) and (d) or (s2) and (d).

Processing details

- These instructions perform an AND operation on the (n) points of data from the device specified by (s1) and the (n) points of data from the device specified by (s2), and store the operation result in the device specified by (d) and later.



- A constant from -32768 to 32767 (16-bit signed binary) can be specified for (s2).



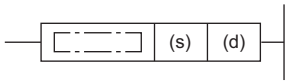
Operation error

Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are partially overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are partially overlapping (except when the same device is specified for (s2) and (d)).

Performing an OR operation on 16-bit data

WOR(P) [when two operands are set]

These instructions perform an OR operation on the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 334 WOR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 334 WOR(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
WOR	
WORP	

Setting data

■Description, range, data type

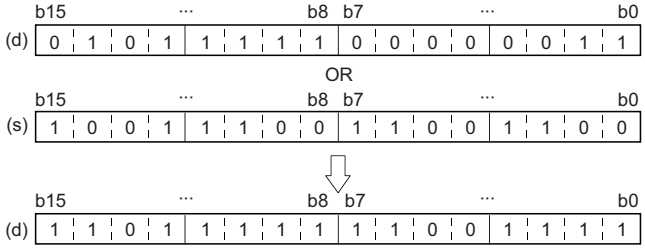
Operand	Description	Range	Data type	Data type (label)
(s)	Logical OR data or the device where logical OR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	-32768 to 32767	16-bit signed binary	ANY16

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions perform an OR operation (bit-by-bit) on the 16-bit binary data in the device specified by (d) and the 16-bit binary data in the device specified by (s), and store the operation result in the device specified by (d).



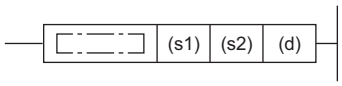
- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

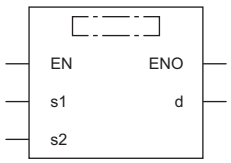
There is no operation error.

WOR(P) [when three operands are set]

These instructions perform an OR operation on the two sets of 16-bit binary data specified.

Ladder	ST
	ENO:=WOR(EN,s1,s2,d); ENO:=WORP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
WOR	
WORP	

Setting data

■Description, range, data type

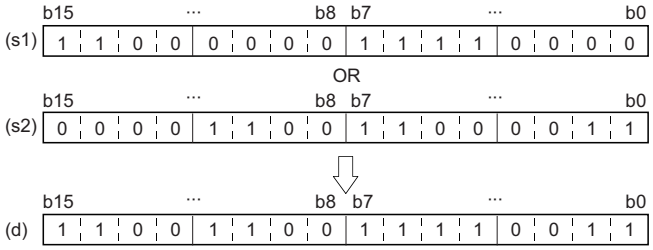
Operand	Description	Range	Data type	Data type (label)
(s1)	Logical OR data or the device where logical OR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(s2)	Logical OR data or the device where logical OR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions perform an OR operation (bit-by-bit) on the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

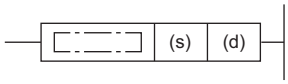
Operation error

There is no operation error.

Performing an OR operation on 32-bit data

DOR(P) [when two operands are set]

These instructions perform an OR operation on the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 338 DOR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 338 DOR(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
DOR	
DORP	

Setting data

■Description, range, data type

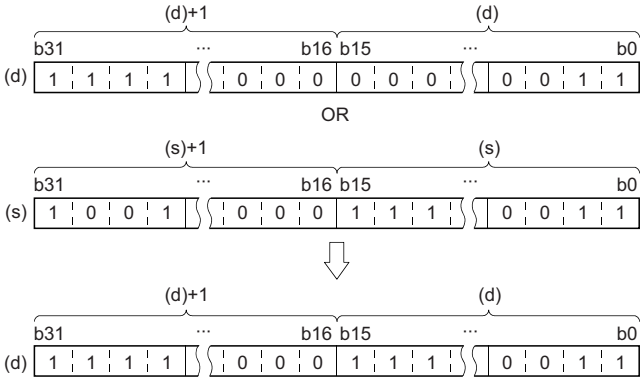
Operand	Description	Range	Data type	Data type (label)
(s)	Logical OR data or the start device where logical OR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	-2147483648 to 2147483647	32-bit signed binary	ANY32

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions perform an OR operation (bit-by-bit) on the 32-bit binary data in the device specified by (d) and the 32-bit binary data in the device specified by (s), and store the operation result in the device specified by (d).



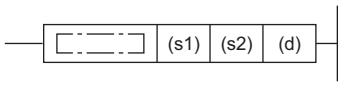
- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

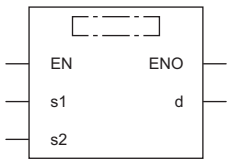
There is no operation error.

DOR(P) [when three operands are set]

These instructions perform an OR operation on the two sets of 32-bit binary data specified.

Ladder	ST
	ENO:=DOR(EN,s1,s2,d); ENO:=DORP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DOR	
DORP	

Setting data

■Description, range, data type

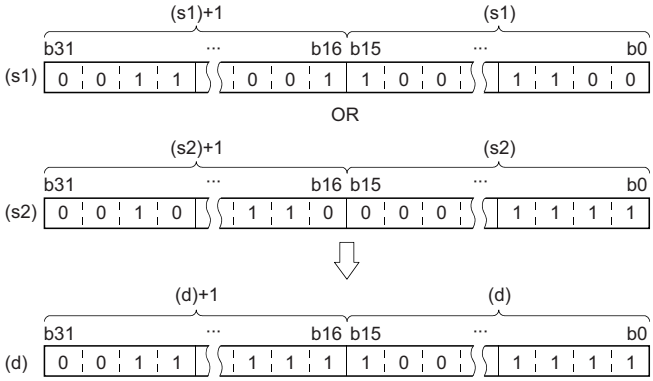
Operand	Description	Range	Data type	Data type (label)
(s1)	Logical OR data or the start device where logical OR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(s2)	Logical OR data or the start device where logical OR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions perform an OR operation (bit-by-bit) on the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

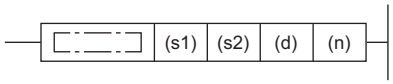
Operation error

There is no operation error.

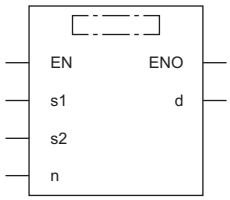
Performing an OR operation on 16-bit block data

BKOR(P)

These instructions perform an OR operation on the two 16-bit binary data blocks specified.

Ladder	ST
	ENO:=BKOR(EN,s1,s2,n,d); ENO:=BKORP(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BKOR	
BKORP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Logical OR data or the start device where logical OR data is stored	—	16-bit signed binary	ANY16*1
(s2)	Logical OR data or the start device where logical OR data is stored	-32768 to 32767	16-bit signed binary	ANY16*1
(d)	Start device for storing the operation result	—	16-bit signed binary	ANY16*1
(n)	Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

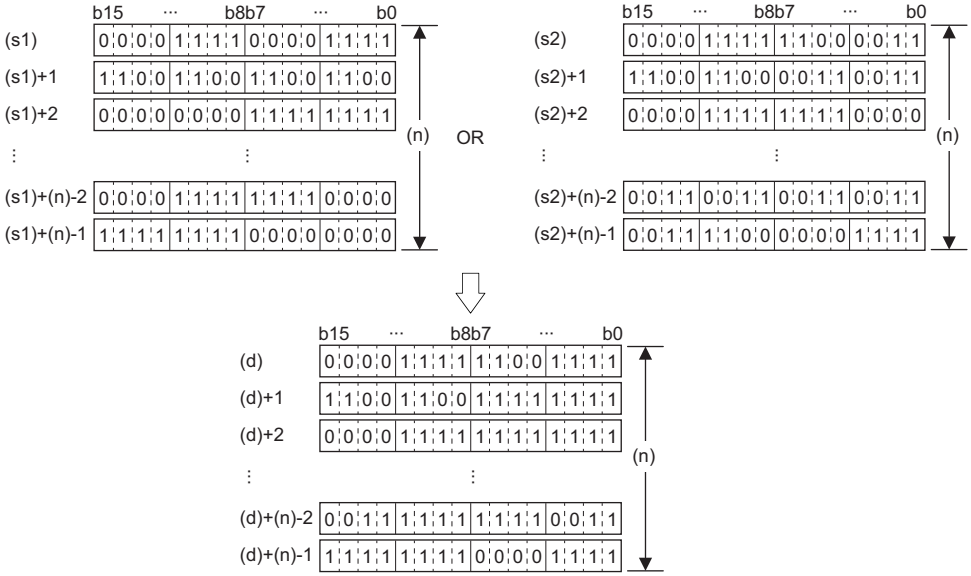
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)*1	—	—	○	—	—	—	—	○	—	—	—	—
(s2)*1	—	—	○	—	—	—	—	○	○	—	—	—
(d)*1	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

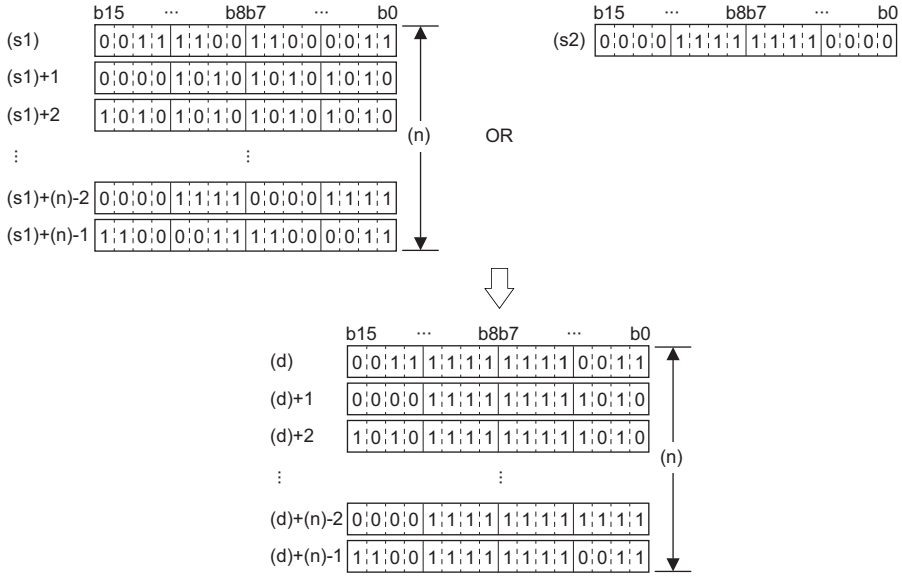
*1 The same device number can be specified for (s1) and (d) or (s2) and (d).

Processing details

- These instructions perform an OR operation on the (n) points of data from the device specified by (s1) and the (n) points of data from the device specified by (s2), and store the operation result in the device specified by (d) and later.



- A constant from -32768 to 32767 (16-bit signed binary) can be specified for (s2).



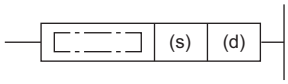
Operation error

Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are partially overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are partially overlapping (except when the same device is specified for (s2) and (d)).

Performing an XOR operation on 16-bit data

WXOR(P) [when two operands are set]

These instructions perform an XOR operation on the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 343 WXOR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 343 WXOR(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
WXOR	
WXORP	

Setting data

■Description, range, data type

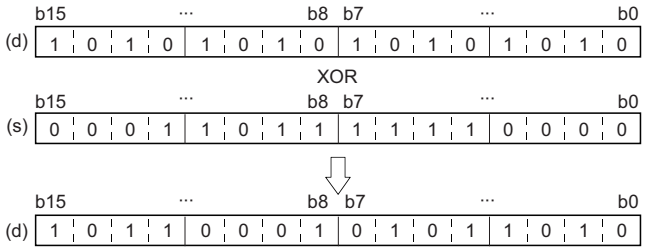
Operand	Description	Range	Data type	Data type (label)
(s)	Exclusive OR data or the device where exclusive OR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	-32768 to 32767	16-bit signed binary	ANY16

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions perform an XOR operation (bit-by-bit) on the 16-bit binary data in the device specified by (d) and the 16-bit binary data in the device specified by (s), and store the operation result in the device specified by (d).



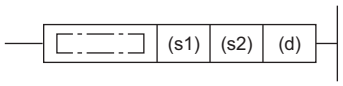
- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

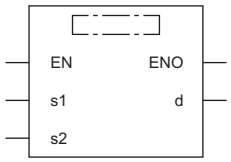
There is no operation error.

WXOR(P) [when three operands are set]

These instructions perform an XOR operation on the two sets of 16-bit binary data specified.

Ladder	ST
	ENO:=WXOR(EN,s1,s2,d); ENO:=WXORP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
WXOR	
WXORP	

Setting data

■Description, range, data type

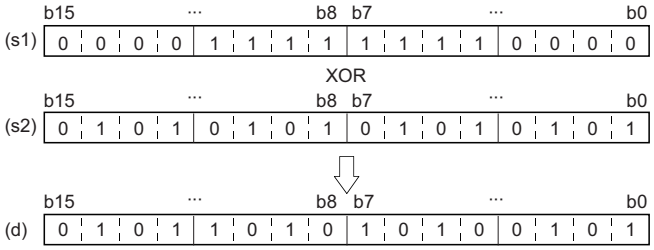
Operand	Description	Range	Data type	Data type (label)
(s1)	Exclusive OR data or the device where exclusive OR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(s2)	Exclusive OR data or the device where exclusive OR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions perform an XOR operation (bit-by-bit) on the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

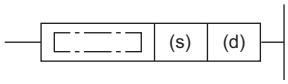
Operation error

There is no operation error.

Performing an XOR operation on 32-bit data

DXOR(P) [when two operands are set]

These instructions perform an XOR operation on the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 347 DXOR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 347 DXOR(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
DXOR	
DXORP	

Setting data

■Description, range, data type

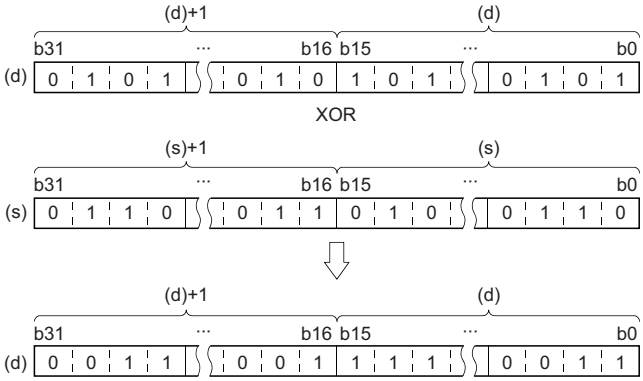
Operand	Description	Range	Data type	Data type (label)
(s)	Exclusive OR data or the start device where exclusive OR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	-2147483648 to 2147483647	32-bit signed binary	ANY32

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions perform an XOR operation (bit-by-bit) on the 32-bit binary data in the device specified by (d) and the 32-bit binary data in the device specified by (s), and store the operation result in the device specified by (d).



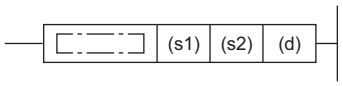
- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

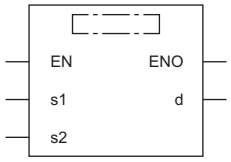
There is no operation error.

DXOR(P) [when three operands are set]

These instructions perform an XOR operation on the two sets of 32-bit binary data specified.

Ladder	ST
	ENO:=DXOR(EN,s1,s2,d); ENO:=DXORP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DXOR	
DXORP	

Setting data

■Description, range, data type

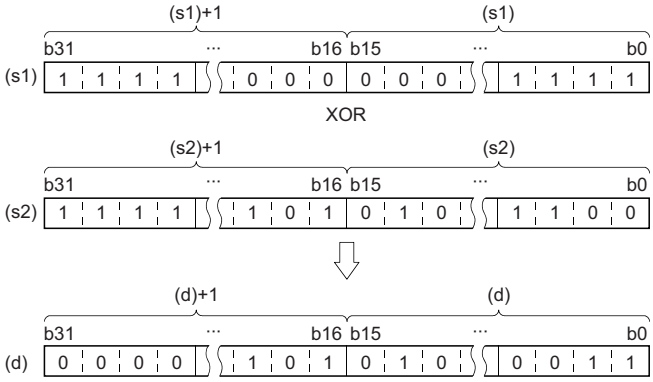
Operand	Description	Range	Data type	Data type (label)
(s1)	Exclusive OR data or the start device where exclusive OR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(s2)	Exclusive OR data or the start device where exclusive OR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions perform an XOR operation (bit-by-bit) on the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

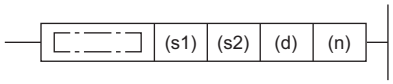
Operation error

There is no operation error.

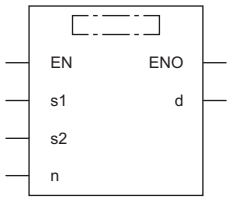
Performing an XOR operation on 16-bit block data

BKXOR(P)

These instructions perform an XOR operation on the two 16-bit binary data blocks specified.

Ladder	ST
	ENO:=BKXOR(EN,s1,s2,n,d); ENO:=BKXORP(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BKXOR	
BKXORP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the logical operation data is stored	—	16-bit signed binary	ANY16*1
(s2)	Logical operation data or the start device where the logical operation data is stored	-32768 to 32767	16-bit signed binary	ANY16*1
(d)	Start device for storing the operation result	—	16-bit signed binary	ANY16*1
(n)	Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

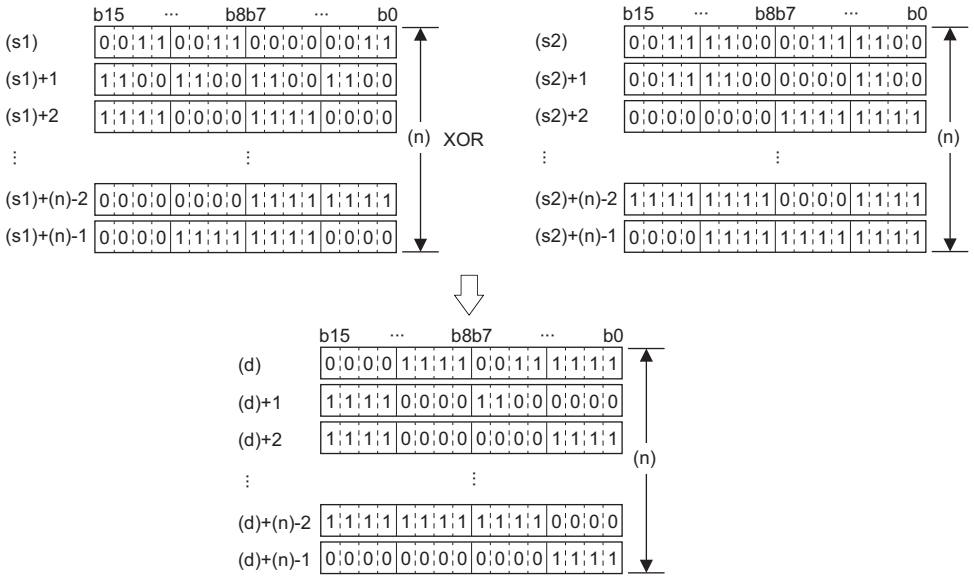
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)*1	—	—	○	—	—	—	—	○	—	—	—	—
(s2)*1	—	—	○	—	—	—	—	○	○	—	—	—
(d)*1	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

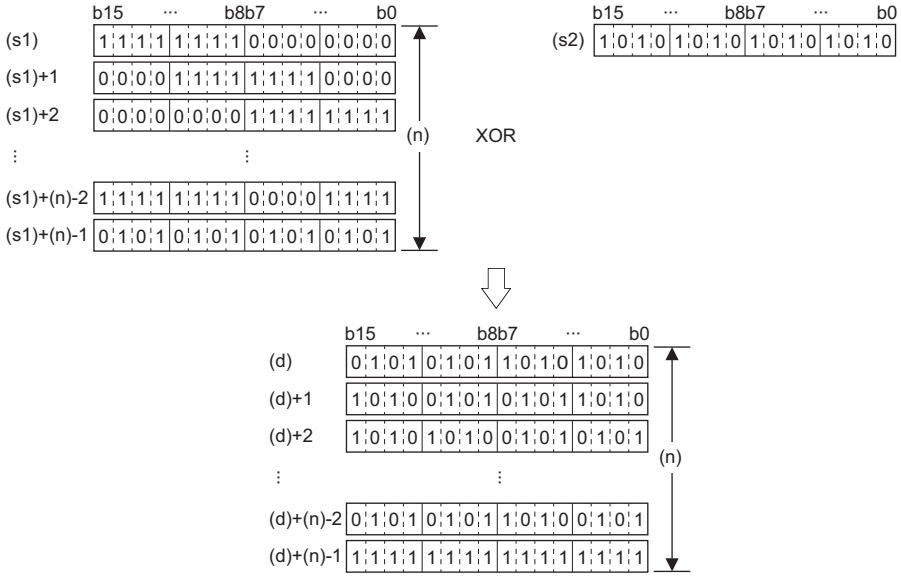
*1 The same device number can be specified for (s1) and (d) or (s2) and (d).

Processing details

- These instructions perform an XOR operation on the (n) points of data from the device specified by (s1) and the (n) points of data from the device specified by (s2), and store the operation result in the device specified by (d) and later.



- A constant from -32768 to 32767 (16-bit signed binary) can be specified for (s2).



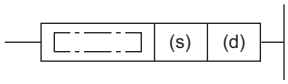
Operation error

Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are partially overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are partially overlapping (except when the same device is specified for (s2) and (d)).

Performing an XNOR operation on 16-bit data

WXNR(P) [when two operands are set]

These instructions perform an XNOR operation on the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 352 WXNR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 352 WXNR(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
WXNR	
WXNRP	

Setting data

■Description, range, data type

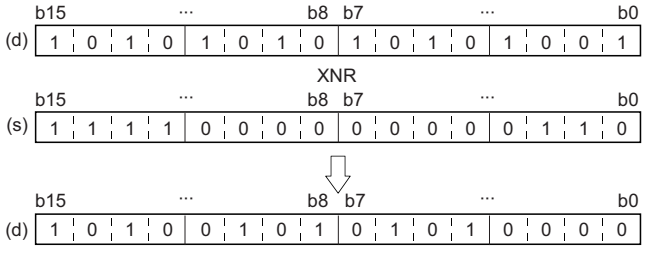
Operand	Description	Range	Data type	Data type (label)
(s)	Exclusive NOR data or the device where exclusive NOR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	-32768 to 32767	16-bit signed binary	ANY16

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions perform an XNOR operation on the 16-bit binary data in the device specified by (d) and the 16-bit binary data in the device specified by (s), and store the operation result in the device specified by (d).



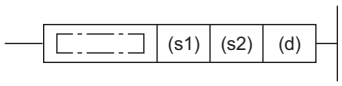
- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

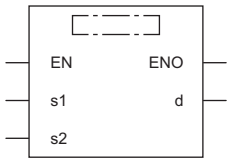
There is no operation error.

WXNR(P) [when three operands are set]

These instructions perform an XNOR operation on the two sets of 16-bit binary data specified.

Ladder	ST
	ENO:=WXNR(EN,s1,s2,d); ENO:=WXNRP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
WXNR	
WXNRP	

Setting data

■Description, range, data type

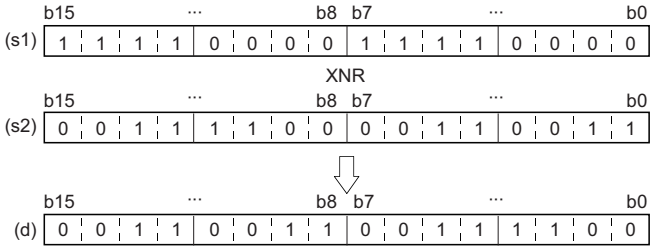
Operand	Description	Range	Data type	Data type (label)
(s1), (s2)	Exclusive NOR data or the device where exclusive NOR data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the operation result	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions perform an exclusive NOR operation on the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

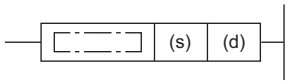
Operation error

There is no operation error.

Performing an XNOR operation on 32-bit data

DXNR(P) [when two operands are set]

These instructions perform an XNOR operation on the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 356 DXNR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 356 DXNR(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
DXNR	
DXNRP	

Setting data

■Description, range, data type

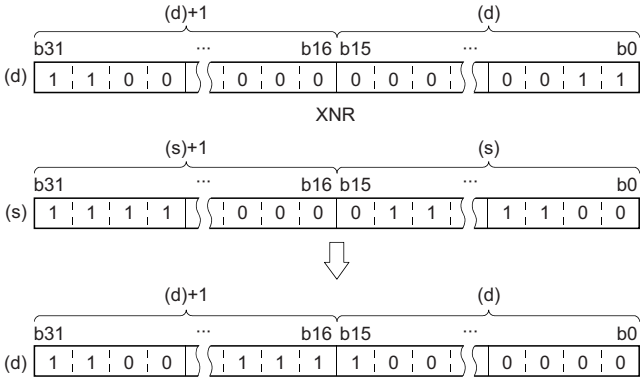
Operand	Description	Range	Data type	Data type (label)
(s)	Exclusive NOR data or the start device where exclusive NOR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	-2147483648 to 2147483647	32-bit signed binary	ANY32

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions perform an XNOR operation on the 32-bit binary data in the device specified by (d) and the 32-bit binary data in the device specified by (s), and store the operation result in the device specified by (d).



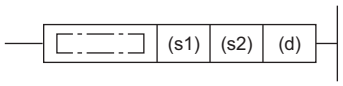
- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

Operation error

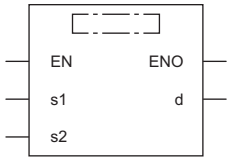
There is no operation error.

DXNR(P) [when three operands are set]

These instructions perform an XNOR operation on the two sets of 32-bit binary data specified.

Ladder	ST
	ENO:=DXNR(EN,s1,s2,d); ENO:=DXNRP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DXNR	
DXNRP	

Setting data

■Description, range, data type

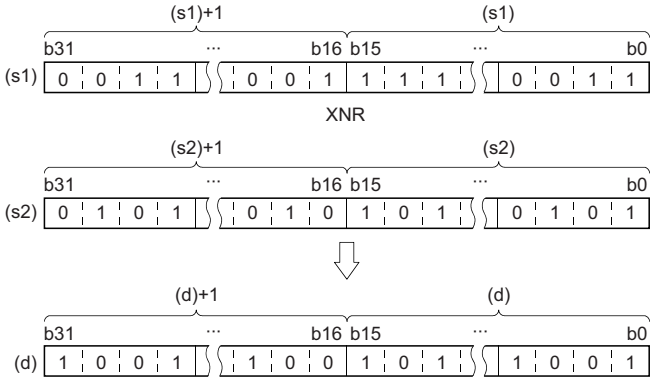
Operand	Description	Range	Data type	Data type (label)
(s1), (s2)	Exclusive NOR data or the start device where exclusive NOR data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the operation result	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions perform an XNOR operation on the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



- When a bit device is specified, the instruction performs an operation by assuming that the ones after the number of digit-specified points are 0.

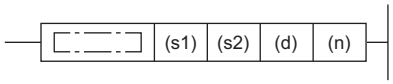
Operation error

There is no operation error.

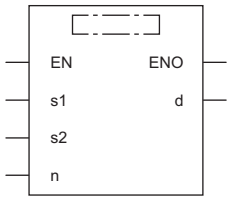
Performing an XNOR operation on 16-bit block data

BKXNR(P)

These instructions perform an XNOR operation on the two 16-bit binary data blocks specified.

Ladder	ST
	ENO:=BKXNR(EN,s1,s2,n,d); ENO:=BKXNRP(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BKXNR	
BKXNRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the logical operation data is stored	—	16-bit signed binary	ANY16*1
(s2)	Logical operation data or the start device where the logical operation data is stored	-32768 to 32767	16-bit signed binary	ANY16*1
(d)	Start device for storing the operation result	—	16-bit signed binary	ANY16*1
(n)	Number of data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

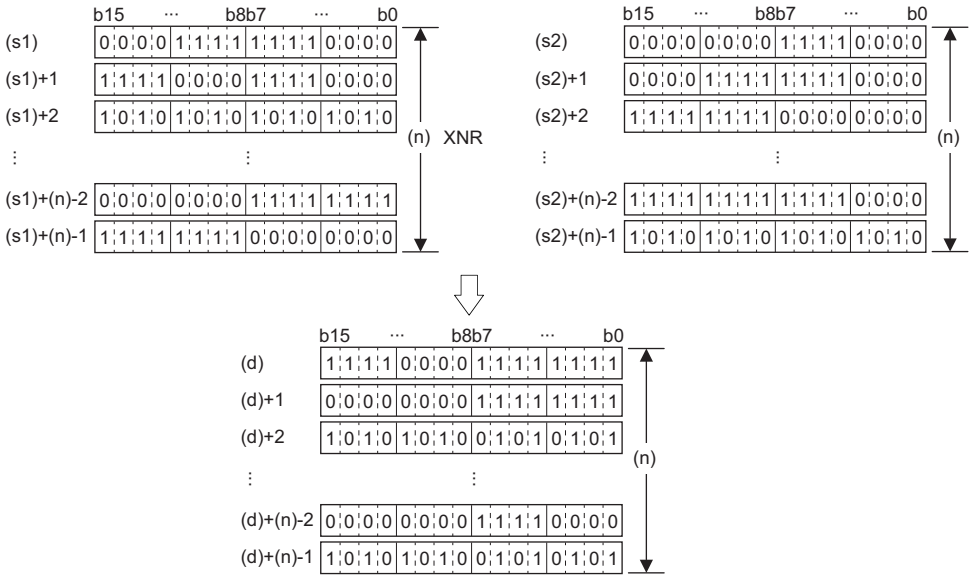
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)*1	—	—	○	—	—	—	—	○	—	—	—	—
(s2)*1	—	—	○	—	—	—	—	○	○	—	—	—
(d)*1	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

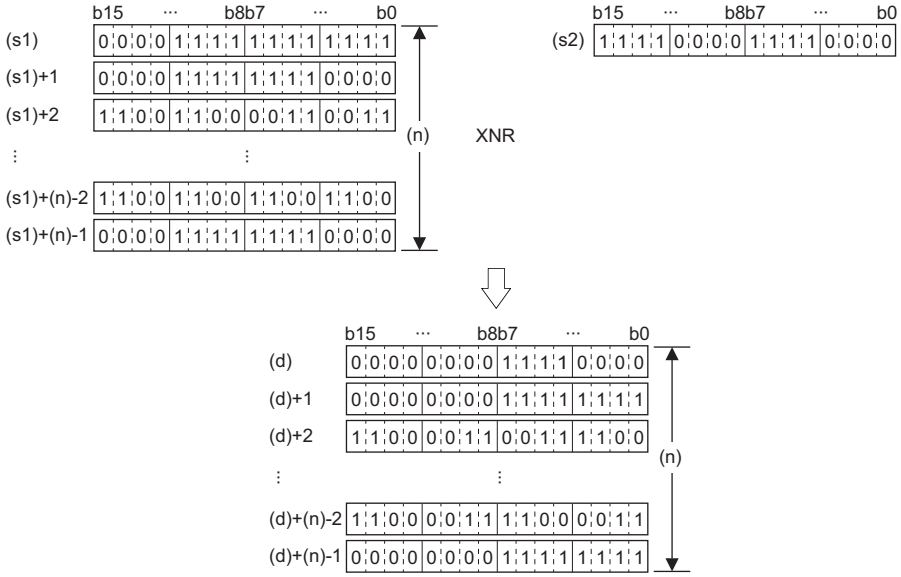
*1 The same device number can be specified for (s1) and (d) or (s2) and (d).

Processing details

- These instructions perform an exclusive NOR operation on the (n) points of data from the device specified by (s1) and the (n) points of data from the device specified by (s2), and store the operation result in the device specified by (d) and later.



- A constant from -32768 to 32767 (16-bit signed binary) can be specified for (s2).



Operation error

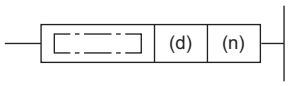
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are partially overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are partially overlapping (except when the same device is specified for (s2) and (d)).

12.4 Bit Processing Instructions

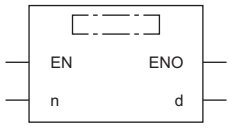
Setting a bit in the word device

BSET(P)

These instructions set the 'n'th bit in the specified word device to 1.

Ladder	ST
	ENO:=BSET(EN,n,d); ENO:=BSETP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BSET	
BSETP	

Setting data

■Description, range, data type

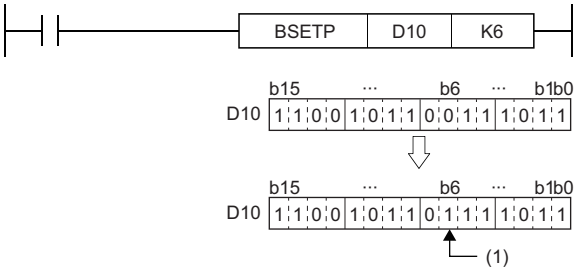
Operand	Description	Range	Data type	Data type (label)
(d)	Set target device	—	16-bit signed binary	ANY16
(n)	Set target bit position	0 to 15	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions set the 'n'th bit in the word device specified by (d) to 1.



(1) Set b6 of D10 to 1.

- If (n) exceeds 15, the instruction sets lower 4 bits of data.

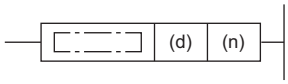
Operation error

There is no operation error.

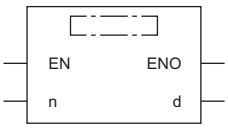
Resetting a bit in the word device

BRST(P)

These instructions reset the 'n'th bit in the specified word device to 0.

Ladder	ST
	ENO:=BRST(EN,n,d); ENO:=BRSTP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BRST	
BRSTP	

Setting data

■Description, range, data type

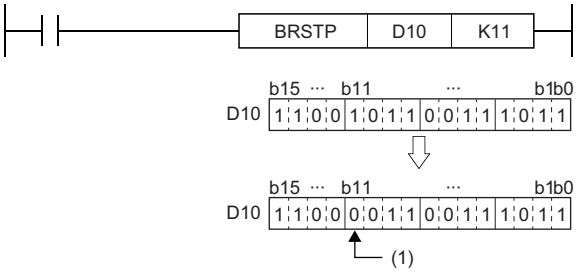
Operand	Description	Range	Data type	Data type (label)
(d)	Reset target device	—	16-bit signed binary	ANY16
(n)	Reset target bit position	0 to 15	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions reset the 'n'th bit in the word device specified by (d) to 0.



(1) Reset the b11 of D10 to 0.

- If (n) exceeds 15, the instruction sets lower 4 bits of data.

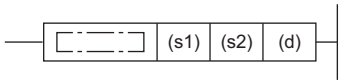
Operation error

There is no operation error.

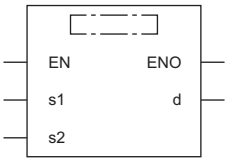
Performing a 16-bit test

TEST(P)

These instructions extract the 'n'th bit in the specified word device.

Ladder	ST
	ENO:=TEST(EN,s1,s2,d); ENO:=TESTP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
TEST	
TESTP	

Setting data

■Description, range, data type

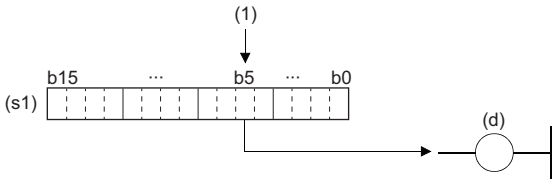
Operand	Description	Range	Data type	Data type (label)
(s1)	Device where the extract target bit data is stored	—	16-bit signed binary	ANY16
(s2)	Extract target bit position	0 to 15	16-bit unsigned binary	ANY16
(d)	Device for storing the extracted bit data	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	—	—	—	○	—	—	—	—

Processing details

- These instructions extract the bit data at the position specified by (s2) of the word device specified by (s1), and write it to the bit device specified by (d).



(1) (s2) bit (When (s2)=5)

- The bit device specified by (d) turns off when the extracted bit data is 0 and turns on when the bit data is 1.
- Specify the bit position (0 to 15) of the word data in (s2). When 16 or a greater value is specified in (s2), the remainder of (s2)÷16 becomes the bit position.

Ex.

When (s2)=18: The remainder of 18÷16 is 2, and therefore the data in bit 2 will be extracted.

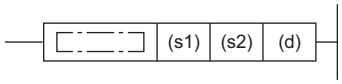
Operation error

There is no operation error.

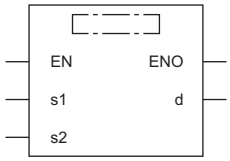
Performing a 32-bit test

DTEST(P)

These instructions extract the 'n'th bit in the specified double-word device.

Ladder	ST
	ENO:=DTEST(EN,s1,s2,d); ENO:=DTESTP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DTEST	
DTESTP	

Setting data

■Description, range, data type

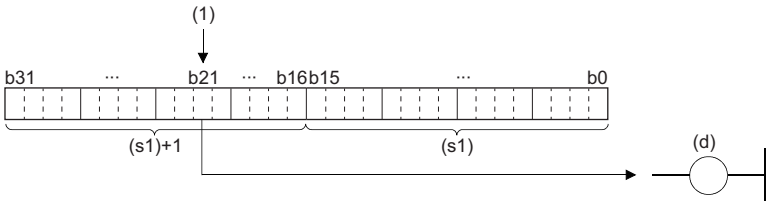
Operand	Description	Range	Data type	Data type (label)
(s1)	Device where the extract target bit data is stored	—	32-bit signed binary	ANY32
(s2)	Extract target bit position	0 to 31	16-bit unsigned binary	ANY16
(d)	Device for storing the extracted bit data	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	○	○	○	○	○	○	○	—	—	—	—
(s2)	—	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	—	—	—	○	—	—	—	—

Processing details

- These instructions extract the bit data at the position specified by (s2) of the double-word device specified by (s1), and write it to the bit device specified by (d).



(1) (s2) bit (When (s2)=21)

- The bit device specified by (d) turns off when the extracted bit data is 0 and turns on when the bit data is 1.
- Specify the bit position (0 to 31) of the double-word data in (s2). When 32 or a greater value is specified in (s2), the remainder of (s2)÷32 becomes the bit position.

Ex.

When (s2)=34: The remainder of 34÷32 is 2, and therefore the data in bit 2 will be extracted.

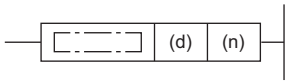
Operation error

There is no operation error.

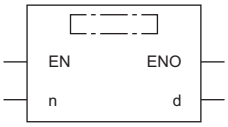
Batch-resetting bit devices

BKRST(P)

These instructions reset the (n) points of bit devices starting from the bit device specified.

Ladder	ST
	ENO:=BKRST(EN,n,d); ENO:=BKRSTP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BKRST	
BKRSTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device to be reset	—	Bit	ANY_BOOL
(n)	Number of reset target devices	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	—	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions reset the (n) points of bit devices starting from the bit device specified by (d).
- The following table lists the reset status of the bit devices.

Device	Status
Annunciator (F)	• The (n) points of data starting from the annunciator (F) number in the device specified by (d) turn off. • The annunciator numbers that turned off are deleted from SD64 to SD79, and the remaining data are compressed forward. • The number of annunciators stored in SD64 to SD79 is stored in SD63.
Timer (T), retentive timer (ST), counter (C), long timer (LT), long retentive timer (LST), long counter (LC)	• The (n) points, starting from the timer (T), retentive timer (ST), counter (C), long timer (LT), long retentive timer (LST), or long counter (LC) number in the device specified by (d), of current values are reset to 0, and the (n) points of coils and contacts are turned off.
Bit devices other than the above	• The (n) points, starting from the device specified by (d), of coils and contacts are turned off.

- When the specified device is off, the device status does not change.

Operation error

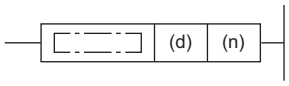
There is no operation error.

12.5 Shift Instructions

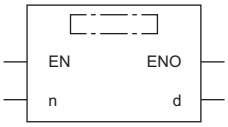
Shifting 16-bit binary data to the right by n bit(s)

SFR(P)

These instructions shift the 16-bit binary data in the specified device to the right by (n) bit(s). In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=SFR(EN,n,d); ENO:=SFRP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFR	
SFRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	16-bit signed binary	ANY16
(n)	Number of shifts	0 to 15	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	○	○	○	○	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

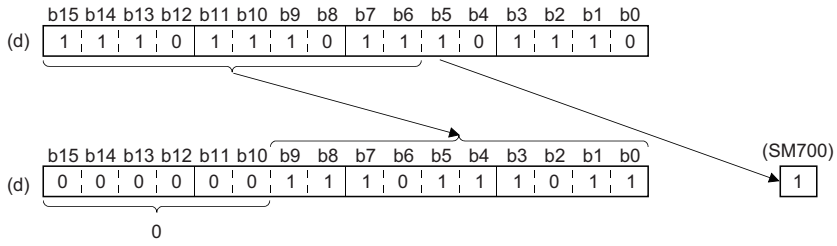
*1 FX and FY cannot be used.

Processing details

- These instructions shift the 16-bit binary data in the device specified by (d) to the right by (n) bit(s).
- The (n) bit(s) from the most significant bit is/are filled with 0(s).
- In SM700 (carry flag), a value in a bit to the right of the shift target area is stored.

Ex.

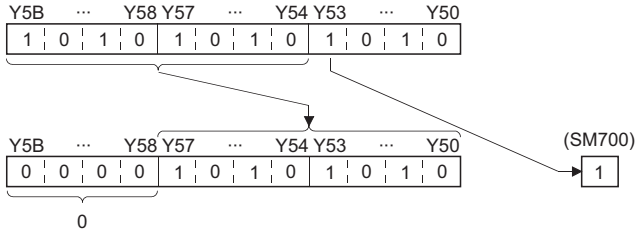
When (n)=6



- When (d) is a bit device, bits are shifted to the right within the device range specified by digit specification.

Ex.

When (n)=4



- The number of bits actually to be shifted is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 8, 7 bits are shifted because 15 divided by 8 equals 1 with a remainder of 7.
- Specify any value between 0 and 15 for (n). If a value 16 or bigger is specified, bits are shifted by the remainder value of n÷16. For example, when (n) is 18, 2 bits are shifted to the right because 18 divided by 16 equals 1 with a remainder of 2.

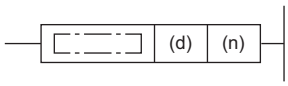
Operation error

There is no operation error.

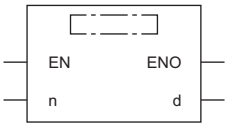
Shifting 16-bit binary data to the left by n bit(s)

SFL(P)

These instructions shift the 16-bit binary data in the specified device to the left by (n) bit(s). In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=SFL(EN,n,d); ENO:=SFLP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFL	
SFLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	16-bit signed binary	ANY16
(n)	Number of shifts	0 to 15	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	○	○	○	○	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

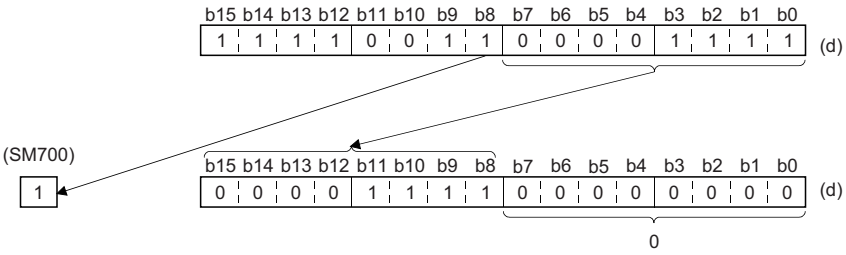
*1 FX and FY cannot be used.

Processing details

- This instruction shifts the 16-bit binary data in the device specified by (d) to the left by (n) bit(s).
- The (n) bit(s) from the least significant bit is/are filled with 0(s).
- In SM700 (carry flag), a value in a bit to the left of the shift target area is stored.

Ex.

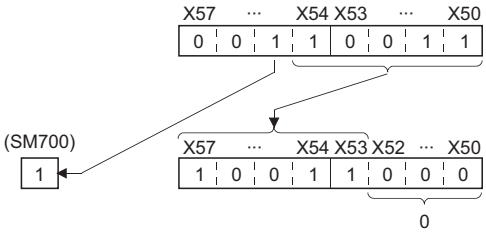
When (n)=8



- When (d) is a bit device, bits are shifted to the left within the device range specified by digit specification.

Ex.

When (n)=5



- The number of bits actually to be shifted is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 8, 7 bits are shifted because 15 divided by 8 equals 1 with a remainder of 7.
- Specify any value between 0 and 15 for (n). If a value 16 or larger is specified, the value is shifted by the remainder value of n÷16 to the left. For example, when (n) is 18, 2 bits are shifted to the left because 18 divided by 16 equals 1 with a remainder of 2.

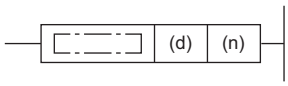
Operation error

There is no operation error.

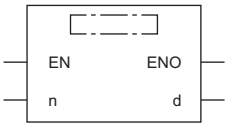
Shifting n-bit data to the right by one bit

BSFR(P)

These instructions shift the (n) points of data starting from the specified device to the right by one bit. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=BSFR(EN,n,d); ENO:=BSFRP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BSFR	
BSFRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Bit	ANY_BOOL
(n)	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

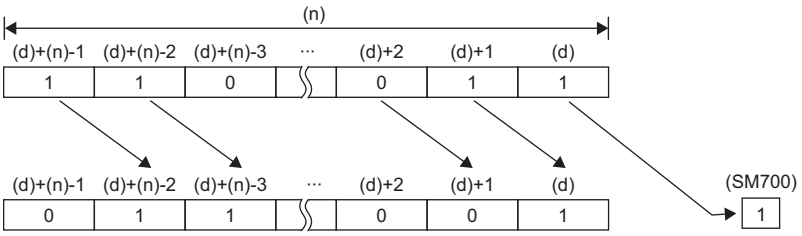
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	—	○	—	—	—	—	○	—	—	—	—
(n)	○*2	○	○	○	○	—	—	○	○	—	—	—

*1 T, C, and ST cannot be used.
*2 FX and FY cannot be used.

Processing details

- These instructions shift the (n) points of data starting from the device specified by (d) to the right by one bit.
- The most significant bit is filled with 0.
- In SM700 (carry flag), a value in a bit to the right of the shift target area is stored.



- If the value specified by (n) is 0, the instruction will be not processed.

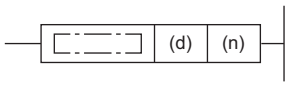
Operation error

There is no operation error.

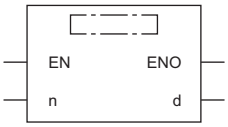
Shifting n-bit data to the left by one bit

BSFL(P)

These instructions shift the (n) points of data starting from the specified device to the left by one bit. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=BSFL(EN,n,d); ENO:=BSFLP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BSFL	
BSFLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Bit	ANY_BOOL
(n)	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

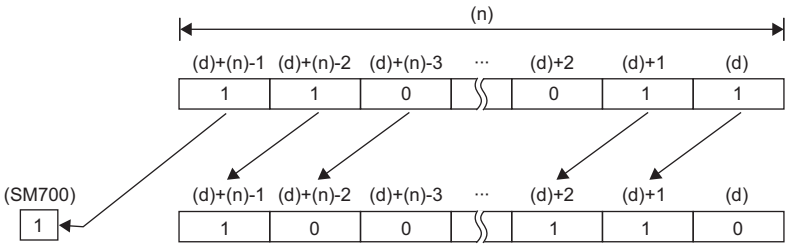
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	—	○	—	—	—	—	○	—	—	—	—
(n)	○*2	○	○	○	○	—	—	○	○	—	—	—

*1 T, C, and ST cannot be used.
*2 FX and FY cannot be used.

Processing details

- These instructions shift the (n) points of data starting from the device specified by (d) to the left by one bit.
- The least significant bit is filled with 0.
- In SM700 (carry flag), a value in a bit to the left of the shift target area is stored.



- If the value specified by (n) is 0, the instruction will be not processed.

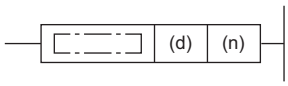
Operation error

There is no operation error.

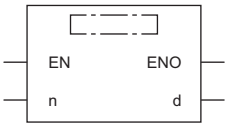
Shifting n-word data to the right by one word

DSFR(P)

These instructions shift the (n) points of data starting from the specified device to the right by one word. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=DSFR(EN,n,d); ENO:=DSFRP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSFR	
DSFRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Word	ANY16
(n)	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

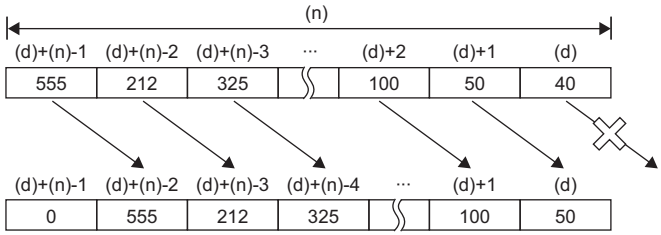
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

*1 FX and FY cannot be used.

Processing details

- These instructions shift the (n) points of data starting from the device specified by (d) to the right by one word.
- One word from the most significant bit is filled with 0.



- If the value specified by (n) is 0, the instruction will be not processed.

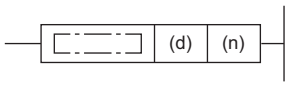
Operation error

There is no operation error.

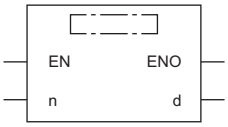
Shifting n-word data to the left by one word

DSFL(P)

These instructions shift the (n) points of data starting from the specified device to the left by one word. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=DSFL(EN,n,d); ENO:=DSFLP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSFL	
DSFLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Word	ANY16
(n)	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

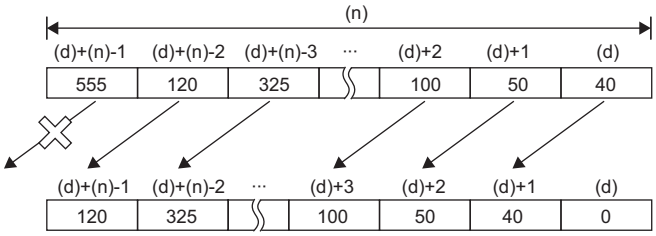
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

*1 FX and FY cannot be used.

Processing details

- These instructions shift the (n) points of data starting from the device specified by (d) to the left by one word.
- One word from the least significant bit is filled with 0.



- If (n) is 0, no processing is performed.

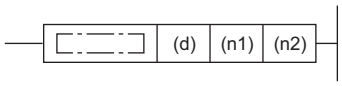
Operation error

There is no operation error.

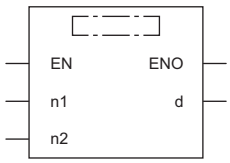
Shifting n-bit data to the right by n bit(s)

SFTBR(P)

These instructions shift the bit data to the right by (n2) bit(s) within (n1) bits starting from the specified device. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=SFTBR(EN,n1,n2,d); ENO:=SFTBRP(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTBR	
SFTBRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Bit	ANY_BOOL
(n1)	Amount of data to be shifted	0 to 64	16-bit unsigned binary	ANY16
(n2)	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*2	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*2	○	○	○	○	—	—	○	○	—	—	—

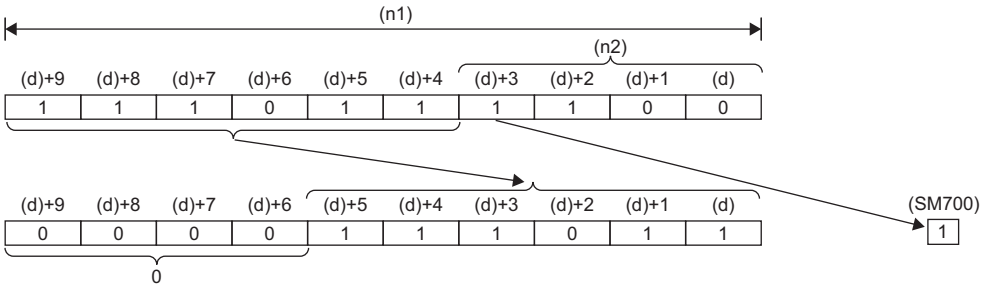
*1 T, C, and ST cannot be used.
*2 FX and FY cannot be used.

Processing details

- These instructions shift bit data to the right by the (n2) bit(s) within the (n1) bits of data area starting from the device specified by (d).
- In SM700 (carry flag), a value in a bit to the right of the shift target area is stored.

Ex.

When (n1)=10 and (n2)=4



- Specify (n1) and (n2) so that the following condition is satisfied: (n1)>(n2). In the case of (n1)≤(n2), data is shifted by the value of the remainder of (n2)÷(n1). However, if the remainder value is 0, no processing is performed.
- Specify (n1) within the range of 1 to 64.
- The (n2) bit(s) from the most significant bit is/are filled with 0(s). In the case of (n1)<(n2), the bits are filled with 0s by the value of the remainder of (n2)÷(n1).
- If (n1) or (n2) is 0, no processing is performed.

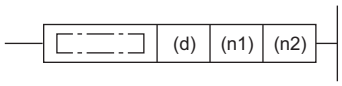
Operation error

Error code (SD0)	Description
3285H	The value specified by (n1) is out of the range, 0 to 64.

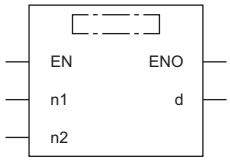
Shifting n-bit data to the left by n bit(s)

SFTBL(P)

These instructions shift the bit data to the left by (n2) bit(s) within (n1) bits starting from the specified device. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=SFTBL(EN,n1,n2,d); ENO:=SFTBLP(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTBL	
SFTBLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Bit	ANY_BOOL
(n1)	Amount of data to be shifted	0 to 64	16-bit unsigned binary	ANY16
(n2)	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*2	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*2	○	○	○	○	—	—	○	○	—	—	—

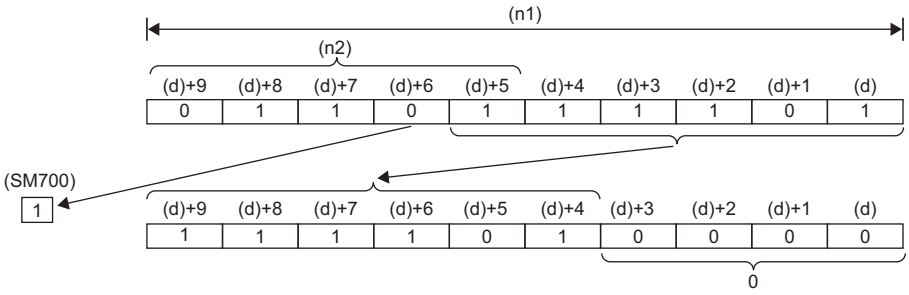
*1 T, C, and ST cannot be used.
*2 FX and FY cannot be used.

Processing details

- These instructions shift bit data to the left by the (n2) bit(s) within the (n1) bits of data area starting from the device specified by (d).
- In SM700 (carry flag), a value in a bit to the left of the shift target area is stored.

Ex.

When (n1)=10 and (n2)=4



- Specify (n1) and (n2) so that the following condition is satisfied: (n1)>(n2). In the case of (n1)≤(n2), data is shifted by the value of the remainder of (n2)÷(n1). However, if the remainder value is 0, no processing is performed.
- Specify (n1) within the range of 1 to 64.
- The (n2) bit(s) from the least significant bit is/are filled with 0(s). In the case of (n1)<(n2), the bits are filled with 0s by the value of the remainder of (n2)÷(n1).
- If (n1) or (n2) is 0, no processing is performed.

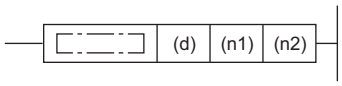
Operation error

Error code (SD0)	Description
3285H	The value specified by (n1) is out of the range, 0 to 64.

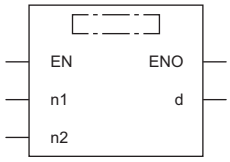
Shifting n-word data to the right by n word(s)

SFTWR(P)

These instructions shift the word data to the right by (n2) word(s) within (n1) words starting from the specified device. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=SFTWR(EN,n1,n2,d); ENO:=SFTWRP(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTWR	
SFTWRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Word	ANY16
(n1)	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16
(n2)	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

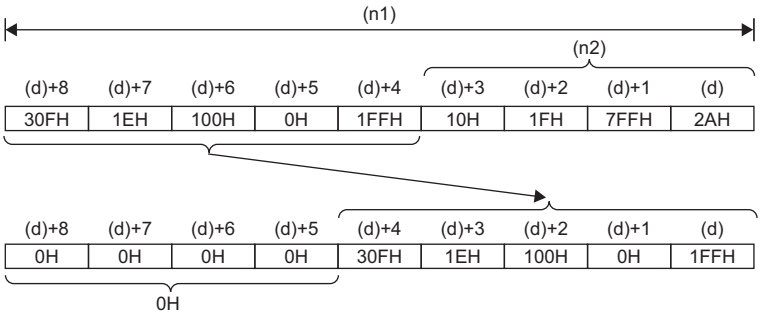
*1 FX and FY cannot be used.

Processing details

- These instructions shift word data to the right by the (n2) word(s) within the (n1) words of data area starting from the device specified by (d).
- The (n2) word(s) from the most significant bit is/are filled with 0H(s).
- If (n1) or (n2) is 0H, no processing is performed.
- In the case of (n1)≤(n2), (n1) words of data starting from the device specified by (d) become all 0Hs.

Ex.

When (n1)=9 and (n2)=4



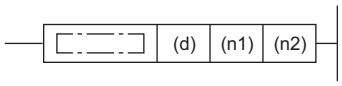
Operation error

There is no operation error.

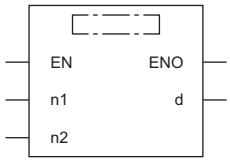
Shifting n-word data to the left by n word(s)

SFTWL(P)

These instructions shift the word data to the left by (n2) word(s) within (n1) words starting from the specified device. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=SFTWL(EN,n1,n2,d); ENO:=SFTWLP(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTWL	
SFTWLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target device	—	Word	ANY16
(n1)	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16
(n2)	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

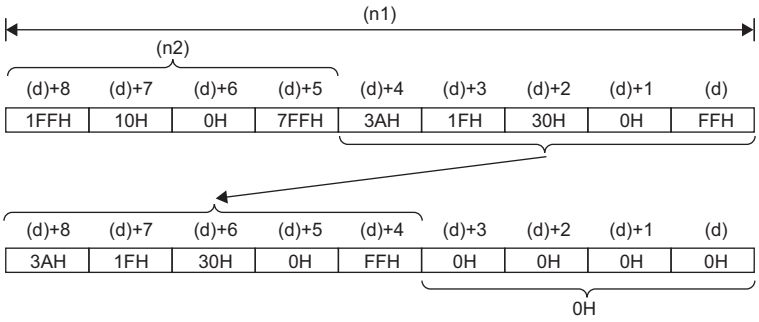
*1 FX and FY cannot be used.

Processing details

- These instructions shift word data to the left by the (n2) word(s) within the (n1) words of data area starting from the device specified by (d).
- The (n2) word(s) from the least significant bit is/are filled with 0H(s).
- If (n1) or (n2) is 0H, no processing is performed.
- In the case of (n1)≤(n2), (n1) words of data starting from the device specified by (d) become all 0Hs.

Ex.

When (n1)=9 and (n2)=4



Operation error

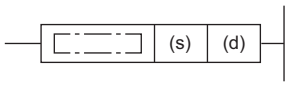
There is no operation error.

12.6 Data Conversion Instructions

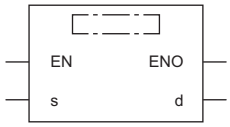
Converting binary data to BCD 4-digit data

BCD(P)

These instructions convert the specified 16-bit binary data to BCD 4-digit data.

Ladder	ST
	ENO:=BCD(EN,s,d); ENO:=BCDP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BCD	
BCDP	

Setting data

■Description, range, data type

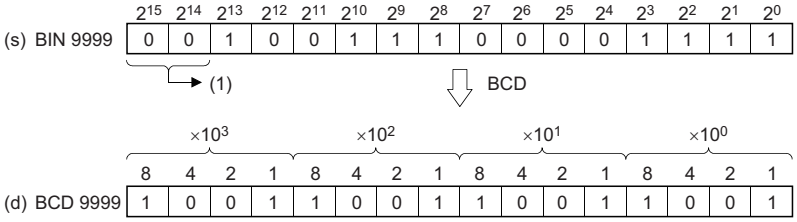
Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the device where the binary data is stored	0 to 9999	16-bit signed binary	ANY16
(d)	Device for storing the converted BCD data	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	○	○	○	○	—	—	○	○	—	—	—
(d)	—	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the 16-bit binary data (0 to 9999) in the device specified by (s) to BCD 4-digit data, and store the converted data in the device specified by (d).



(1) Set 0s.

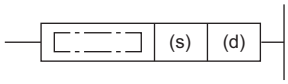
Operation error

Error code (SD0)	Description
3281H	Data in the device specified by (s) is out of the range, 0 to 9999.

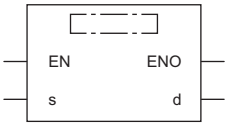
Converting binary data to BCD 8-digit data

DBCD(P)

These instructions convert the specified 32-bit binary data to BCD 8-digit data.

Ladder	ST
	ENO:=DBCD(EN,s,d); ENO:=DBCDP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBCD	
DBCDP	

Setting data

■Description, range, data type

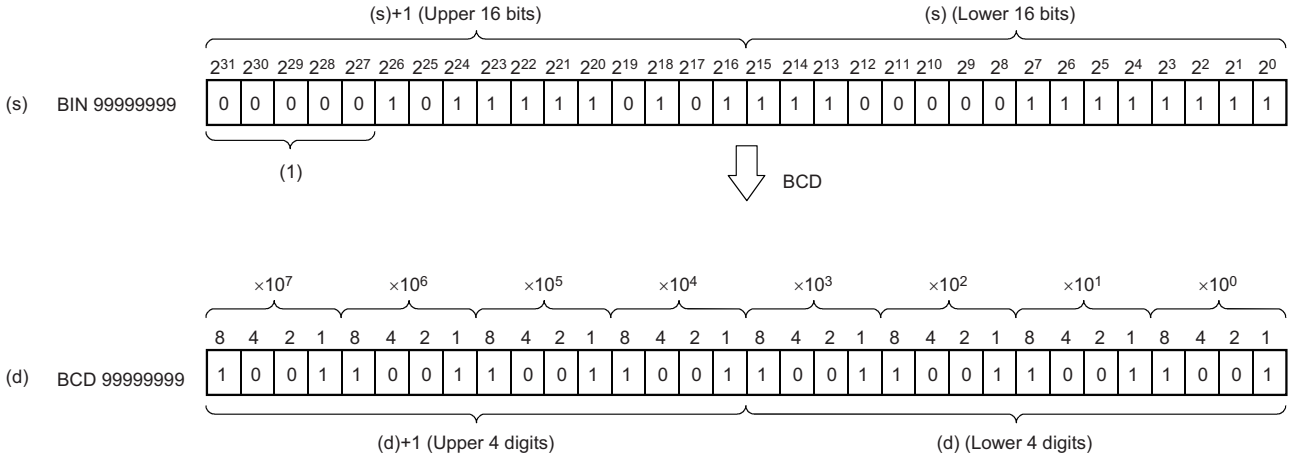
Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the start device where the binary data is stored	0 to 99999999	32-bit signed binary	ANY32
(d)	Start device for storing the converted BCD data	—	BCD 8-digit	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 32-bit binary data (0 to 99999999) in the device specified by (s) to BCD 8-digit data, and store the converted data in the device specified by (d).



(1) Set 0s. (Upper 5 bits)

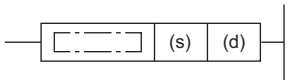
Operation error

Error code (SD0)	Description
3281H	Data in the device specified by (s) is out of the range, 0 to 99999999.

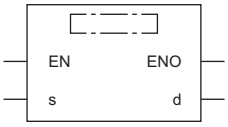
Converting BCD 4-digit data to 16-bit binary data

BIN(P)

These instructions convert the specified BCD 4-digit data to 16-bit binary data.

Ladder	ST
	ENO:=BIN(EN,s,d); ENO:=BINP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BIN	
BINP	

Setting data

■Description, range, data type

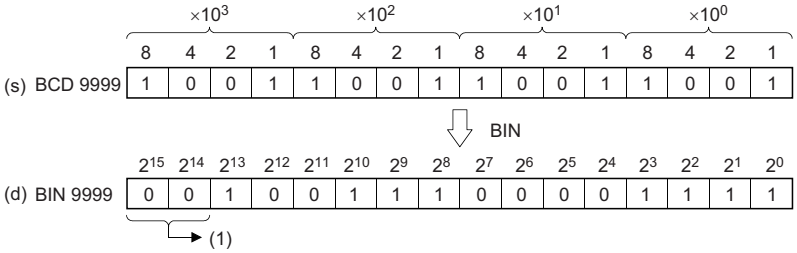
Operand	Description	Range	Data type	Data type (label)
(s)	BCD data or the device where the BCD data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Device for storing the converted binary data	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the BCD 4-digit data (0 to 9999) in the device specified by (s) to 16-bit binary data, and store the converted data in the device specified by (d).



(1) Filled with 0s.

Operation error

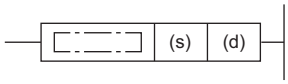
Error code (SD0)	Description
3281H	A value other than 0 to 9 exists at any digit of the value in the device specified by (s).*1

*1 Turning on SM754 (BIN, DBIN instruction error disable flag) can prevent this error from being detected.
If the specified value is out of the valid range, the BIN(P) instruction is not executed regardless of the status (on/off) of SM754.
The BIN(P) instruction does not execute the next operation until the command (execution condition) is turned off and on regardless of the presence of an error.

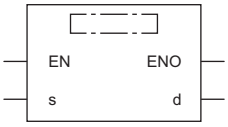
Converting BCD 8-digit data to 32-bit binary data

DBIN(P)

These instructions convert the specified BCD 8-digit data to 32-bit binary data.

Ladder	ST
	ENO:=DBIN(EN,s,d); ENO:=DBINP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBIN	
DBINP	

Setting data

■Description, range, data type

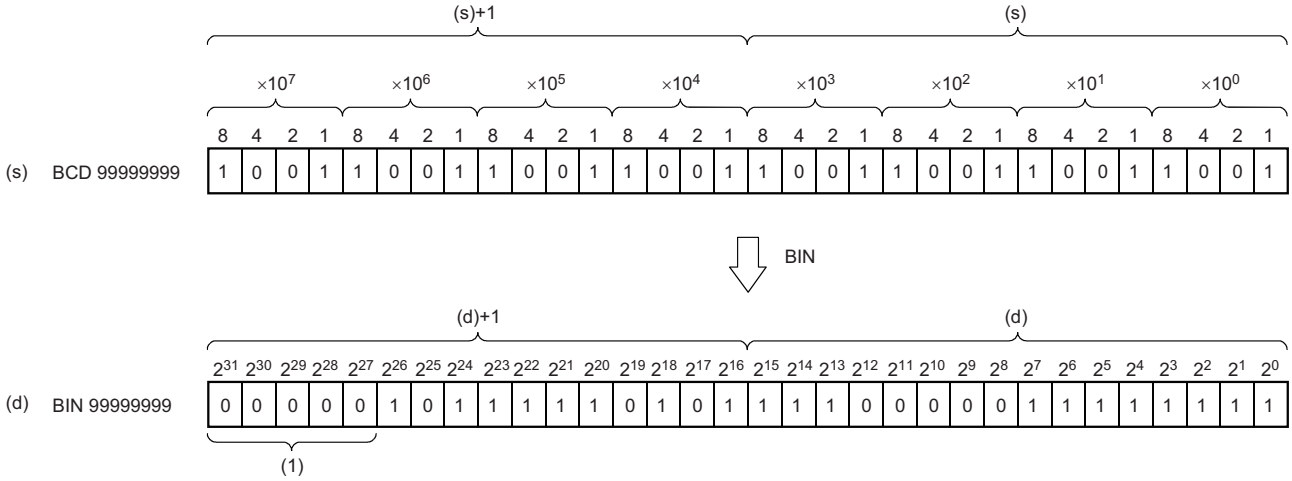
Operand	Description	Range	Data type	Data type (label)
(s)	BCD data or the start device where the BCD data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the converted binary data	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the BCD 8-digit data (0 to 99999999) in the device specified by (s) to 32-bit binary data, and store the converted data in the device specified by (d).



(1) Filled with 0s.

Operation error

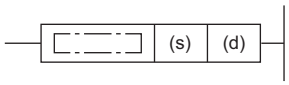
Error code (SD0)	Description
3281H	A value other than 0 to 9 exists at any digit of the value in the device specified by (s).*1

*1 Turning on SM754 (BIN, DBIN instruction error disable flag) can prevent this error from being detected.
If the specified value is out of the valid range, the BIN(P) instruction is not executed regardless of the status (on/off) of SM754.
The DBIN(P) instruction does not execute the next operation until the command (execution condition) is turned off and on regardless of the presence of an error.

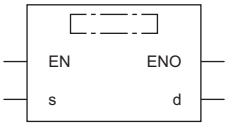
Converting single-precision real number to 16-bit signed binary data

FLT2INT(P)

These instructions convert the specified single-precision real number to 16-bit signed binary data.

Ladder	ST
	ENO:=FLT2INT(EN,s,d); ENO:=FLT2INTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FLT2INT	
FLT2INTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Single-precision real number or the start device where the single-precision real number is stored	-32768 to 32767	Single-precision real number	ANYREAL_32
(d)	Device for storing the converted binary data	—	16-bit signed binary	ANY16_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

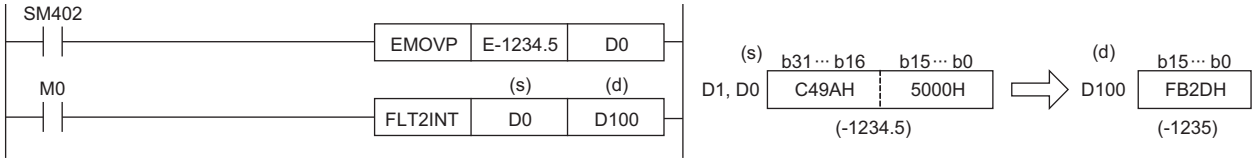
Processing details

- These instructions convert the single-precision real number in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the single-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 40 Precautions

Ex.

When M0 turns on, the single-precision real number stored in D0 and D1 are converted to 16-bit signed binary data, and the converted data is stored in D100.



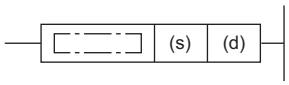
Operation error

Error code (SD0)	Description
3281H	The single-precision real number in the device specified by (s) is out of the range, -32768 to 32767.
3282H	An unusual number is set to (s). • The single-precision real number set to (s) is not within the following range: $0, 2^{-126} \leq \lvert(s)\rvert < 2^{128}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

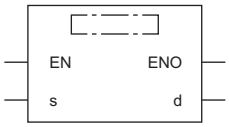
Converting single-precision real number to 16-bit unsigned binary data

FLT2UINT(P)

These instructions convert the specified single-precision real number to 16-bit unsigned binary data.

Ladder	ST
	ENO:=FLT2UINT(EN,s,d); ENO:=FLT2UINTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FLT2UINT	
FLT2UINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Single-precision real number or the start device where the single-precision real number is stored	0 to 65535	Single-precision real number	ANYREAL_32
(d)	Device for storing the converted binary data	—	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

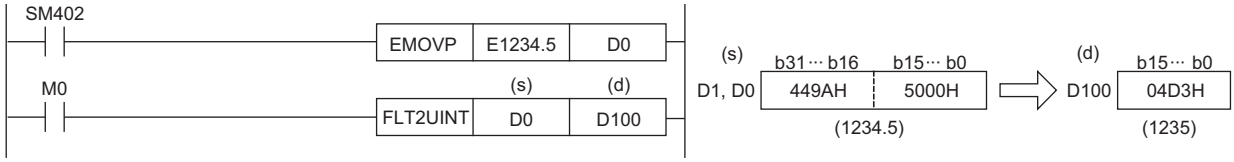
Processing details

- These instructions convert the single-precision real number in the device specified by (s) to 16-bit unsigned binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the single-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 40 Precautions

Ex.

When M0 turns on, the single-precision real number stored in D0 and D1 are converted to 16-bit unsigned binary data, and the converted data is stored in D100.



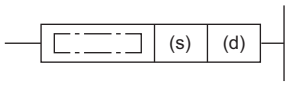
Operation error

Error code (SD0)	Description
3281H	The single-precision real number in the device specified by (s) is out of the range, 0 to 65535.
3282H	An unusual number is set to (s). • The single-precision real number set to (s) is not within the following range: $0, 2^{-126} \leq \lvert(s)\rvert < 2^{128}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

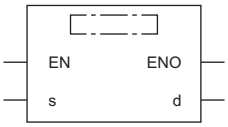
Converting single-precision real number to 32-bit signed binary data

FLT2DINT(P)

These instructions convert the specified single-precision real number to 32-bit signed binary data.

Ladder	ST
	ENO:=FLT2DINT(EN,s,d); ENO:=FLT2DINTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FLT2DINT	
FLT2DINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Single-precision real number or the start device where the single-precision real number is stored	-2147483648 to 2147483647	Single-precision real number	ANYREAL_32
(d)	Start device for storing the converted binary data	—	32-bit signed binary	ANY32_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

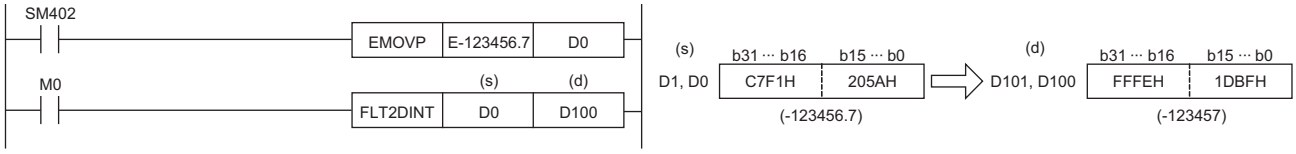
Processing details

- These instructions convert the single-precision real number in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the single-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 40 Precautions

Ex.

When M0 turns on, the single-precision real number stored in D0 and D1 are converted to 32-bit signed binary data, and the converted data is stored in D100 and D101.



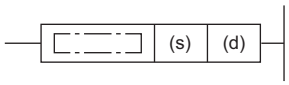
Operation error

Error code (SD0)	Description
3281H	The single-precision real number in the device specified by (s) is out of the range, -2147483648 to 2147483647.
3282H	An unusual number is set to (s). • The single-precision real number set to (s) is not within the following range: $0, 2^{-126} \leq \lvert(s)\rvert < 2^{128}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

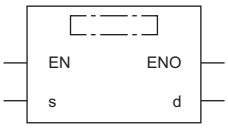
Converting single-precision real number to 32-bit unsigned binary data

FLT2UDINT(P)

These instructions convert the specified single-precision real number to 32-bit unsigned binary data.

Ladder	ST
	ENO:=FLT2UDINT(EN,s,d); ENO:=FLT2UDINTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FLT2UDINT	
FLT2UDINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Single-precision real number or the start device where the single-precision real number is stored	0 to 4294967295	Single-precision real number	ANYREAL_32
(d)	Start device for storing the converted binary data	—	32-bit unsigned binary	ANY32_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

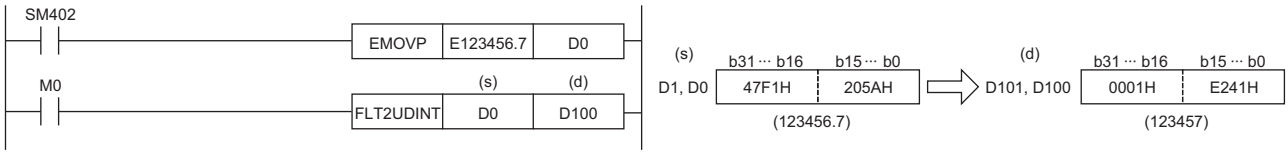
Processing details

- These instructions convert the single-precision real number in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the single-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 40 Precautions

Ex.

When M0 turns on, the single-precision real number stored in D0 and D1 are converted to 32-bit unsigned binary data, and the converted data is stored in D100 and D101.



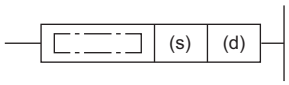
Operation error

Error code (SD0)	Description
3281H	The single-precision real number in the device specified by (s) is out of the range, 0 to 4294967295.
3282H	An unusual number is set to (s). • The single-precision real number set to (s) is not within the following range: $0, 2^{-126} \leq \lvert(s)\rvert < 2^{128}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

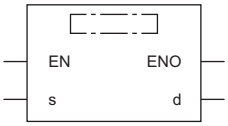
Converting double-precision real number to 16-bit signed binary data

DBL2INT(P)

These instructions convert the specified double-precision real number to 16-bit signed binary data.

Ladder	ST
	ENO:=DBL2INT(EN,s,d); ENO:=DBL2INTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBL2INT	
DBL2INTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Double-precision real number or the start device where the double-precision real number is stored	-32768 to 32767	Double-precision real number	ANYREAL_64
(d)	Device for storing the converted binary data	—	16-bit signed binary	ANY16_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○	—	○	—	—	○	—	—	—	—

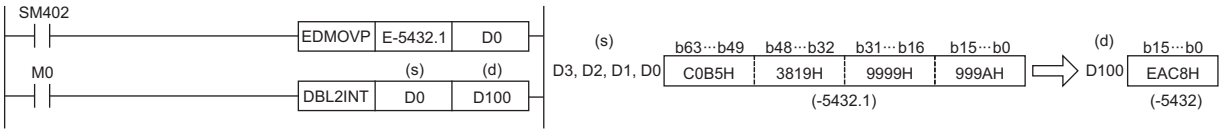
Processing details

- These instructions convert the double-precision real number in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the double-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 40 Precautions

Ex.

When M0 turns on, the double-precision real number stored in D0 to D3 are converted to 16-bit signed binary data, and the converted data is stored in D100.



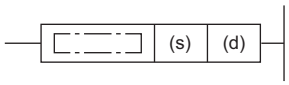
Operation error

Error code (SD0)	Description
3281H	The double-precision real number in the device specified by (s) is out of the range, -32768 to 32767.
3282H	An unusual number is set to (s). • The double-precision real number set to (s) is not within the following range: $0, 2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

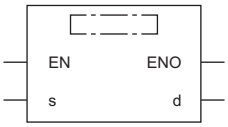
Converting double-precision real number to 16-bit unsigned binary data

DBL2UINT(P)

These instructions convert the specified double-precision real number to 16-bit unsigned binary data.

Ladder	ST
	ENO:=DBL2UINT(EN,s,d); ENO:=DBL2UINTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBL2UINT	
DBL2UINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Double-precision real number or the start device where the double-precision real number is stored	0 to 65535	Double-precision real number	ANYREAL_64
(d)	Device for storing the converted binary data	—	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○	—	○	—	—	○	—	—	—	—

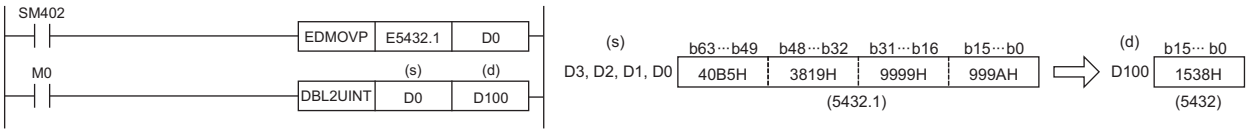
Processing details

- These instructions convert the double-precision real number in the device specified by (s) to 16-bit unsigned binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the double-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 40 Precautions

Ex.

When M0 turns on, the double-precision real number stored in D0 to D3 are converted to 16-bit unsigned binary data, and the converted data is stored in D100.



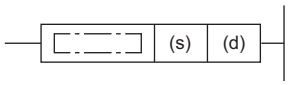
Operation error

Error code (SD0)	Description
3281H	The double-precision real number in the device specified by (s) is out of the range, 0 to 65535.
3282H	An unusual number is set to (s). • The double-precision real number set to (s) is not within the following range: 0, $2^{-1022} \leq |(s)| < 2^{1024}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

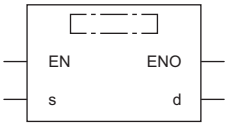
Converting double-precision real number to 32-bit signed binary data

DBL2DINT(P)

These instructions convert the specified double-precision real number to 32-bit signed binary data.

Ladder	ST
	ENO:=DBL2DINT(EN,s,d); ENO:=DBL2DINTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBL2DINT	
DBL2DINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Double-precision real number or the start device where the double-precision real number is stored	-2147483648 to 2147483647	Double-precision real number	ANYREAL_64
(d)	Start device for storing the converted binary data	—	32-bit signed binary	ANY32_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○	—	○	○	○	○	—	—	—	—

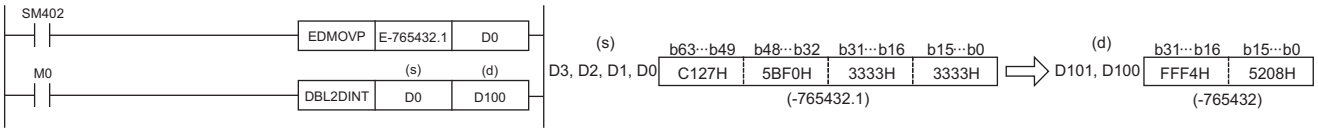
Processing details

- These instructions convert the double-precision real number in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the double-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 40 Precautions

Ex.

When M0 turns on, the double-precision real number stored in D0 to D3 are converted to 32-bit signed binary data, and the converted data is stored in D100 and D101.



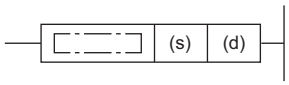
Operation error

Error code (SD0)	Description
3281H	The double-precision real number in the device specified by (s) is out of the range, -2147483648 to 2147483647.
3282H	An unusual number is set to (s). • The double-precision real number set to (s) is not within the following range: $0, 2^{-1022} \leq \|(s)\| < 2^{1024}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

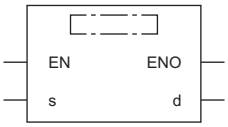
Converting double-precision real number to 32-bit unsigned binary data

DBL2UDINT(P)

These instructions convert the specified double-precision real number to 32-bit unsigned binary data.

Ladder	ST
	ENO:=DBL2UDINT(EN,s,d); ENO:=DBL2UDINTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBL2UDINT	
DBL2UDINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Double-precision real number or the start device where the double-precision real number is stored	0 to 4294967295	Double-precision real number	ANYREAL_64
(d)	Start device for storing the converted binary data	—	32-bit unsigned binary	ANY32_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○	—	○	○	○	○	—	—	—	—

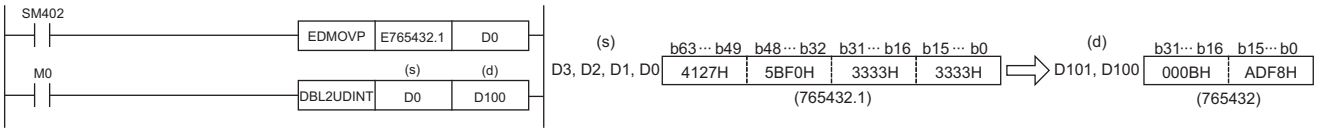
Processing details

- These instructions convert the double-precision real number in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d).
- After conversion, the first digit after the decimal point of the double-precision real number is rounded off.
- When an input value is set using the engineering tool, a rounding error may occur. For the precautions on setting an input value using the engineering tool, refer to the following.

☞ Page 40 Precautions

Ex.

When M0 turns on, the double-precision real number stored in D0 to D3 are converted to 32-bit unsigned binary data, and the converted data is stored in D100 and D101.



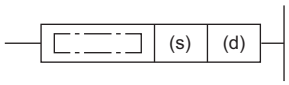
Operation error

Error code (SD0)	Description
3281H	The double-precision real number in the device specified by (s) is out of the range, 0 to 4294967295.
3282H	An unusual number is set to (s). • The double-precision real number set to (s) is not within the following range: 0, $2^{-1022} \leq |(s)| < 2^{1024}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

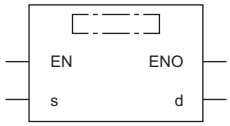
Converting 16-bit signed binary data to 16-bit unsigned binary data

INT2UINT(P)

These instructions convert the specified 16-bit signed binary data to 16-bit unsigned binary data.

Ladder	ST
	ENO:=INT2UINT(EN,s,d); ENO:=INT2UINTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
INT2UINT	
INT2UINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the label where the binary data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
(d)	Label for storing the converted binary data	—	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Point

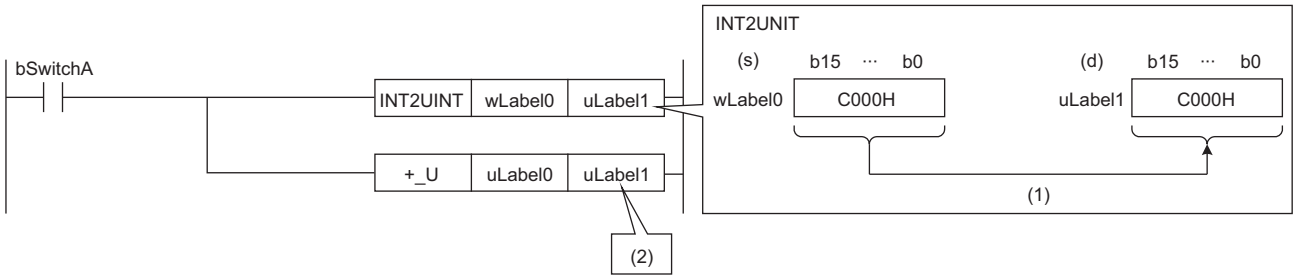
The INT2UINT(P) instruction is used in programming using labels. The purpose of using this instruction is to match the data type of the specified label with the data type that can be specified by the instruction operand. In programming using devices, use of the INT2UINT(P) instruction is not required.

Processing details

- These instructions convert the 16-bit signed binary data (ANY16_S) in the label specified by (s) to 16-bit unsigned binary data (ANY16_U), and store the converted data in the label specified by (d).
- The following figure shows a program example using the INT2UINT(P) instruction.

Ex.

The +_U instruction requires ANY16_U to be specified by the operand, and therefore, before the +_U instruction is executed, the INT2UINT instruction is used to convert wLabel0 of ANY16_S to uLabel1 of ANY16_U.
The value in wLabel0 is stored in uLabel1 as is.



bSwitchA: Bit
wLabel0: Word [signed]
uLabel0, uLabel1: Word [unsigned]/bit string [16 bits]
(1) The value is stored as is.
(2) The data type of the value is converted to the one of the operand in the +_U instruction, and the operation starts.

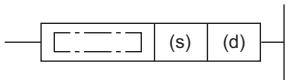
Operation error

There is no operation error.

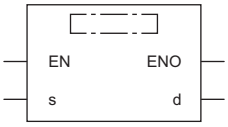
Converting 16-bit signed binary data to 32-bit signed binary data

INT2DINT(P)

These instructions convert the specified 16-bit signed binary data to 32-bit signed binary data.

Ladder	ST
	ENO:=INT2DINT(EN,s,d); ENO:=INT2DINTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
INT2DINT	
INT2DINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the device where the binary data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
(d)	Start device for storing the converted binary data	—	32-bit signed binary	ANY32_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

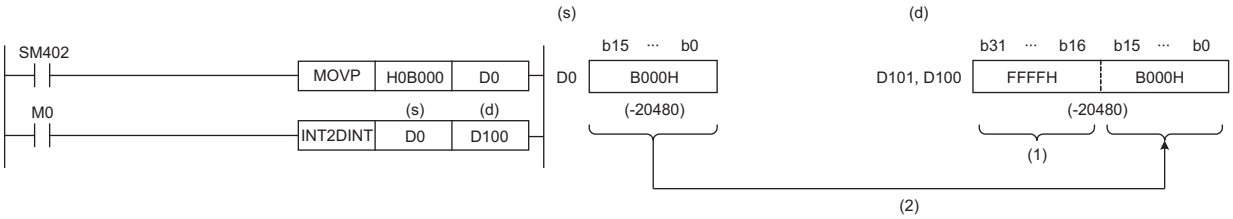
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 16-bit signed binary data in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d).

Ex.

When M0 turns on, the 16-bit signed binary data stored in D0 is converted to 32-bit signed binary data, and the converted data is stored in D100 and D101.



(1) The most significant bit of data before conversion is stored.

(2) Data before conversion is stored in the lower 16 bits.

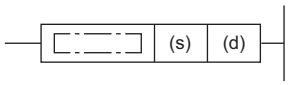
Operation error

There is no operation error.

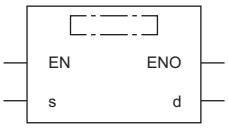
Converting 16-bit signed binary data to 32-bit unsigned binary data

INT2UDINT(P)

These instructions convert the specified 16-bit signed binary data to 32-bit unsigned binary data.

Ladder	ST
	ENO:=INT2UDINT(EN,s,d); ENO:=INT2UDINTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
INT2UDINT	
INT2UDINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the device where the binary data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
(d)	Start device for storing the converted binary data	—	32-bit unsigned binary	ANY32_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

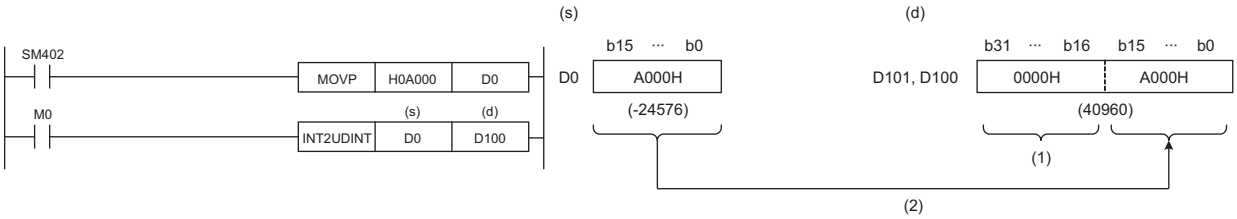
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 16-bit signed binary data in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d).

Ex.

When M0 turns on, the 16-bit signed binary data stored in D0 is converted to 32-bit unsigned binary data, and the converted data is stored in D100 and D101.



(1) The value, 0, is stored.

(2) Data before conversion is stored in the lower 16 bits.

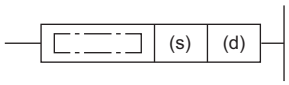
Operation error

There is no operation error.

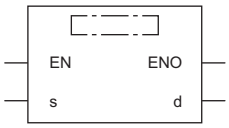
Converting 16-bit unsigned binary data to 16-bit signed binary data

UINT2INT(P)

These instructions convert the specified 16-bit unsigned binary data to 16-bit signed binary data.

Ladder	ST
	ENO:=UINT2INT(EN,s,d); ENO:=UINT2INTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UINT2INT	
UINT2INTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the label where the binary data is stored	0 to 65535	16-bit unsigned binary	ANY16_U
(d)	Label for storing the converted binary data	—	16-bit signed binary	ANY16_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Point

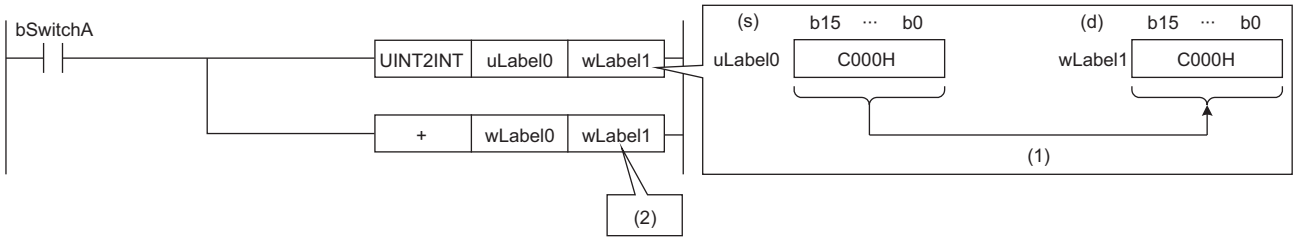
The UINT2INT(P) instruction is used in programming using labels. The purpose of using this instruction is to match the data type of the specified label with the data type that can be specified by the instruction operand. In programming using devices, use of the UINT2INT(P) instruction is not required.

Processing details

- These instructions convert the 16-bit signed binary data (ANY16_U) in the label specified by (s) to 16-bit unsigned binary data (ANY16_S), and store the converted data in the label specified by (d).
- The following figure shows a program example using the UINT2INT(P) instruction.

Ex.

The + instruction requires ANY16_S to be specified by the operand, and therefore, before the + instruction is executed, the UINT2INT instruction is used to convert uLabel0 of ANY16_U to wLabel1 of ANY16_S.
The value in uLabel0 is stored in wLabel1 as is.



bSwitchA: Bit
wLabel0, wLabel1: Word [signed]
uLabel0: Word [unsigned]/bit string [16 bits]
(1) The value is stored as is.
(2) The data type of the value is converted to the one of the operand in the + instruction, and the operation starts.

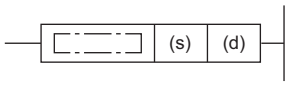
Operation error

There is no operation error.

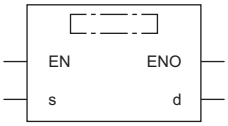
Converting 16-bit unsigned binary data to 32-bit signed binary data

UINT2DINT(P)

These instructions convert the specified 16-bit unsigned binary data to 32-bit signed binary data.

Ladder	ST
	ENO:=UINT2DINT(EN,s,d); ENO:=UINT2DINTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UINT2DINT	
UINT2DINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the device where the binary data is stored	0 to 65535	16-bit unsigned binary	ANY16_U
(d)	Start device for storing the converted binary data	—	32-bit signed binary	ANY32_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

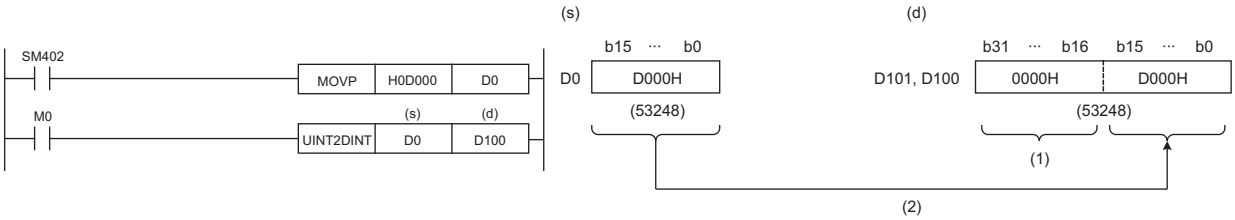
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 16-bit unsigned binary data in the device specified by (s) to 32-bit signed binary data, and store the converted data in the device specified by (d).

Ex.

When M0 turns on, the 16-bit unsigned binary data stored in D0 is converted to 32-bit signed binary data, and the converted data is stored in D100 and D101.



(1) The value, 0, is stored.

(2) Data before conversion is stored in the lower 16 bits.

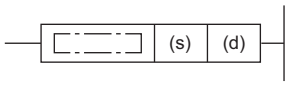
Operation error

There is no operation error.

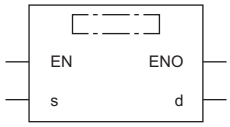
Converting 16-bit unsigned binary data to 32-bit unsigned binary data

UINT2UDINT(P)

These instructions convert the specified 16-bit unsigned binary data to 32-bit unsigned binary data.

Ladder	ST
	ENO:=UINT2UDINT(EN,s,d); ENO:=UINT2UDINTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UINT2UDINT	
UINT2UDINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the device where the binary data is stored	0 to 65535	16-bit unsigned binary	ANY16_U
(d)	Start device for storing the converted binary data	—	32-bit unsigned binary	ANY32_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

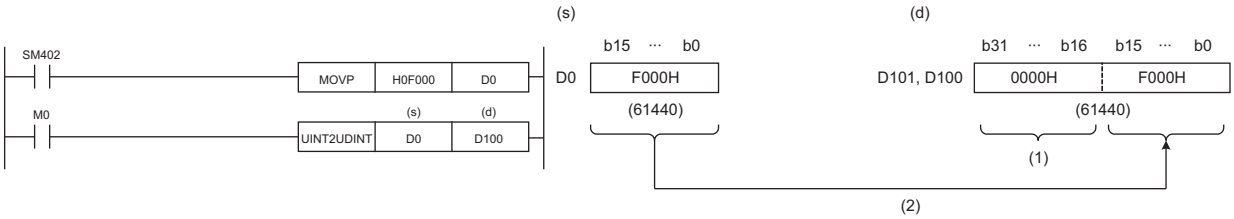
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 16-bit unsigned binary data in the device specified by (s) to 32-bit unsigned binary data, and store the converted data in the device specified by (d).

Ex.

When M0 turns on, the 16-bit unsigned binary data stored in D0 is converted to 32-bit unsigned binary data, and the converted data is stored in D100 and D101.



(1) The value, 0, is stored.

(2) Data before conversion is stored in the lower 16 bits.

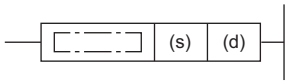
Operation error

There is no operation error.

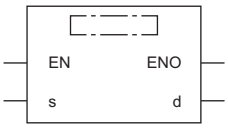
Converting 32-bit signed binary data to 16-bit signed binary data

DINT2INT(P)

These instructions convert the specified 32-bit signed binary data to 16-bit signed binary data.

Ladder	ST
	ENO:=DINT2INT(EN,s,d); ENO:=DINT2INTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DINT2INT	
DINT2INTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the start device where the binary data is stored	-32768 to 32767	32-bit signed binary	ANY32_S
(d)	Device for storing the converted binary data	—	16-bit signed binary	ANY16_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

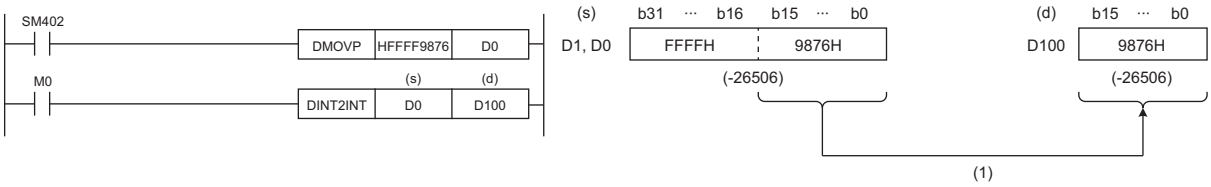
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the 32-bit signed binary data in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d).

Ex.

When M0 turns on, the 32-bit signed binary data stored in D0 and D1 is converted to 16-bit signed binary data, and the converted data is stored in D100.



(1) Data before conversion is stored in the lower 16 bits.

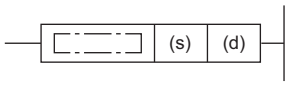
Operation error

Error code (SD0)	Description
3281H	The 32-bit signed binary data in the device specified by (s) is out of the range, -32768 to 32767.

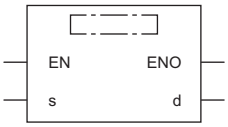
Converting 32-bit signed binary data to 16-bit unsigned binary data

DINT2UINT(P)

These instructions convert the specified 32-bit signed binary data to 16-bit unsigned binary data.

Ladder	ST
	ENO:=DINT2UINT(EN,s,d); ENO:=DINT2UINTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DINT2UINT	
DINT2UINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the start device where the binary data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
(d)	Device for storing the converted binary data	—	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

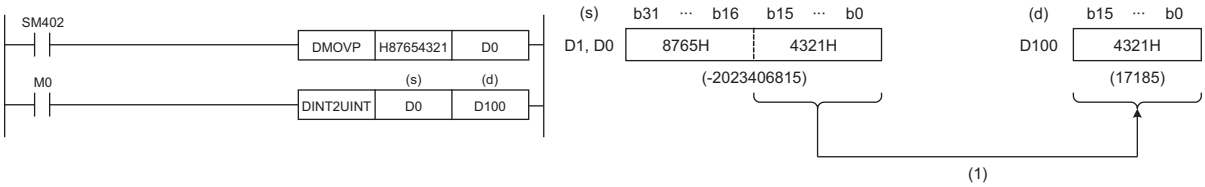
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the 32-bit signed binary data in the device specified by (s) to 16-bit unsigned binary data, and store the converted data in the device specified by (d).

Ex.

When M0 turns on, the 32-bit signed binary data stored in D0 and D1 is converted to 16-bit unsigned binary data, and the converted data is stored in D100.



(1) Data before conversion is stored in the lower 16 bits.

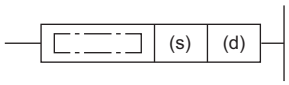
Operation error

There is no operation error.

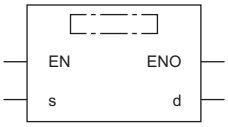
Converting 32-bit signed binary data to 32-bit unsigned binary data

DINT2UDINT(P)

These instructions convert the specified 32-bit signed binary data to 32-bit unsigned binary data.

Ladder	ST
	ENO:=DINT2UDINT(EN,s,d); ENO:=DINT2UDINTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DINT2UDINT	
DINT2UDINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the label where the binary data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
(d)	Label for storing the converted binary data	—	32-bit unsigned binary	ANY32_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Point

The DINT2UDINT(P) instruction is used in programming using labels. The purpose of using this instruction is to match the data type of the specified label with the data type that can be specified by the instruction operand.

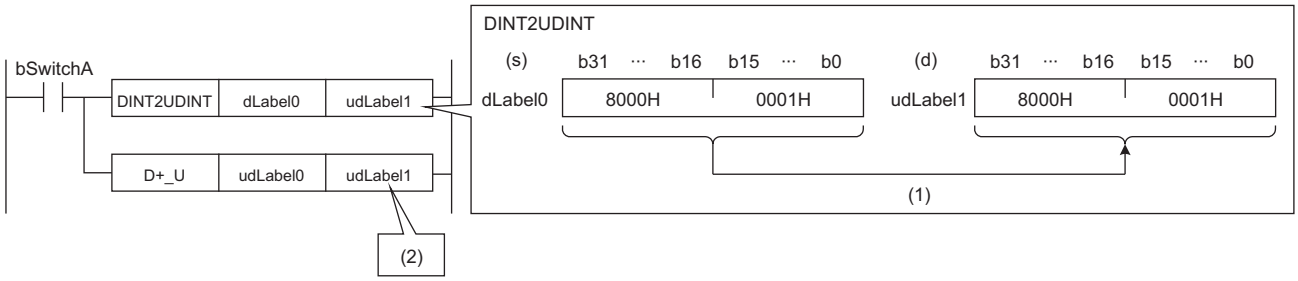
In programming using devices, use of the DINT2UDINT(P) instruction is not required.

Processing details

- These instructions convert the 32-bit signed binary data (ANY32_S) in the label specified by (s) to 32-bit unsigned binary data (ANY32_U), and store the converted data in the label specified by (d).
- The following figure shows a program example using the DINT2UDINT(P) instruction.

Ex.

The D+_U instruction requires ANY32_U to be specified by the operand, and therefore, before the D+_U instruction is executed, the DINT2UDINT instruction is used to convert dLabel0 of ANY32_S to udLabel1 of ANY32_U.
The value in dLabel0 is stored in udLabel1 as is.



bSwitchA: Bit
dLabel0: Double word [signed]
udLabel0, udLabel1: Double word [unsigned]/bit string [32 bits]
(1) The value is stored as is.
(2) The data type of the value is converted to the one of the operand in the D+_U instruction, and the operation starts.

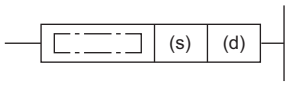
Operation error

There is no operation error.

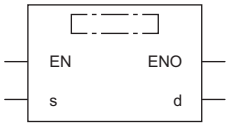
Converting 32-bit unsigned binary data to 16-bit signed binary data

UDINT2INT(P)

These instructions convert the specified 32-bit unsigned binary data to 16-bit signed binary data.

Ladder	ST
	ENO:=UDINT2INT(EN,s,d); ENO:=UDINT2INTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UDINT2INT	
UDINT2INTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the start device where the binary data is stored	0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	Device for storing the converted binary data	—	16-bit signed binary	ANY16_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

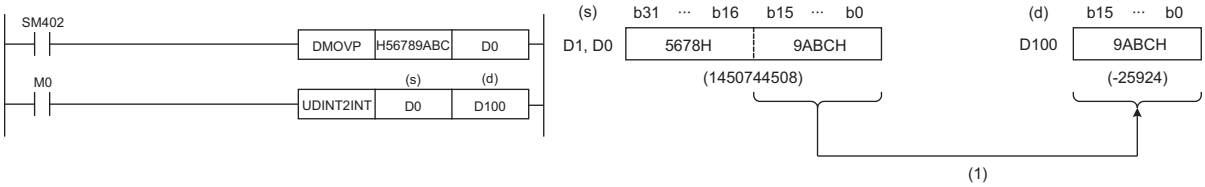
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the 32-bit unsigned binary data in the device specified by (s) to 16-bit signed binary data, and store the converted data in the device specified by (d).

Ex.

When M0 turns on, the 32-bit unsigned binary data stored in D0 and D1 is converted to 16-bit signed binary data, and the converted data is stored in D100.



(1) Data before conversion is stored in the lower 16 bits.

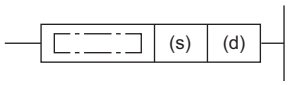
Operation error

There is no operation error.

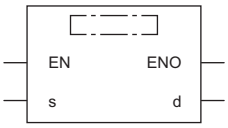
Converting 32-bit unsigned binary data to 16-bit unsigned binary data

UDINT2UINT(P)

These instructions convert the specified 32-bit unsigned binary data to 16-bit unsigned binary data.

Ladder	ST
	ENO:=UDINT2UINT(EN,s,d); ENO:=UDINT2UINTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UDINT2UINT	
UDINT2UINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the start device where the binary data is stored	0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	Device for storing the converted binary data	—	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

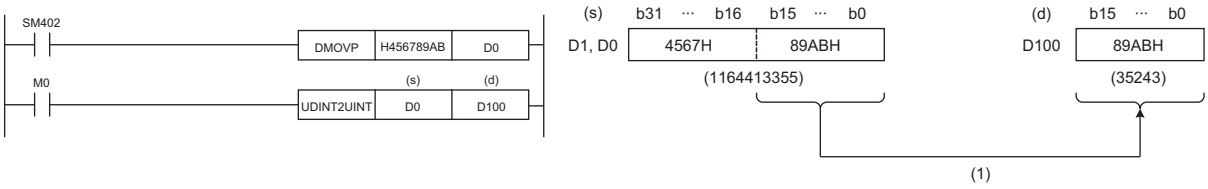
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the 32-bit unsigned binary data in the device specified by (s) to 16-bit unsigned binary data, and store the converted data in the device specified by (d).

Ex.

When M0 turns on, the 32-bit unsigned binary data stored in D0 and D1 is converted to 16-bit unsigned binary data, and the converted data is stored in D100.



(1) Data before conversion is stored in the lower 16 bits.

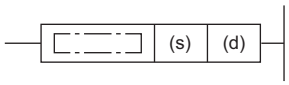
Operation error

There is no operation error.

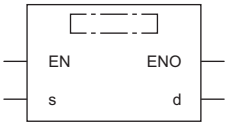
Converting 32-bit unsigned binary data to 32-bit signed binary data

UDINT2DINT(P)

These instructions convert the specified 32-bit unsigned binary data to 32-bit signed binary data.

Ladder	ST
	ENO:=UDINT2DINT(EN,s,d); ENO:=UDINT2DINTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UDINT2DINT	
UDINT2DINTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the label where the binary data is stored	0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	Label for storing the converted binary data	—	32-bit signed binary	ANY32_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Point

The UDINT2DINT(P) instruction is used in programming using labels. The purpose of using this instruction is to match the data type of the specified label with the data type that can be specified by the instruction operand.

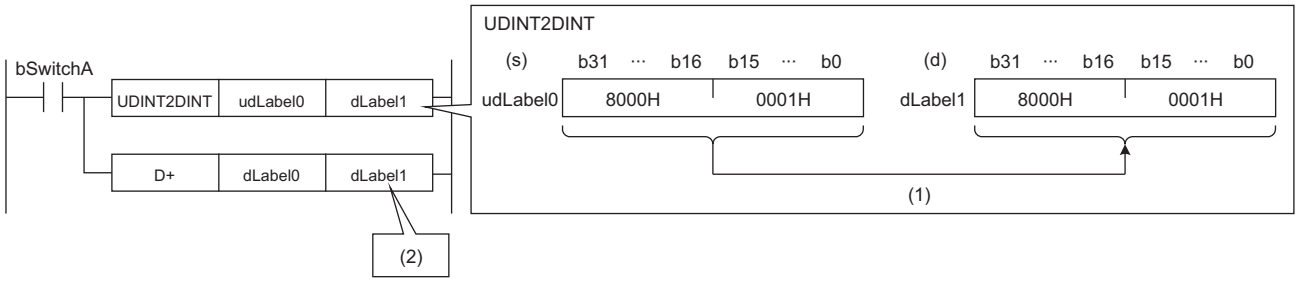
In programming using devices, use of the UDINT2DINT(P) instruction is not required.

Processing details

- These instructions convert the 32-bit signed binary data (ANY32_U) in the label specified by (s) to 32-bit unsigned binary data (ANY32_S), and store the converted data in the label specified by (d).
- The following figure shows a program example using the UDINT2DINT(P) instruction.

Ex.

The D+ instruction requires ANY32_S to be specified by the operand, and therefore, before the D+ instruction is executed, the UDINT2DINT instruction is used to convert udLabel0 of ANY32_U to dLabel1 of ANY32_S.
The value in udLabel0 is stored in dLabel1 as is.



bSwitchA: Bit
dLabel0, dLabel1: Double word [signed]
udLabel0: Double word [unsigned]/bit string [32 bits]
(1) The value is stored as is.
(2) The data type of the value is converted to the one of the operand in the D+ instruction, and the operation starts.

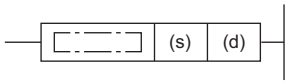
Operation error

There is no operation error.

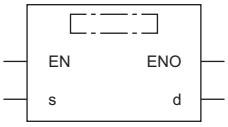
Converting 16-bit binary data to Gray code data

GRY(P)(_U)

These instructions convert the specified 16-bit binary data to 16-bit binary Gray code data.

Ladder	ST	
	ENO:=GRY(EN,s,d); ENO:=GRYP(EN,s,d);	ENO:=GRY_U(EN,s,d); ENO:=GRYP_U(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GRY GRY_U	
GRYP GRYP_U	

Setting data

■Description, range, data type

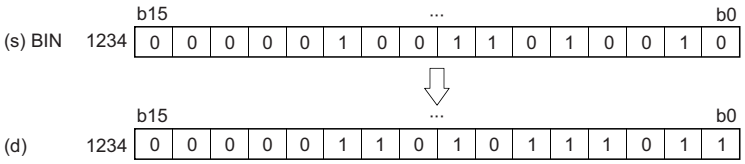
Operand		Description	Range	Data type	Data type (label)
(s)	GRY(P)	Binary data or the device where the binary data is stored	0 to 32767	16-bit signed binary	ANY16_S
	GRY(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	GRY(P)	Device for storing the converted Gray code data	—	16-bit signed binary	ANY16_S
	GRY(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the 16-bit binary data in the device specified by (s) to 16-bit binary Gray code data, and store the converted data in the device specified by (d).



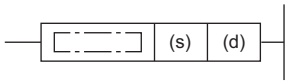
Operation error

Error code (SD0)	Description
3281H	When the GRY(P) instruction is used, the value in the device specified by (s) is out of the range, 0 to 32767.

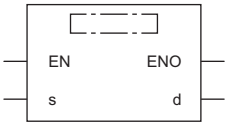
Converting 32-bit binary data to Gray code data

DGRY(P)(_U)

These instructions convert the specified 32-bit binary data to 32-bit binary Gray code data.

Ladder	ST	
	ENO:=DGRY(EN,s,d); ENO:=DGRYP(EN,s,d);	ENO:=DGRY_U(EN,s,d); ENO:=DGRYP_U(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DGRY DGRY_U	
DGRYP DGRYP_U	

Setting data

■Description, range, data type

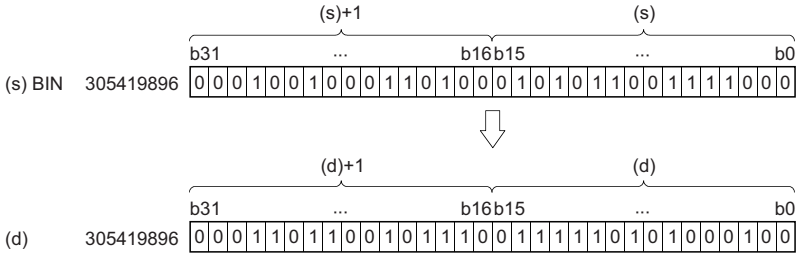
Operand		Description	Range	Data type	Data type (label)
(s)	DGRY(P)	Binary data or the start device where the binary data is stored	0 to 2147483647	32-bit signed binary	ANY32_S
	DGRY(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	DGRY(P)	Start device for storing the converted Gray code data	—	32-bit signed binary	ANY32_S
	DGRY(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 32-bit binary data in the device specified by (s) to 32-bit binary Gray code data, and store the converted data in the device specified by (d).



(s)+1: Upper 16 bits
(s): Lower 16 bits

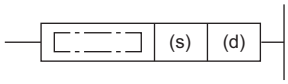
Operation error

Error code (SD0)	Description
3281H	When the DGRY(P) instruction is used, the value in the device specified by (s) is out of the range, 0 to 2147483647.

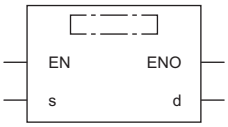
Converting 16-bit binary Gray code data to 16-bit binary data

GBIN(P)(_U)

These instructions convert the specified 16-bit binary Gray code data to 16-bit binary data.

Ladder	ST	
	ENO:=GBIN(EN,s,d); ENO:=GBINP(EN,s,d);	ENO:=GBIN_U(EN,s,d); ENO:=GBINP_U(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GBIN GBIN_U	
GBINP GBINP_U	

Setting data

■Description, range, data type

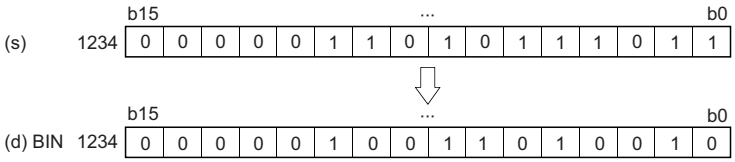
Operand		Description	Range	Data type	Data type (label)
(s)	GBIN(P)	Gray code data or the device where the Gray code data is stored	0 to 32767	16-bit signed binary	ANY16_S
	GBIN(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	GBIN(P)	Device for storing the converted binary data	—	16-bit signed binary	ANY16_S
	GBIN(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions convert the 16-bit binary Gray code data in the device specified by (s) to 16-bit binary data, and store the converted data in the device specified by (d).



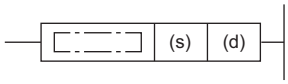
Operation error

Error code (SD0)	Description
3281H	When the GBIN(P) instruction is used, the value specified by (s) is out of the range, 0 to 32767.

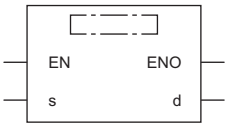
Converting 32-bit binary Gray code data to 32-bit binary data

DGBIN(P)(_U)

These instructions convert the specified 32-bit binary Gray code data to 32-bit binary data.

Ladder	ST	
	ENO:=DGBIN(EN,s,d); ENO:=DGBINP(EN,s,d);	ENO:=DGBIN_U(EN,s,d); ENO:=DGBINP_U(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DGBIN DGBIN_U	
DGBINP DGBINP_U	

Setting data

■Description, range, data type

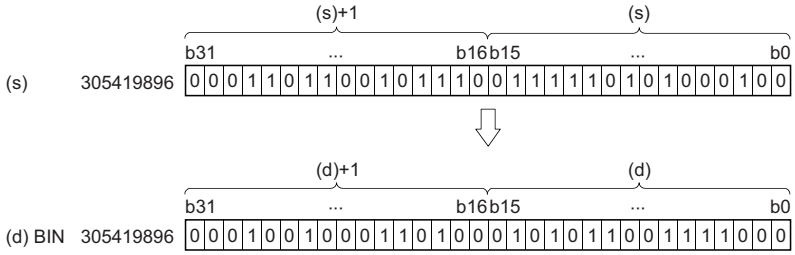
Operand		Description	Range	Data type	Data type (label)
(s)	DGBIN(P)	Gray code data or the start device where the Gray code data is stored	0 to 2147483647	32-bit signed binary	ANY32_S
	DGBIN(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	DGBIN(P)	Start device for storing the converted binary data	—	32-bit signed binary	ANY32_S
	DGBIN(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 32-bit binary Gray code data in the device specified by (s) to 32-bit binary data, and store the converted data in the device specified by (d).



(s)+1, (d)+1: Upper 16 bits
(s), (d): Lower 16 bits

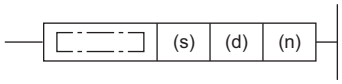
Operation error

Error code (SD0)	Description
3281H	When the DGBIN(P) instruction is used, the value in the device specified by (s) is out of the range, 0 to 2147483647.

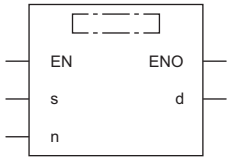
Converting 16-bit binary data block to BCD 4-digit data block

BKBCD(P)

These instructions convert (n) points of binary data (0 to 9999) starting from the specified device to BCD data.

Ladder	ST
	ENO:=BKBCD(EN,s,n,d); ENO:=BKBCDP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BKBCD	
BKBCDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the binary data is stored	—	16-bit signed binary	ANY16*1
(d)	Start device for storing the converted BCD data	—	BCD 4-digit	ANY16*1
(n)	Number of variables	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

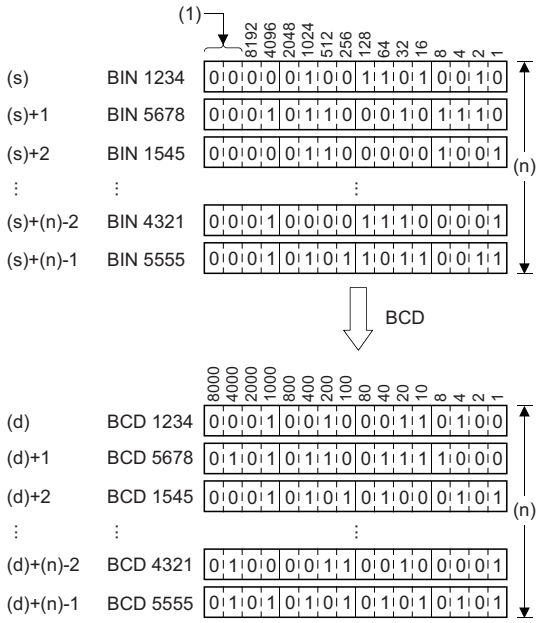
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions convert the (n) points of 16-bit binary data (0 to 9999) starting from the device specified by (s) to BCD data, and store the converted data in the device specified by (d) and later.



(1) Set 0s.

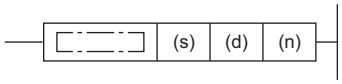
Operation error

Error code (SD0)	Description
2821H	The device ranges specified by (s) and (d) are overlapping.
3281H	The (n) points of data starting from the device specified by (s) is out of the range, 0 to 9999.

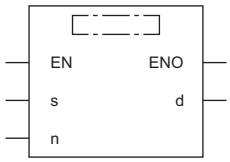
Converting BCD 4-digit block data to 16-bit binary block data

BKBIN(P)

These instructions convert (n) points of BCD data (0 to 9999) starting from the specified device to binary data.

Ladder	ST
	ENO:=BKBIN(EN,s,n,d); ENO:=BKBINP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BKBIN	
BKBINP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the BCD data is stored	—	BCD 4-digit	ANY16*1
(d)	Start device for storing the converted binary data	—	16-bit signed binary	ANY16*1
(n)	Number of variables	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

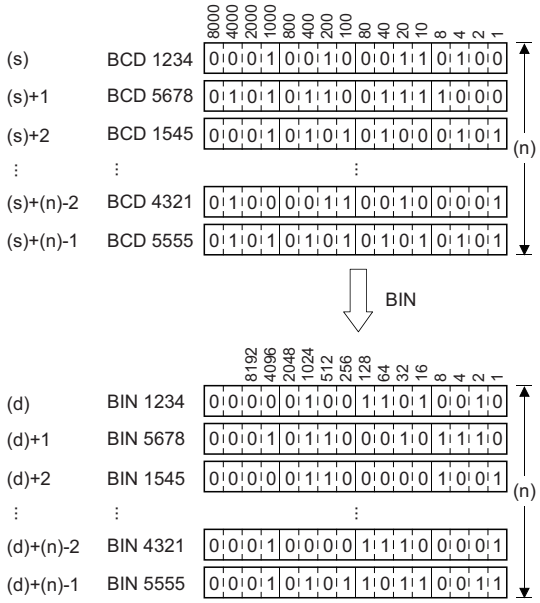
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions convert the (n) points of BCD data (0 to 9999) starting from the device specified by (s) to 16-bit binary data, and store the converted data in the device specified by (d) and later.



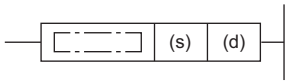
Operation error

Error code (SD0)	Description
2821H	The device ranges specified by (s) and (d) are overlapping.
3281H	The (n) points of data starting from the device specified by (s) is out of the range, 0 to 9999.

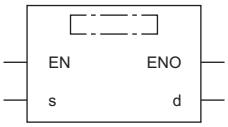
Converting decimal ASCII data to 16-bit binary data

DABIN(P)(_U)

These instructions convert decimal ASCII data to 16-bit binary data.

Ladder	ST	
	ENO:=DABIN(EN,s,d); ENO:=DABINP(EN,s,d);	ENO:=DABIN_U(EN,s,d); ENO:=DABINP_U(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DABIN DABIN_U	
DABINP DABINP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)		ASCII data to be converted to binary data or the start device where the ASCII data is stored	—	String	ANYSTRING_SINGLE
(d)	DABIN(P)	Device for storing the converted binary data	—	16-bit signed binary	ANY16_S
	DABIN(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

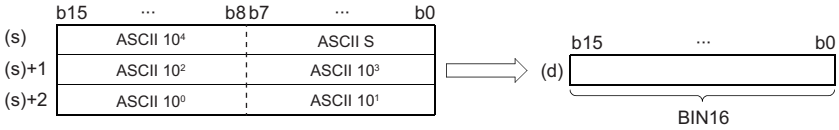
Processing details

- These instructions convert the decimal ASCII data in the device specified by (s) and later to 16-bit binary data, and store the converted data in the device specified by (d).
- The setting method of the decimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705	Setting method of (s)	Reference
Off	Set (s) with a fixed number of digits (a sign + 5 digits in the numeric part).	Page 451 Setting method of (s) for when SM705 (Number of conversion digits selection) is off
On	Set (s) with a desired number of digits (maximum: a sign + 5 digits in the numeric part).	Page 452 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

■Setting method of (s) for when SM705 (Number of conversion digits selection) is off

Set decimal ASCII data with the fixed number of digits in (s) to (s)+2.

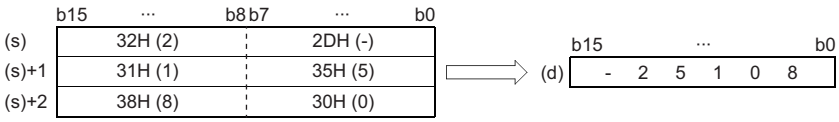


ASCII S: Sign data of ASCII code
ASCII 10^4: Ten-thousands place of ASCII code
ASCII 10^3: Thousands place of ASCII code
ASCII 10^2: Hundreds place of ASCII code
ASCII 10^1: Tens place of ASCII code
ASCII 10^0: Ones place of ASCII code

- The ASCII data in the device specified by (s) to (s)+2 is within the range from -32768 to 32767 for the DABIN(P) instruction, and it is within the range from 0 to 65535 for the DABIN(P)_U instruction.
- The data of (s)+3 or later is ignored.
- As sign data, set 20H (space) when the ASCII data is positive, and set 2DH (-) when the data is negative. (If a value other than 20H and 2DH is set, the data will be processed as positive data.)
- A value from 30H to 39H can be set in each place of ASCII code.
- If a value 20H or 00H is set in each place of ASCII code, the value will be processed as 30H.

Ex.

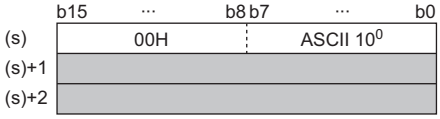
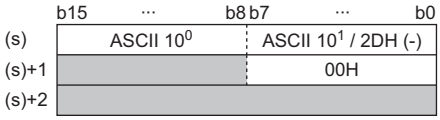
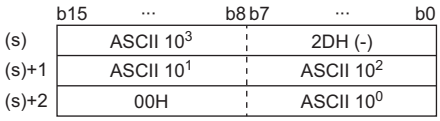
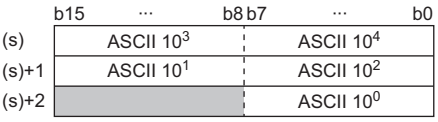
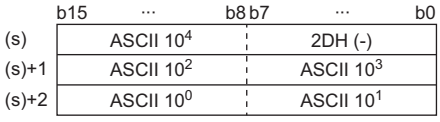
"-25108" is set in (s) when the DABIN(P) instruction is used



■Setting method of (s) for when SM705 (Number of conversion digits selection) is on

Set decimal ASCII data with a desired number of digits (including 00H (NULL code)) in (s). Note that 00H (NULL code) is not required to be set if the integral part has the maximum number of digits (5 digits).

The following table lists the setting method of (s).

<table>
<tr><th>Value to be set in (s)</th><th>Data of (s) to (s)+2</th><th>Value to be set in (s)</th><th>Data of (s) to (s)+2</th></tr>
<tr><td>• 0
• Positive value (1 digit in numeric part)</td><td>• Set 00H in the upper byte of (s).
• The data of (s)+1 or later is ignored.
</td><td>• Positive value (2 digits in numeric part)
• Negative value (1 digit in numeric part)</td><td>• Set 00H in the lower byte of (s)+1.
• The data of the upper byte of (s)+1 or later is ignored.
</td></tr>
<tr><td colspan="4">⋮</td></tr>
<tr><td>• Negative value (4 digits in numeric part)</td><td>• Set 00H in the upper byte of (s)+2.
</td><td>• Positive value (5 digits in numeric part)</td><td>• The data of the upper byte of (s)+2 or later is ignored. Since the number of digits is the maximum, 00H is not required to be set.
</td></tr>
<tr><td>• Negative value (5 digits in numeric part)</td><td>• The data of (s)+3 or later is ignored. Since the number of digits is the maximum, 00H is not required to be set.
</td><td colspan="2">ASCII 10^0: Ones place of ASCII code
ASCII 10^1: Tens place of ASCII code
⋮
ASCII 10^4: Ten-thousands place of ASCII code</td></tr>
</table>

- The ASCII data in the device specified by (s) to (s)+2 is within the range from -32768 to 32767 for the DABIN(P) instruction, and it is within the range from 0 to 65535 for the DABIN(P)_U instruction.
- Set 2DH (-) to lower byte of (s)+0 as sign data when the ASCII data is negative. Set an ASCII code of the uppermost digit instead of setting sign data when the ASCII data is 0 or positive.
- A value from 30H to 39H can be set in each place of ASCII code.
- If the value is positive and the numeric part has 5 digits, the data of the upper byte of (s)+2 or later is ignored. If the value is negative and the numeric part has 5 digits, the data of (s)+3 or later is ignored.
- If a value 20H is set in each place of ASCII code, the value is processed as 30H. If a value 00H is set, the value is processed as the end of the decimal ASCII data.
- In the following cases, 0 is stored.
  - The first character is 00H (NULL).
  - The first character is 2DH (-) and the second character is 00H (NULL).

Operation error

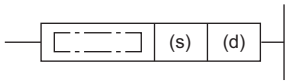
Error code (SD0)	Description
3281H	Invalid data that cannot be converted are input in (s) to (s)+2. • The ASCII code of the first character is other than 2DH, 30H to 39H, 20H, and 00H.*1 • The ASCII code of the second character or later is other than 30H to 39H, 20H, and 00H. • When the DABIN(P) instruction is used, ASCII data is out of the range from -32768 to 32767. • When the DABIN(P)_U instruction is used, ASCII data is out of the range from 0 to 65535.

*1 When SM705 (Number of conversion digits selection) is off, no error is detected no matter what value is set for the ASCII code of the first character.

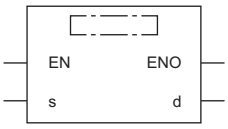
Converting decimal ASCII data to 32-bit binary data

DDABIN(P)(_U)

These instructions convert decimal ASCII data to 32-bit binary data.

Ladder	ST	
	ENO:=DDABIN(EN,s,d); ENO:=DDABINP(EN,s,d);	ENO:=DDABIN_U(EN,s,d); ENO:=DDABINP_U(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DDABIN DDABIN_U	
DDABINP DDABINP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)		ASCII data to be converted to binary data or the start device where the ASCII data is stored	—	String	ANYSTRING_SINGLE
(d)	DDABIN(P)	Start device for storing the conversion result	—	32-bit signed binary	ANY32_S
	DDABIN(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

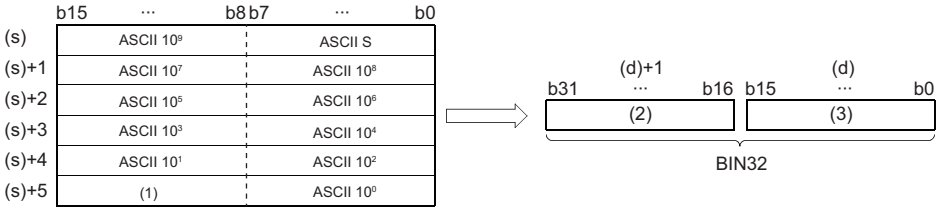
Processing details

- These instructions convert the decimal ASCII data in the device areas specified by (s) and later to 32-bit binary data, and store the converted data in the device specified by (d).
- The setting method of the decimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705	Setting method of (s)	Reference
Off	Set (s) with a fixed number of digits (a sign + 10 digits in the numeric part).	Page 454 Setting method of (s) for when SM705 (Number of conversion digits selection) is off
On	Set (s) with a desired number of digits (maximum: a sign + 10 digits in the numeric part).	Page 455 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

■Setting method of (s) for when SM705 (Number of conversion digits selection) is off

Set decimal ASCII data with the fixed number of digits in (s) to (s)+5.

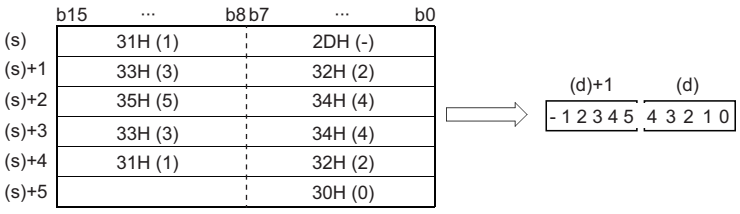


ASCII S: Sign data of ASCII code
ASCII 10^0: Ones place of ASCII code
ASCII 10^1: Tens place of ASCII code
ASCII 10^2: Hundreds place of ASCII code
ASCII 10^3: Thousands place of ASCII code
ASCII 10^4: Ten-thousands place of ASCII code
ASCII 10^5: Hundred-thousands place of ASCII code
ASCII 10^6: Millions place of ASCII code
ASCII 10^7: Ten-millions place of ASCII code
ASCII 10^8: Hundred-millions place of ASCII code
ASCII 10^9: Billions place of ASCII code
(1) Ignored.
(2) Upper 16 bits
(3) Lower 16 bits

- The ASCII data in the device specified by (s) to (s)+5 is within the range from -2147483648 to 2147483647 for the DDABIN(P) instruction, and it is within the range from 0 to 4294967295 for the DDABIN(P)_U instruction. Any data stored in the upper bytes in the device specified by (s)+5 and data in the device specified by (s)+6 and later are ignored.
- As sign data, set 20H if the ASCII data is positive, and set 2DH if the data is negative. (If a value other than 20H and 2DH is set, the data will be processed as positive data.)
- A value from 30H to 39H can be set in each place of ASCII code.
- If a value 20H or 00H is set in each place of ASCII code, the value will be processed as 30H.

Ex.

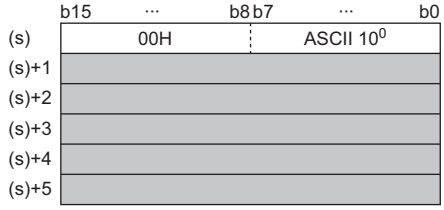
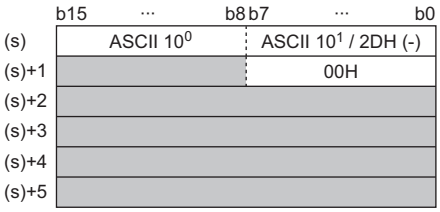
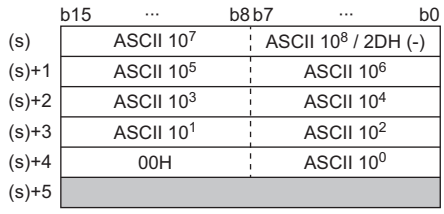
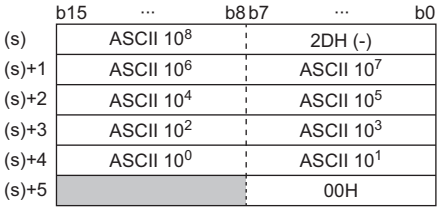
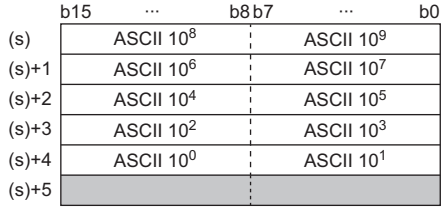
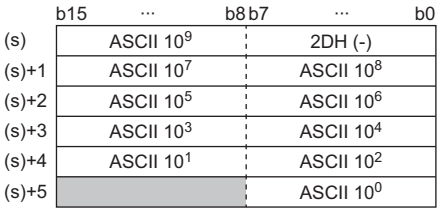
"-1234543210" is set in (s) when the DDABIN(P) instruction is used



■Setting method of (s) for when SM705 (Number of conversion digits selection) is on

Set decimal ASCII data with a desired number of digits (including 00H (NULL code)) in (s). Note that 00H (NULL code) is not required to be set if the integral part has the maximum number of digits (10 digits).

The following table lists the setting method of (s).

<table>
<tr><th>Value to be set in (s)</th><th>Data of (s) to (s)+5</th><th>Value to be set in (s)</th><th>Data of (s) to (s)+5</th></tr>
<tr><td>• 0
• Positive value (1 digit in numeric part)</td><td>• Set 00H in the upper byte of (s).
• The data of (s)+1 or later is ignored.
</td><td>• Positive value (2 digits in numeric part)
• Negative value (1 digit in numeric part)</td><td>• Set 00H in the lower byte of (s)+1.
• The data of the upper byte of (s)+1 or later is ignored.
</td></tr>
<tr><td colspan="4">⋮</td></tr>
<tr><td>• Positive value (9 digits in numeric part)
• Negative value (8 digits in numeric part)</td><td>• Set 00H in the upper byte of (s)+4.
• (s)+5 is ignored.
</td><td>• Negative value (9 digits in numeric part)</td><td>• Set 00H in the lower byte of (s)+5.
• The data of the upper byte of (s)+5 is ignored.
</td></tr>
<tr><td>• Positive value (10 digits in numeric part)</td><td>• (s)+5 is ignored. Since the number of digits is the maximum, 00H is not required to be set.
</td><td>• Negative value (10 digits in numeric part)</td><td>• The data of the upper byte of (s)+5 is ignored. Since the number of digits is the maximum, 00H is not required to be set.
</td></tr>
</table>

ASCII 10^0: Ones place of ASCII code
ASCII 10^1: Tens place of ASCII code
⋮
ASCII 10^9: Billions place of ASCII code

- The ASCII data in the device specified by (s) to (s)+5 is within the range from -2147483648 to 2147483647 for the DDABIN(P) instruction, and it is within the range from 0 to 4294967295 for the DDABIN(P)_U instruction.
- Set 2DH (-) to lower byte of (s)+0 as sign data when the ASCII data is negative. Set an ASCII code of the uppermost digit instead of setting sign data when the ASCII data is 0 or positive.
- A value from 30H to 39H can be set in each place of ASCII code.
- If the value is positive and the numeric part has 10 digits, the data stored in (s)+5 or later is ignored. If the value is negative and the numeric part has 10 digits, the data stored in the upper byte of (s)+5 or later is ignored.
- If a value 20H is set in each place of ASCII code, the value is processed as 30H. If a value 00H is set, the value is processed as the end of the decimal ASCII data.
- In the following cases, "0" is stored in (d).
  - The first character is 00H (NULL).
  - The first character is 2DH (-) and the second character is 00H (NULL).

Operation error

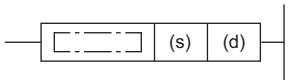
Error code (SD0)	Description
3281H	Invalid data that cannot be converted are input in (s) to (s)+5. • The ASCII code of the first character is other than 2DH, 30H to 39H, 20H, and 00H.*1 • The ASCII code of the second character or later is other than 30H to 39H, 20H, and 00H. • When the DDABIN(P) instruction is used, ASCII data is out of the range from -2147483648 to 2147483647. • When the DDABIN(P)_U instruction is used, ASCII data is out of the range from 0 to 4294967295.

*1 When SM705 (Number of conversion digits selection) is off, no error is detected no matter what value is set for the ASCII code of the first character.

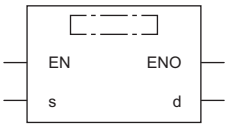
Converting hexadecimal ASCII data to 16-bit binary data

HABIN(P)

These instructions convert hexadecimal ASCII data to 16-bit binary data.

Ladder	ST
	ENO:=HABIN(EN,s,d); ENO:=HABINP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
HABIN	
HABINP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	ASCII data to be converted to binary data or the start device where the ASCII data is stored	—	String	ANYSTRING_SINGLE
(d)	Device for storing the conversion result	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

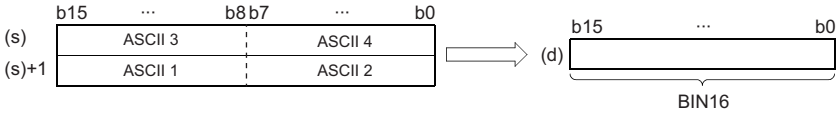
Processing details

- These instructions convert the hexadecimal ASCII data stored in the device areas specified by (s) and later to 16-bit binary data, and store the converted data in the device specified by (d).
- The setting method of the hexadecimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705	Setting method of (s)	Reference
Off	Set (s) with a fixed number of digits (4 digits).	Page 458 Setting method of (s) for when SM705 (Number of conversion digits selection) is off
On	Set (s) with a desired number of digits (maximum: 4 digits).	Page 458 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

■Setting method of (s) for when SM705 (Number of conversion digits selection) is off

- Set hexadecimal ASCII data with 4 digits (fixed) in (s) to (s)+1.

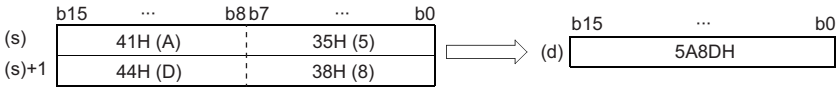


ASCII □: ASCII code (□th digit)

- The ASCII data in the device specified by (s) to (s)+1 is within the range from 0000H to FFFFH.
- The data of (s)+2 or later is ignored.
- A value from 30H to 39H and 41H to 46H can be set in each place of ASCII code.

Ex.

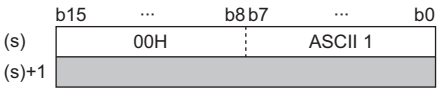
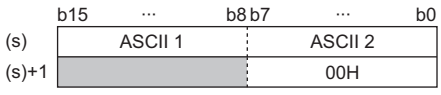
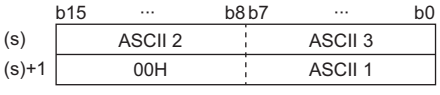
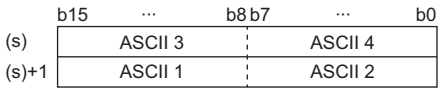
When 5A8DH is specified in (s)



■Setting method of (s) for when SM705 (Number of conversion digits selection) is on

Set hexadecimal ASCII data with a desired number of digits (including 00H (NULL code)) in (s). Note that 00H (NULL code) is not required to be set for the maximum number of digits (4 digits).

The following table lists the setting method of (s).

Value to be set in (s)	Data of (s) to (s)+1	Value to be set in (s)	Data of (s) to (s)+1
• 0H to FH	• Set 00H in the upper byte of (s)+0. • The data of (s)+1 or later is ignored. 	• 10H to FFH	• Set 00H in the lower byte of (s)+1. • The data of the upper byte of (s)+1 or later is ignored. 
• 100H to FFFH	• Set 00H in the upper byte of (s)+1. • The data of (s)+2 or later is ignored. 	• 1000H to FFFFH	• The data of (s)+2 or later is ignored. 

ASCII □: ASCII code (□th digit)

- The ASCII data in the device specified by (s) to (s)+1 is within the range from 0000H to FFFFH.
- The data of (s)+2 or later is ignored.
- A value from 30H to 39H and 41H to 46H can be set in each place of ASCII code.
- If a value 00H is set in each place of ASCII code, the value will be processed as the end of the hexadecimal ASCII data.

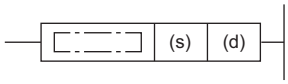
Operation error

Error code (SD0)	Description
3281H	Invalid data that cannot be converted are input in (s) to (s)+1. • A value in each place of ASCII code is other than "30H" to "39H" and "41H" to "46H".

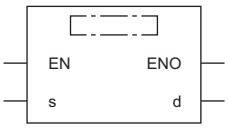
Converting hexadecimal ASCII data to 32-bit binary data

DHABIN(P)

These instructions convert hexadecimal ASCII data to 32-bit binary data.

Ladder	ST
	ENO:=DHABIN(EN,s,d); ENO:=DHABINP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DHABIN	
DHABINP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	ASCII data to be converted to binary data or the start device where the ASCII data is stored	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the conversion result	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

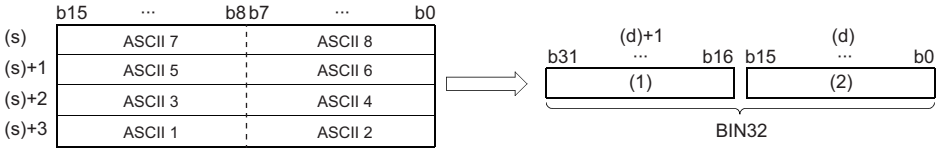
Processing details

- These instructions convert the hexadecimal ASCII data stored in the device specified by (s) and later to 32-bit binary data, and store the converted data in the device specified by (d).
- The setting method of the hexadecimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705	Setting method of (s)	Reference
Off	Set (s) with a fixed number of digits (8 digits).	Page 460 Setting method of (s) for when SM705 (Number of conversion digits selection) is off
On	Set (s) with a desired number of digits (maximum: 8 digits).	Page 461 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

■Setting method of (s) for when SM705 (Number of conversion digits selection) is off

- Set hexadecimal ASCII data with 8 digits (fixed) in (s) to (s)+3.



ASCII □: ASCII code (□th digit)

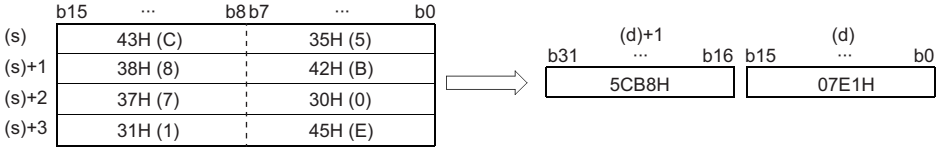
(1) Upper 16 bits

(2) Lower 16 bits

- The ASCII data in the device specified by (s) to (s)+3 is within the range from 00000000H to FFFFFFFFH.
- The data of (s)+4 or later is ignored.
- A value from 30H to 39H and 41H to 46H can be set in each place of ASCII code.

Ex.

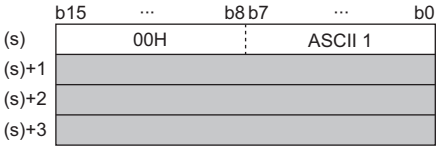
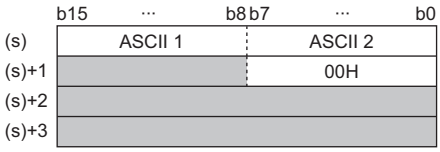
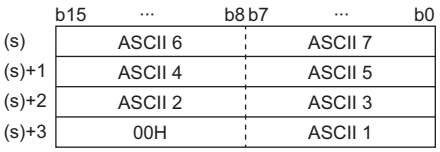
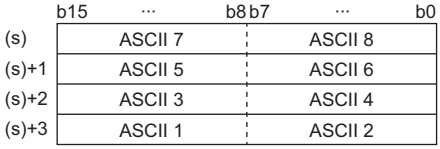
When 5CB807E1H is specified in (s)



■Setting method of (s) for when SM705 (Number of conversion digits selection) is on

Set hexadecimal ASCII data with a desired number of digits (including 00H (NULL code)) in (s). Note that 00H (NULL code) is not required to be set for the maximum number of digits (8 digits).

The following table lists the setting method of (s).

<table>
<tr><th>Value to be set in (s)</th><th>Data of (s) to (s)+3</th><th>Value to be set in (s)</th><th>Data of (s) to (s)+3</th></tr>
<tr><td>• 0H to FH</td><td>• Set 00H in the upper byte of (s).
• The data of (s)+1 or later is ignored.

</td><td>• 10H to FFH</td><td>• Set 00H in the lower byte of (s)+1.
• The data of the upper byte of (s)+1 or later is ignored.

</td></tr>
<tr><td colspan="4">⋮</td></tr>
<tr><td>• 1000000H to FFFFFFFH</td><td>• Set 00H in the upper byte of (s)+3.
• The data of (s)+4 or later is ignored.

</td><td>• 10000000H to FFFFFFFFH</td><td>• The data of (s)+4 or later is ignored.

</td></tr>
</table>

ASCII □: ASCII code (□th digit)

- The ASCII data in the device specified by (s) to (s)+3 is within the range from 00000000H to FFFFFFFFH.
- The data of (s)+4 or later is ignored.
- A value from 30H to 39H and 41H to 46H can be set in each place of ASCII code.
- If a value 00H is set in each place of ASCII code, the value will be processed as the end of the hexadecimal ASCII data.

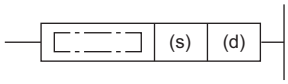
Operation error

Error code (SD0)	Description
3281H	Invalid data that cannot be converted are input in (s) to (s)+3. • A value in each place of ASCII code is other than "30H" to "39H" and "41H" to "46H".

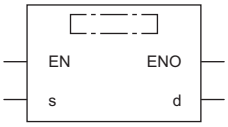
Converting decimal ASCII data to BCD 4-digit data

DABCD(P)

These instructions convert decimal ASCII data to BCD 4-digit data.

Ladder	ST
	ENO:=DABCD(EN,s,d); ENO:=DABCDP(EN,s,d)

FBD/LD



■Execution condition

Instruction	Execution condition
DABCD	
DABCDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	ASCII data to be converted to BCD data or the start device where the ASCII data is stored	—	String	ANYSTRING_SINGLE
(d)	Device for storing the conversion result	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

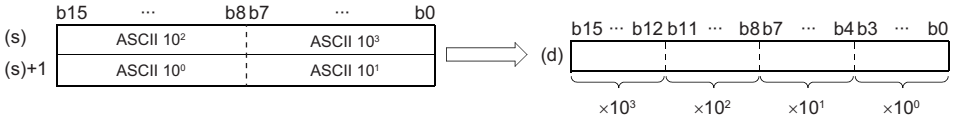
Processing details

- These instructions convert the decimal ASCII data stored in the device areas specified by (s) and later to BCD 4-digit data, and store the converted data in the device specified by (d).
- The setting method of the decimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705	Setting method of (s)	Reference
Off	Set (s) with a fixed number of digits (4 digits).	Page 463 Setting method of (s) for when SM705 (Number of conversion digits selection) is off
On	Set (s) with a desired number of digits (maximum: 4 digits).	Page 464 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

■Setting method of (s) for when SM705 (Number of conversion digits selection) is off

- Set decimal ASCII data the 4 digits (fixed) in (s) to (s)+1.

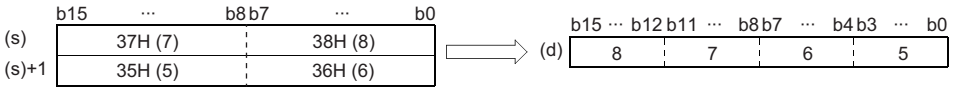


ASCII 10^3: Thousands place of ASCII code
ASCII 10^2: Hundreds place of ASCII code
ASCII 10^1: Tens place of ASCII code
ASCII 10^0: Ones place of ASCII code

- The ASCII data in the device specified by (s) to (s)+1 is within the range from 0 to 9999.
- The data of (s)+2 or later is ignored.
- A value from 30H to 39H can be set in each place of ASCII code.
- If a value 20H or 00H is set in each place of ASCII code, the value will be processed as 30H.

Ex.

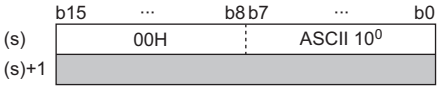
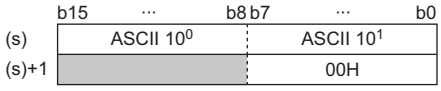
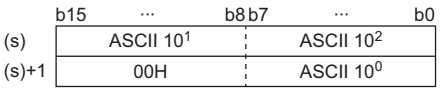
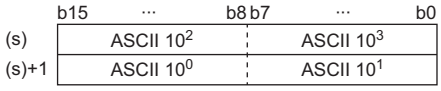
When 8765 is specified in (s)



■Setting method of (s) for when SM705 (Number of conversion digits selection) is on

Set decimal ASCII data with a desired number of digits (including 00H (NULL code)) in (s). Note that 00H (NULL code) is not required to be set for the maximum number of digits (4 digits).

The following table lists the setting method of (s).

<table>
<tr><th>Value to be set in (s)</th><th>Data of (s) to (s)+1</th><th>Value to be set in (s)</th><th>Data of (s) to (s)+1</th></tr>
<tr><td>• 0 to 9</td><td>• Set 00H in the upper byte of (s)+0.
• The data of (s)+1 or later is ignored.

</td><td>• 10 to 99</td><td>• Set 00H in the lower byte of (s)+1.
• The data of the upper byte of (s)+1 or later is ignored.

</td></tr>
<tr><td>• 100 to 999</td><td>• Set 00H in the upper byte of (s)+1.
• The data of (s)+2 or later is ignored.

</td><td>• 1000 to 9999</td><td>• The data of (s)+2 or later is ignored.

</td></tr>
</table>

$ASCII\ 10^3$: Thousands place of ASCII code
$ASCII\ 10^2$: Hundreds place of ASCII code
$ASCII\ 10^1$: Tens place of ASCII code
$ASCII\ 10^0$: Ones place of ASCII code

- The ASCII data in the device specified by (s) to (s)+1 is within the range from 0 to 9999.
- The data of (s)+2 or later is ignored.
- A value from 30H to 39H can be set in each place of ASCII code.
- If a value 20H is set in each place of ASCII code, the value is processed as 30H. If a value 00H is set, the value is processed as the end of the decimal ASCII data.

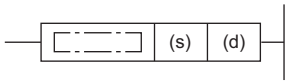
Operation error

Error code (SD0)	Description
3281H	Invalid data that cannot be converted are input in (s). • A character other than 0 to 9 exists in the data.

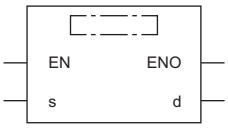
Converting decimal ASCII data to BCD 8-digit data

DDABCD(P)

These instructions convert decimal ASCII data to BCD 8-digit data.

Ladder	ST
	ENO:=DDABCD(EN,s,d); ENO:=DDABCDP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DDABCD	
DDABCDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	ASCII data to be converted to BCD data or the start device where the ASCII data is stored	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the conversion result	—	BCD 8-digit	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

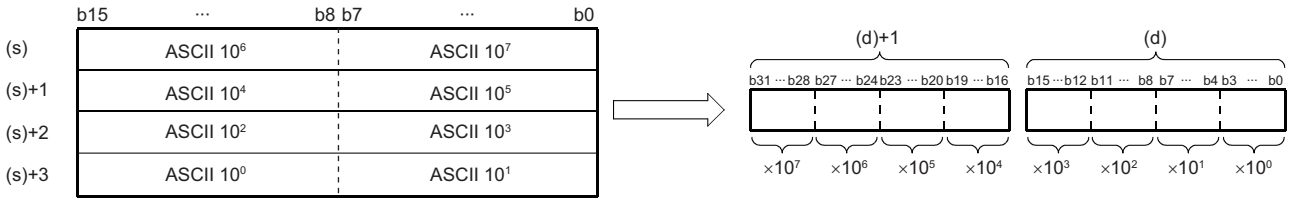
Processing details

- These instructions convert the decimal ASCII data stored in the device areas specified by (s) and later to BCD 8-digit data, and store the converted data in the device number specified by (d).
- The setting method of the decimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705	Setting method of (s)	Reference
Off	Set (s) with a fixed number of digits (8 digits).	Page 466 Setting method of (s) for when SM705 (Number of conversion digits selection) is off
On	Set (s) with a desired number of digits (maximum: 8 digits).	Page 467 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

■Setting method of (s) for when SM705 (Number of conversion digits selection) is off

- Set decimal ASCII data with 8 digits (fixed) in (s) to (s)+3.

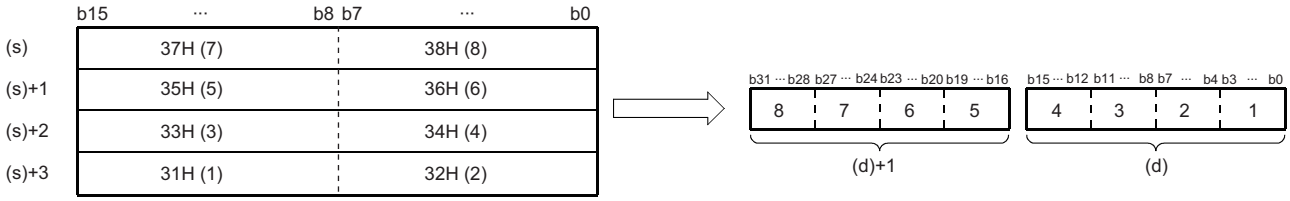


ASCII 10^7: Ten-millions place of ASCII code
ASCII 10^6: Millions place of ASCII code
ASCII 10^5: Hundred-thousands place of ASCII code
ASCII 10^4: Ten-thousands place of ASCII code
ASCII 10^3: Thousands place of ASCII code
ASCII 10^2: Hundreds place of ASCII code
ASCII 10^1: Tens place of ASCII code
ASCII 10^0: Ones place of ASCII code

- The ASCII data in the device specified by (s) to (s)+3 is within the range from 0 to 99999999.
- The data of (s)+4 or later is ignored.
- A value from 30H to 39H can be set in each place of ASCII code.
- If a value 20H or 00H is set in each place of ASCII code, the value will be processed as 30H.

Ex.

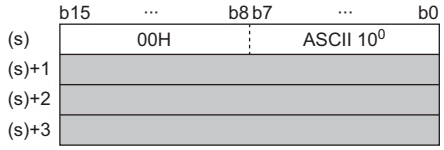
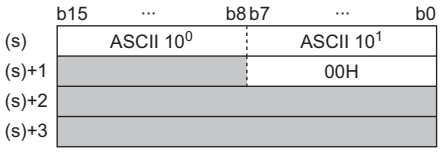
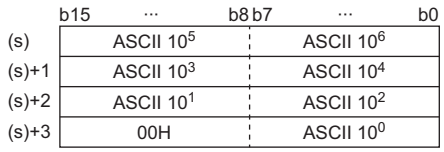
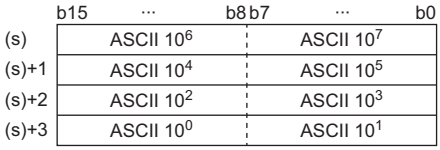
When 87654321 is specified in (s)



■Setting method of (s) for when SM705 (Number of conversion digits selection) is on

Set decimal ASCII data with a desired number of digits (including 00H (NULL code)) in (s). Note that 00H (NULL code) is not required to be set for the maximum number of digits (8 digits).

The following table lists the setting method of (s).

Value to be set in (s)	Data of (s) to (s)+3	Value to be set in (s)	Data of (s) to (s)+3
• 0 to 9	• Set 00H in the upper byte of (s). • The data of (s)+1 or later is ignored. 	• 10 to 99	• Set 00H in the lower byte of (s)+1. • The data of the upper byte of (s)+1 or later is ignored. 
⋮			
• 1000000 to 9999999	• Set 00H in the upper byte of (s)+3. • The data of (s)+4 or later is ignored. 	• 10000000 to 99999999	• The data of (s)+4 or later is ignored. 

ASCII 10^7: Ten-millions place of ASCII code
ASCII 10^6: Millions place of ASCII code
ASCII 10^5: Hundred-thousands place of ASCII code
ASCII 10^4: Ten-thousands place of ASCII code
ASCII 10^3: Thousands place of ASCII code
ASCII 10^2: Hundreds place of ASCII code
ASCII 10^1: Tens place of ASCII code
ASCII 10^0: Ones place of ASCII code

- The ASCII data in the device specified by (s) to (s)+3 is within the range from 0 to 99999999.
- The data of (s)+4 or later is ignored.
- A value from 30H to 39H can be set in each place of ASCII code.
- If a value 20H is set in each place of ASCII code, the value is processed as 30H. If a value 00H is set, the value is processed as the end of the decimal ASCII data.

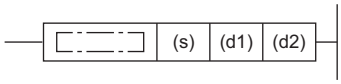
Operation error

Error code (SD0)	Description
3281H	Invalid data that cannot be converted are input in (s). • A character other than 0 to 9 exists in the data.

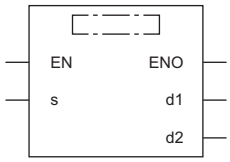
Converting decimal string data to 16-bit binary data

VAL(P)(_U)

These instructions convert strings to 16-bit binary data.

Ladder	ST	
	ENO:=VAL(EN,s,d1,d2); ENO:=VALP(EN,s,d1,d2);	ENO:=VAL_U(EN,s,d1,d2); ENO:=VALP_U(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
VAL VAL_U	
VALP VALP_U	

Setting data

■Description, range, data type

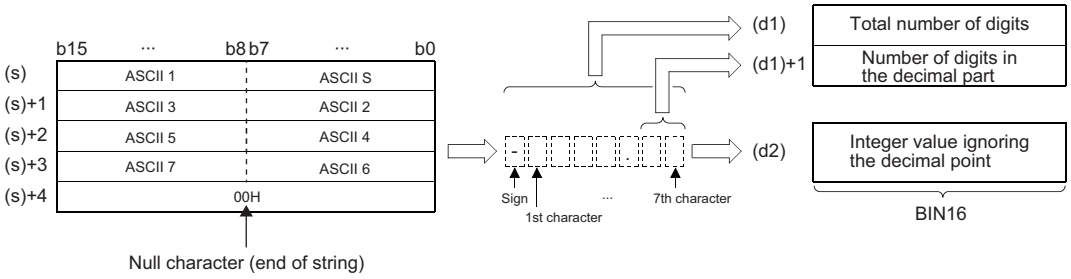
Operand		Description	Range	Data type	Data type (label)
(s)		String data to be converted to binary data or the start device where the string data is stored	—	String	ANYSTRING_SINGLE
(d1)	VAL(P)	Start device for storing the number of binary digits after conversion	—	16-bit signed binary	ANY16_S_ARRAY (Number of elements: 2)
	VAL(P)_U			16-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 2)
(d2)	VAL(P)	Device for storing the binary data after conversion	—	16-bit signed binary	ANY16_S
	VAL(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

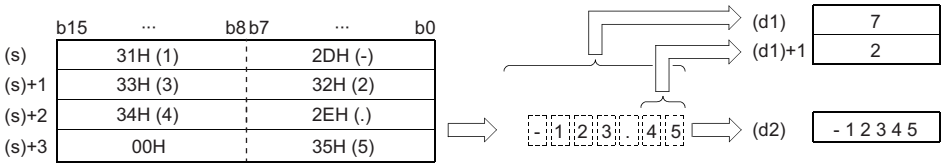
- These instructions convert the strings stored in the device numbers specified by (s) and later to 16-bit binary data, and store the number of digits in (d1) and converted binary data in the device specified by (d2). For conversion of strings to binary data, the data from the device number specified by (s) to the device number containing "00H" is processed as strings.



ASCII S: ASCII code (sign data)
ASCII □: ASCII code (□th character)

Ex.

When a string "-123.45" (signed) is stored in the device specified by (s) or later

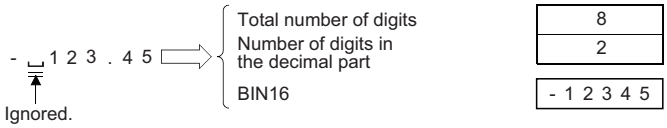


- The total number of characters of the string stored in the device specified by (s) is 2 to 8.
- Of the string stored in the device specified by (s), the number of characters in the decimal part is 0 to 5. Note, however, that the number must not exceed the total number of digits minus 3.
- A numerical string that can be converted to binary data ranges from -32768 to 32767 when a signed value is specified ignoring the decimal point or from 0 to 65535 when an unsigned value is specified. Numerical strings excluding signs and decimal points can be specified only within the range from 30H to 39H. (A value ignoring the decimal point···"-12345.6" for example becomes "-123456".)
- For the sign, "20H" can be set to indicate a positive numerical value, or "2DH" can be set to indicate a negative numerical value.
- "2EH" is set for the decimal point.
- The total number of digits stored in the device specified by (d1) includes all characters (including signs and decimal points) that represent a numerical value. The number of digits in the decimal part to be stored in the device specified by (d1)+1 represents the decimal part after 2EH(.) For the 16-bit binary data to be stored in the device specified by (d2), the string is converted to binary data by ignoring the decimal point and stored.

- If "20H" (space) or "30H" (0) exists between the sign and the first numerical value other than 0 in the string in the device specified by (s), the instruction performs conversion to binary data by ignoring "20H" and "30H".

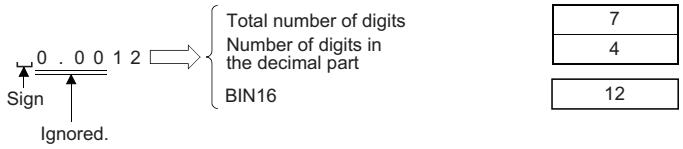
Ex.

When "20H" exists between the sign and the first numerical value other than 0 (when a signed value is specified)



Ex.

When "30H" exists between the sign and the first numerical value other than 0



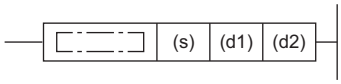
Operation error

Error code (SD0)	Description
2820H	00H is not set between the device number specified by (s) and the last device number of the relevant device.
3281H	Invalid data that cannot be converted are input in (s). • The number of characters is not between 2 and 8. • The number of characters in the decimal part is not between 0 and 5. • The relationship between the total number of characters and the number of characters in the decimal part is not in the following range. Total number of characters - 3≥number of characters in the decimal part • When the VAL(P) instruction is used, an ASCII code other than "20H" and "2DH" is set as a sign. • When the VAL(P)_U instruction is used, an ASCII code other than "20H" is set as a sign. • An ASCII code other than "30H" to "39H" and "2EH" (decimal point) is set as a digit of individual numbers. • More than one decimal point are set. The converted binary value exceeds the range in which each instruction can implement conversion.
3285H	The number of characters of the string in the device specified by (s) exceeds 16383.

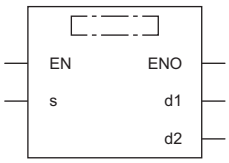
Converting decimal string data to 32-bit binary data

DVAL(P)(_U)

These instructions convert strings to 32-bit binary data.

Ladder	ST	
	ENO:=DVAL(EN,s,d1,d2); ENO:=DVALP(EN,s,d1,d2);	ENO:=DVAL_U(EN,s,d1,d2); ENO:=DVALP_U(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DVAL DVAL_U	
DVALP DVALP_U	

Setting data

■Description, range, data type

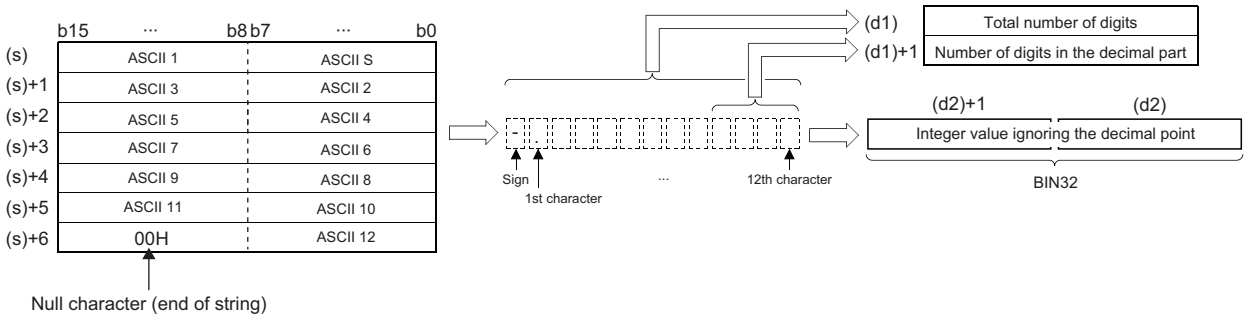
Operand		Description	Range	Data type	Data type (label)
(s)		String data to be converted to binary data or the start device where the string data is stored	—	String	ANYSTRING_SINGLE
(d1)	DVAL(P)	Start device for storing the number of binary digits after conversion	—	16-bit signed binary	ANY16_S_ARRAY (Number of elements: 2)
	DVAL(P)_U			16-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 2)
(d2)	DVAL(P)	Start device for storing the converted binary data	—	32-bit signed binary	ANY32_S
	DVAL(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

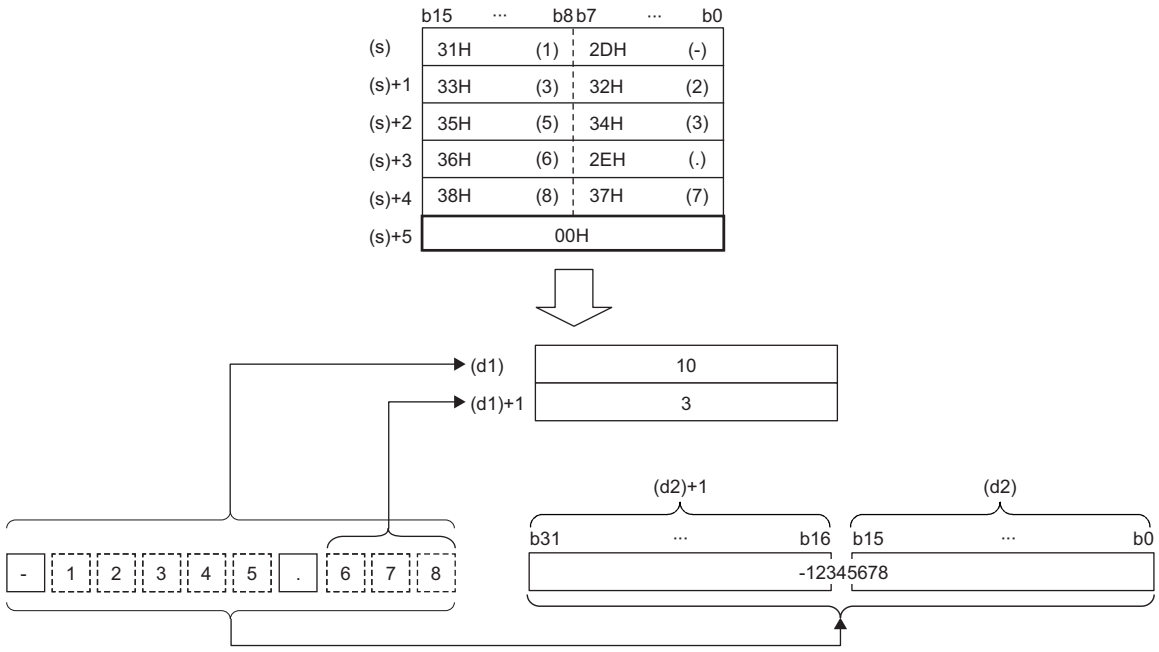
- These instructions convert the strings stored in the device numbers specified by (s) and later to 32-bit binary data, and store the number of digits in (d1) and converted binary data in the device specified by (d2). For conversion of strings to binary data, the data from the device number specified by (s) to the device number containing "00H" is processed as strings.



ASCII S: ASCII code (sign data)
ASCII □: ASCII code (□th character)

Ex.

When a string "-12345.678" (signed) is stored in the device specified by (s) or later

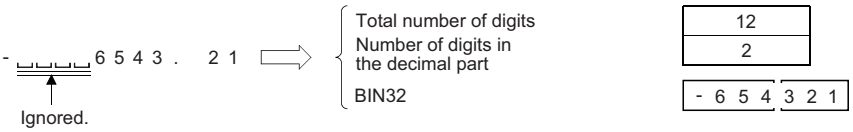


- The total number of characters of the string stored in the device specified by (s) is 2 to 13.
- Of the string stored in the device specified by (s), the number of characters in the decimal part is 0 to 10. Note, however, that the number must not exceed the total number of digits minus 3.
- The range of numerical strings that can be converted to binary is as follows. Numerical strings excluding signs and decimal points can be specified only within the range from 30H to 39H. (A value ignoring the decimal point···"-12345.6" for example becomes "-123456".)
  - When a signed value ignoring the decimal point is specified: -2147483648 to 2147483647
  - When an unsigned value ignoring the decimal point is specified: 0 to 4294967295
- For the sign, "20H" can be set to indicate a positive numerical value, or "2DH" can be set to indicate a negative numerical value.
- "2EH" is set for the decimal point.
- The total number of digits stored in the device specified by (d1) includes all characters (including signs and decimal points) that represent a numerical value. The number of digits in the decimal part to be stored in the device specified by (d1)+1 represents the decimal part after 2EH(.) For the 32-bit binary data to be stored in the device specified by (d2), the string is converted to binary data by ignoring the decimal point and stored.

- If "20H" (space) or "30H" (0) exists between the sign and the first numerical value other than 0 in the string in the device specified by (s), the instruction performs conversion to binary data by ignoring "20H" and "30H".

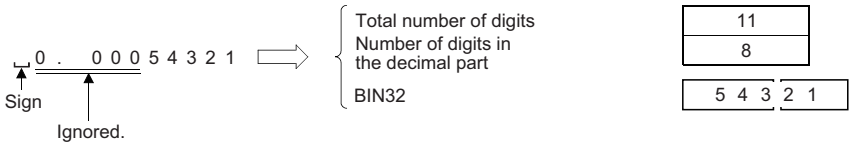
Ex.

When "20H" exists between the sign and the first numerical value other than 0 (when a signed value is specified)



Ex.

When "30H" exists between the sign and the first numerical value other than 0



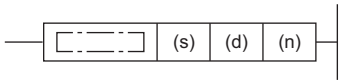
Operation error

Error code (SD0)	Description
2820H	00H is not set between the device number specified by (s) and the last device number of the relevant device.
3281H	Invalid data that cannot be converted are input in (s). • The number of characters of the string is not between 2 and 13. • The number of characters in the decimal part of the string is not between 0 and 10. • The relationship between the total number of characters and the number of characters in the decimal part is not in the following range. Total number of characters - 3≥number of characters in the decimal part • When the DVAL(P) instruction is used, an ASCII code other than "20H" and "2DH" is set as a sign. • When the DVAL(P)_U instruction is used, an ASCII code other than "20H" is set as a sign. • An ASCII code other than "30H" to "39H" and "2EH" (decimal point) is set as a digit of individual numbers. • More than one decimal point are set.
	The converted binary value exceeds the range in which each instruction can implement conversion.
3285H	The number of characters of the string in the device specified by (s) exceeds 16383.

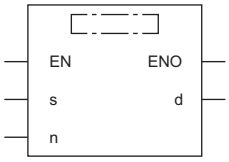
Converting hexadecimal ASCII to hexadecimal binary data

ASC2INT(P)

These instructions convert hexadecimal ASCII data to binary data.

Ladder	ST
	ENO:=ASC2INT(EN,s,n,d); ENO:=ASC2INTP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ASC2INT	
ASC2INTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the string data to be converted to binary data	—	String	ANYSTRING_SINGLE*1
(d)	Start device for storing the converted binary data	—	16-bit signed binary	ANY16*1
(n)	Number of characters to be stored	0 to 16383	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

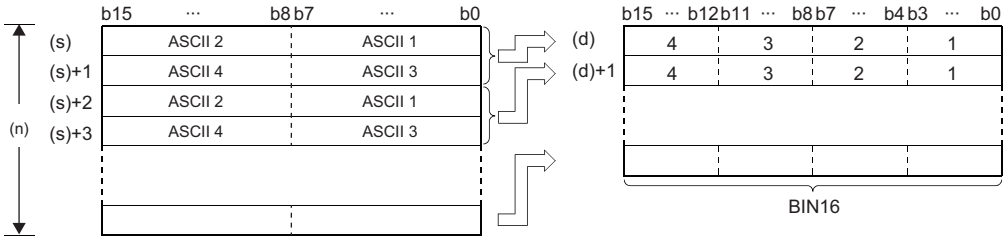
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions convert the hexadecimal ASCII data stored in the device by the number of characters specified by (n) after the device number specified by (s) and later to binary data, and store the converted data in the device number specified by (d) and later.

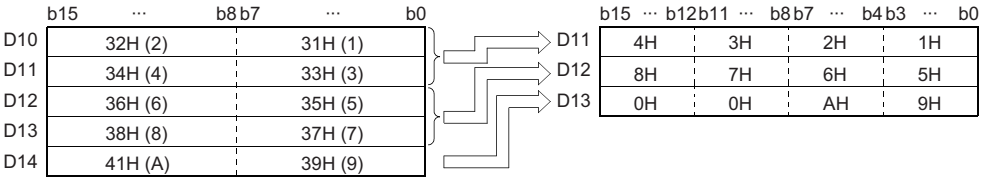


ASCII ☐: ASCII code (☐th digit)

1 to 4: 1st to 4th digit

(n): Number of characters specified by (n)

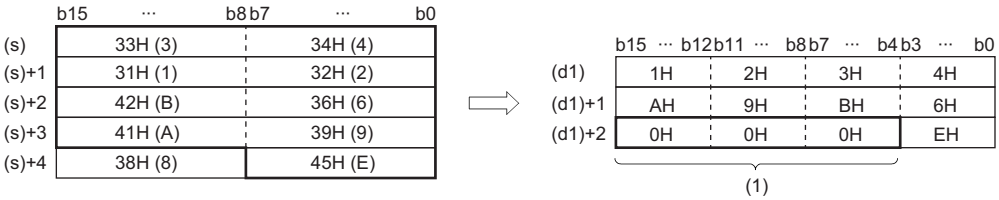
- Setting the number of characters for (n) automatically determines the range of the string in the device specified by (s) and the device range in which the binary data in the device specified by (d) is stored.
- Processing is performed normally even if the device range in which the ASCII data to be converted and the device range for storing the converted binary data overlap.



- If the number of characters in the device specified by (n) is not a multiple of 4, "0H" is automatically stored after the specified number of the last device number among device numbers for storing the converted binary data.

Ex.

When the number of characters in (n) is 9



(1) 0H is automatically stored.

- If the number of characters in the device specified by (n) is 0, no processing is performed.
- The ASCII code that can be specified by (s) must be in the range from "30H" to "39H" or "41H to 46H".

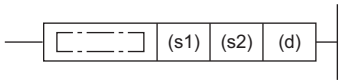
Operation error

Error code (SD0)	Description
3281H	A character other than hexadecimal numerical string (an ASCII code other than "30H" to "39H" and "41H" to "46H") is set in the device specified by (s).
3285H	Out-of-range data is set in the device specified by (n). • The specified number of characters is not between 0 and 16383.

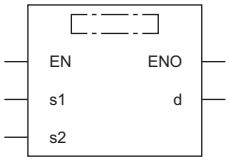
Converting single-precision real number to BCD format data

EMOD(P)

These instructions convert single-precision real number data to the BCD floating point format data.

Ladder	ST
	ENO:=EMOD(EN,s1,s2,d); ENO:=EMODP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EMOD	
EMODP	

Setting data

■Description, range, data type

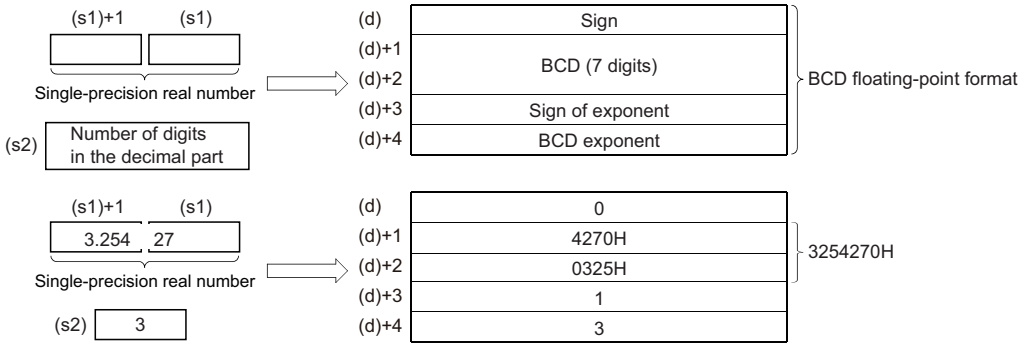
Operand	Description	Range	Data type	Data type (label)
(s1)	Single-precision real number data or the start device where the single-precision real number data is stored	0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Decimal part digit data	0 to 7	16-bit signed binary	ANY16
(d)	Start device for storing the BCD format data	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

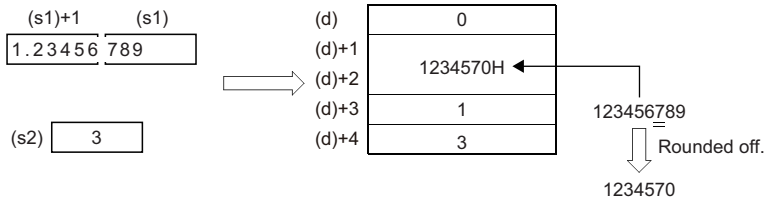
- These instructions convert the single-precision real number data stored in the device specified by (s1) to the BCD floating point format based on the number of decimal part digits stored in the device specified by (s2), and store the converted data in the device number specified by (d) and later.



- For the sign in (d) and the exponent sign in (d)+3, 0 is set for positive and 1 is set for negative.
- For the BCD exponent in (d)+4, a value between 0 and 38 is stored.
- The number of decimal part digits of the single-precision real number data in the device specified by (s1) is stored in the device specified by (s2). The example in the above figure shows the following.



- Six-digit BCD data, determined by rounding off the seventh digit, is stored in (d)+1 and (d)+2.



- A value of 0 to 7 can be set for the number of decimal part digits in the device specified by (s2).
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

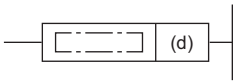
Operation error

Error code (SD0)	Description
3281H	The number of decimal part digits in the device specified by (s2) is out of the range from 0 to 7.
3282H	The value set to a device or label in (s1) is -0, a subnormal number, NaN (not a number), or ±∞.

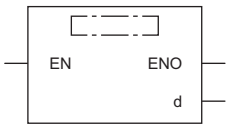
Two's complement of 16-bit binary data (sign inversion)

NEG(P)

Invert the sign of 16-bit binary device.

Ladder	ST
	ENO:=NEG(EN,d); ENO:=NEGP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
NEG	
NEGP	

Setting data

■Description, range, data type

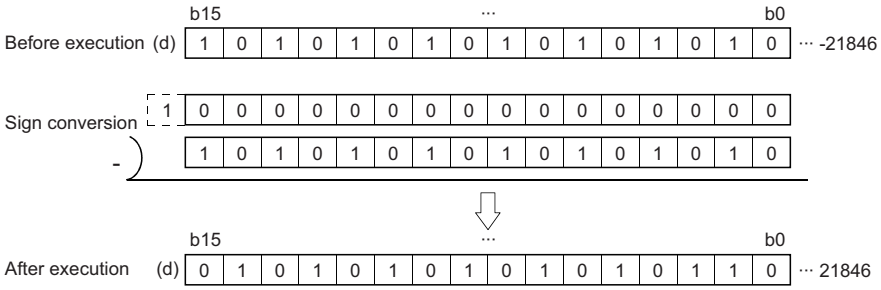
Operand	Description	Range	Data type	Data type (label)
(d)	Device where the data subjected to two's complement is stored	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions invert the sign of the 16-bit binary data in the device specified by (d) and store the inverted data in the device specified by (d).
- The instructions are used to invert positive and negative signs.



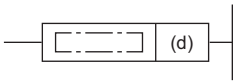
Operation error

There is no operation error.

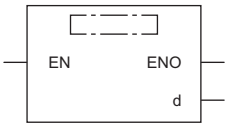
Two's complement of 32-bit binary data (sign inversion)

DNEG(P)

These instructions invert the sign of 32-bit binary device.

Ladder	ST
	ENO:=DNEG(EN,d); ENO:=DNEGP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DNEG	
DNEGP	

Setting data

■Description, range, data type

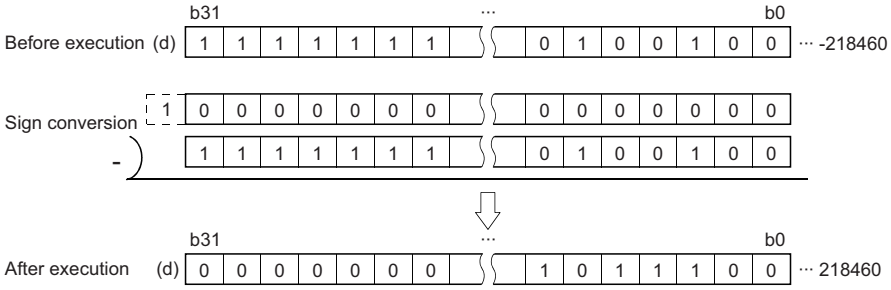
Operand	Description	Range	Data type	Data type (label)
(d)	Start device where the data subjected to two's complement is stored	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions invert the sign of the 32-bit binary data in the device specified by (d) and store the inverted data in the device specified by (d).
- The instructions are used to invert positive and negative signs.



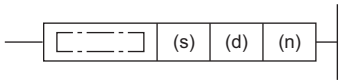
Operation error

There is no operation error.

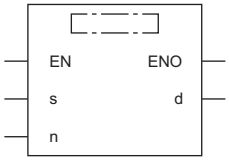
Decoding 8-bit data to 256-bit data

DECO(P)

These instructions decode the lower (n) bits of the specified device.

Ladder	ST
	ENO:=DECO(EN,s,n,d); ENO:=DECOP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DECO	
DECOP	

Setting data

■Description, range, data type

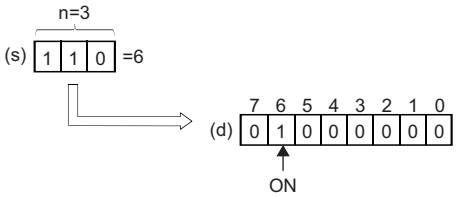
Operand	Description	Range	Data type	Data type (label)
(s)	Decode data or the start number of the device where the decode data is stored	—	Bit/16-bit signed binary	ANY_ELEMENTARY
(d)	Device for storing the decoded data	—	Bit/Word	ANY_ELEMENTARY
(n)	Effective bit length	1 to 8	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions turn on the bit, corresponding to the binary value specified by the lower (n) bits in the device specified by (s), in the device specified by (d)



- Specify a value 1 to 8 for (n).
- When (n)=0, no processing is performed and the values in the device specified by (d) remain unchanged.
- A bit device is treated as 1 bit, and a word device is treated as 16 bits.

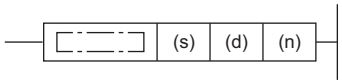
Operation error

Error code (SD0)	Description
3281H	The value specified by (n) is out of the range, 0 to 8.

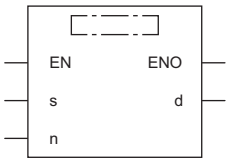
Encoding 256-bit data to 8-bit data

ENCO(P)

These instructions encode the bit data of 'n'th power of 2.

Ladder	ST
	ENO:=ENCO(EN,s,n,d); ENO:=ENCOP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ENCO	
ENCOP	

Setting data

■Description, range, data type

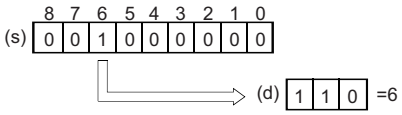
Operand	Description	Range	Data type	Data type (label)
(s)	Device where the encode data is stored	—	Bit/Word	ANY_ELEMENTARY
(d)	Start number of the device for storing the encoded data	—	Bit/16-bit signed binary	ANY_ELEMENTARY
(n)	Effective bit length	1 to 8	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions store the binary value, corresponding to the bit which is set to 1 in the $2^{(n)}$ bit data in the device specified by (s), in the device specified by (d).



- Specify a value 1 to 8 for (n).
- When (n)=0, no processing is performed and the values in the device specified by (d) remain unchanged.
- A bit device is treated as 1 bit, and a word device is treated as 16 bits.
- When two or more bits are 1, the upper bit position is used for processing.

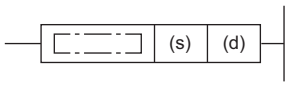
Operation error

Error code (SD0)	Description
3281H	The value specified by (n) is out of the range, 0 to 8.
	The bits in the $2^{(n)}$ bit data in the device specified by (s) are all 0s.

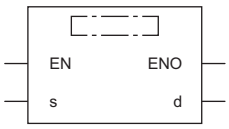
Decoding data to seven-segment display data

SEG(P)

These instructions decode the data consisting of 0 to F specified by the lower 4 bits of the device to seven-segment display data.

Ladder	ST
	ENO:=SEG(EN,s,d); ENO:=SEGP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SEG	
SEGP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Decode data or the device where the decode data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the decoded data	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions decode the data consisting of 0 to F specified by the lower 4 bits in the device specified by (s) to seven-segment display data, and store the decoded data in the device specified by (d).
- In the case of a bit device, (d) indicates the start device for storing 7-segment display data. In the case of a word device, it indicates the device number for storing the data.

Ex.

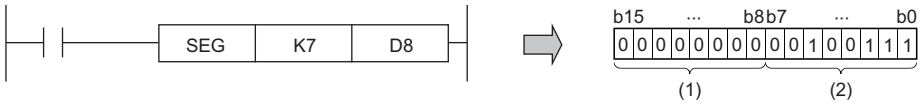
Bit device



The data in Y48 to Y4F does not change until the next data is output.

Ex.

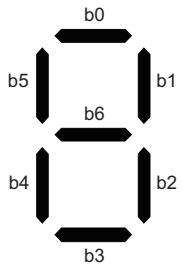
Word device



(1) The upper 8 bits are filled with 0s.

(2) The seven-segment display data are stored in the lower 8 bits.

• The following is the truth table for the seven-segment display.

(s)		Seven-segment display	(d)								Display data
Hexadecimal	Bit pattern		b7	b6	b5	b4	b3	b2	b1	b0	
0	0000		0	0	1	1	1	1	1	1	0
1	0001		0	0	0	0	0	1	1	0	1
2	0010		0	1	0	1	1	0	1	1	2
3	0011		0	1	0	0	1	1	1	1	3
4	0100		0	1	1	0	0	1	1	0	4
5	0101		0	1	1	0	1	1	0	1	5
6	0110		0	1	1	1	1	1	0	1	6
7	0111		0	0	1	0	0	1	1	1	7
8	1000		0	1	1	1	1	1	1	1	8
9	1001		0	1	1	0	1	1	1	1	9
A	1010		0	1	1	1	0	1	1	1	A
B	1011		0	1	1	1	1	1	0	0	b
C	1100		0	0	1	1	1	0	0	1	C
D	1101		0	1	0	1	1	1	1	0	d
E	1110		0	1	1	1	1	0	0	1	E
F	1111		0	1	1	1	0	0	0	1	F

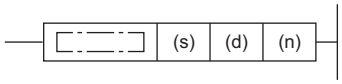
Operation error

There is no operation error.

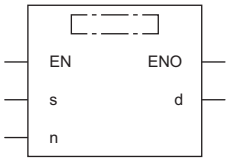
Separating data in units of 4 bits

DIS(P)

These instructions store the lower (n) nibble(s) of 16-bit binary data in another device range specified.

Ladder	ST
	ENO:=DIS(EN,s,n,d); ENO:=DISP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DIS	
DISP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device where the separation target data is stored	—	16-bit signed binary	ANY16
(d)	Start device for storing the separated data	—	16-bit signed binary	ANY16*1
(n)	Number of separation units	1 to 4	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

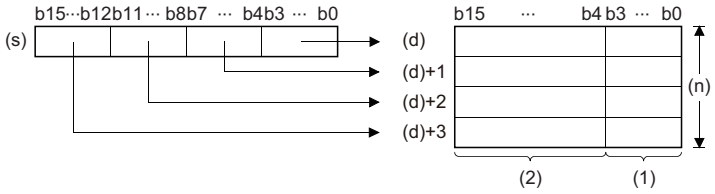
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions separate the lower (n) nibble(s) (4 bits/nibble) from the 16-bit binary data in the device specified by (s), and store each of the separated data in the lower 4 bits of the (n) points of data in the device specified by (d).



(1) Data storage area
(2) Filled with 0s.

- The upper 12 bits of the (n) points of data in the device specified by (d) are filled with 0s.
- Specify a value 1 to 4 for (n).
- When (n)=0, no processing is performed and the (n) points of data starting from the device specified by (d) remain unchanged.

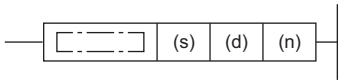
Operation error

Error code (SD0)	Description
3281H	The value specified by (n) is out of the range, 0 to 4.

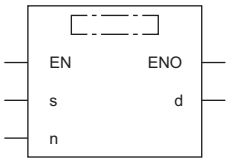
Combining data in units of 4 bits

UNI(P)

These instructions store the lower 4 bits of the (n) points of 16-bit binary data in another 16-bit device.

Ladder	ST
	ENO:=UNI(EN,s,n,d); ENO:=UNIP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UNI	
UNIP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the combination target data is stored	—	16-bit signed binary	ANY16*1
(d)	Device for storing the combined data	—	16-bit signed binary	ANY16
(n)	Number of combination units	1 to 4	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

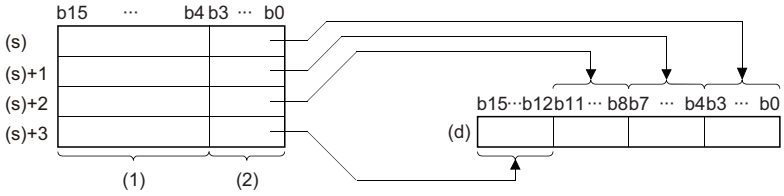
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions store the lower 4 bits of the (n) points of 16-bit binary data in the device specified by (s) in the 16-bit device specified by (d).



(1) Ignored.

(2) Data to be connected

- The upper bits (bits in the (4-n) nibble(s)) of data in the device specified by (d) are filled with 0s.
- Specify a value 1 to 4 for (n).
- When (n)=0, no processing is performed and the data in the device specified by (d) remain unchanged.

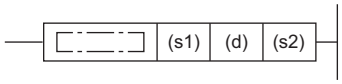
Operation error

Error code (SD0)	Description
3281H	The value specified by (n) is out of the range, 0 to 4.

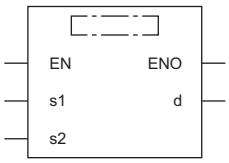
Separating data in units of bits

NDIS(P)

These instructions separate the data in units of bits. (The number of bits can be specified as desired.)

Ladder	ST
	ENO:=NDIS(EN,s1,s2,d); ENO:=NDISP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
NDIS	
NDISP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the separation target data is stored	—	16-bit signed binary	ANY16*1
(d)	Start device for storing the separated data	—	16-bit signed binary	ANY16*1
(s2)	Start device for storing the separation unit	—	16-bit signed binary	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

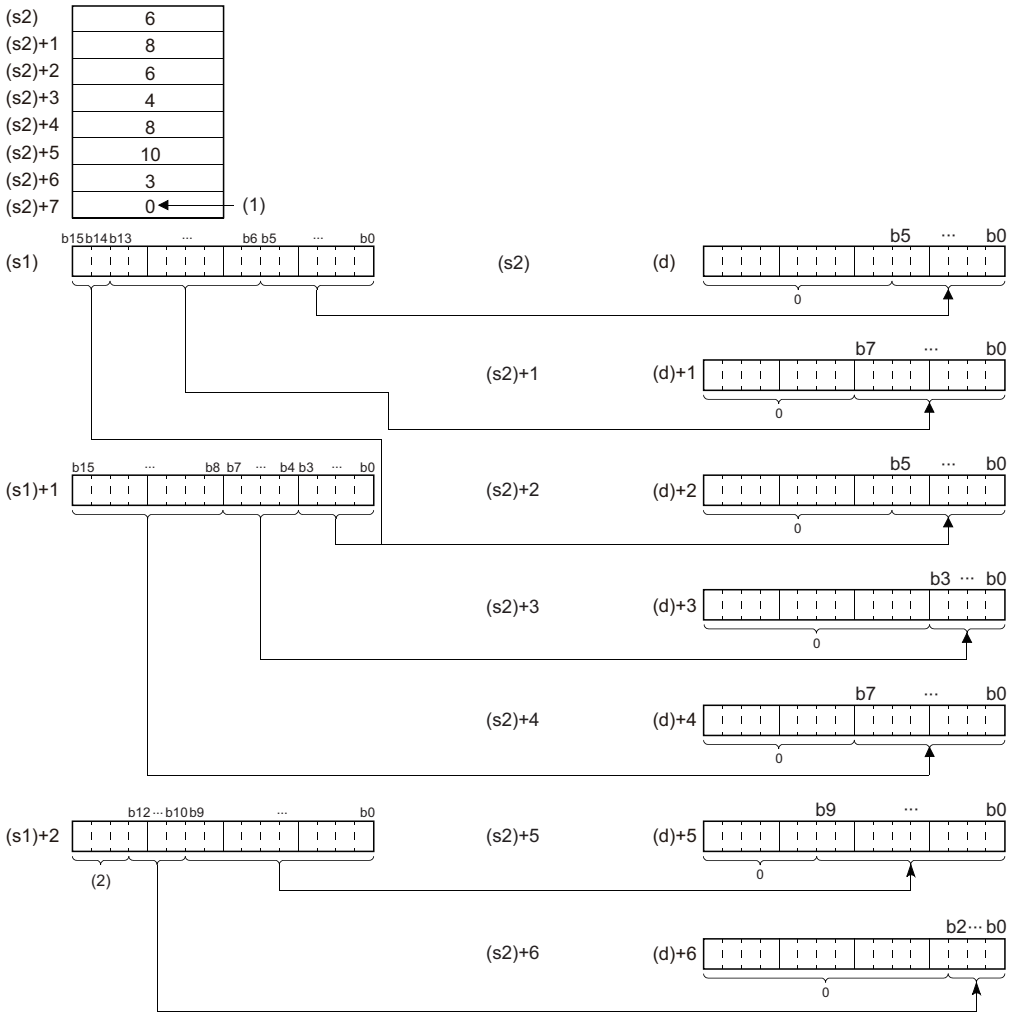
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions separate the bits of data in the device specified by (s1) and later in units of bits specified by (s2), and store the separated data in the device range specified by (d) and later.



(s2) to (s2)+6: Number of bits specified by (s2) to (s2)+6

(1) The value, 0, indicates the end of setting.

(2) Ignored.

- Specify the value 1 to 16 for (s2).
- The device areas from the one specified by (s2) to the one storing "0" are processed.
- Specify the devices so that the range of the device where the separation target data is stored ((s1) and later) and the range of the device for storing the separated data ((d) and later) do not overlap. If they overlap, a correct operation result may not be obtained.
- Do not overlap the device numbers that are specified by (s1), (s2), and (d).

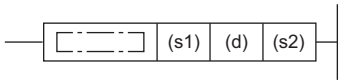
Operation error

Error code (SD0)	Description
2821H	The device ranges specified by (s1) and (s2) are overlapping.
	The device ranges specified by (s1) and (d) are overlapping.
	The device ranges specified by (s2) and (d) are overlapping.
3281H	Invalid data that cannot be converted are input in (s2). • The value specified is out of the range, 1 to 16. • There is no 0 set between the specified device and the last of the assignment range of the relevant device/label.

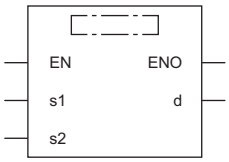
Combining data in units of bits

NUNI(P)

These instructions combine the data in units of bits. (The number of bits can be specified as desired.)

Ladder	ST
	ENO:=NUNI(EN,s1,s2,d); ENO:=NUNIP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
NUNI	
NUNIP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the combination target data is stored	—	16-bit signed binary	ANY16*1
(d)	Start device for storing the combined data	—	16-bit signed binary	ANY16*1
(s2)	Start device for storing the combination unit	—	16-bit signed binary	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

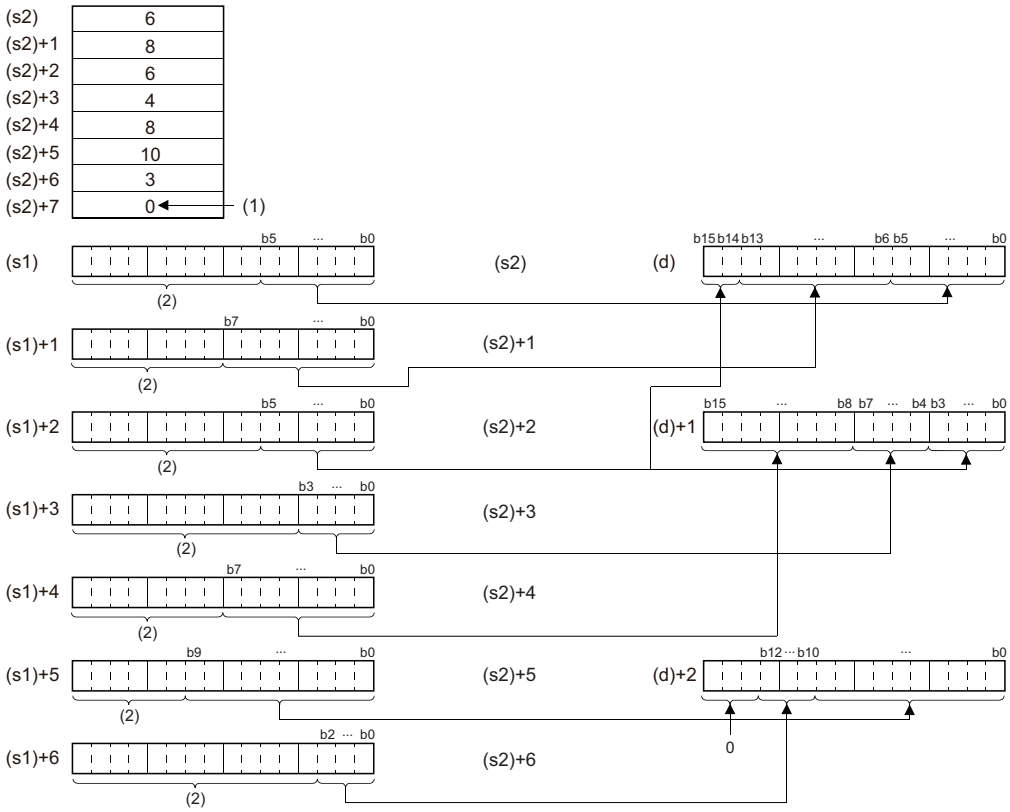
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions combine the bits of data in the device specified by (s1) and later in units of bits specified by (s2), and store the combined data in the device specified by (d) and later.



(s2) to (s2)+6: Number of bits specified by (s2) to (s2)+6

(1) The value, 0, indicates the end of setting.

(2) Ignored.

- Specify the value 1 to 16 for (s2).
- The device areas from the one specified by (s2) to the one storing "0" are processed.
- Specify the devices so that the range of the device where the combination target data is stored ((s1) and later) and the range of the device for storing the combined data ((d) and later) do not overlap. If they overlap, a correct operation result may not be obtained.
- Do not overlap the device numbers that are specified by (s1), (s2), and (d).

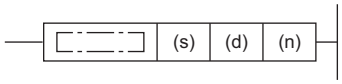
Operation error

Error code (SD0)	Description
2821H	The device ranges specified by (s1) and (s2) are overlapping.
	The device ranges specified by (s1) and (d) are overlapping.
	The device ranges specified by (s2) and (d) are overlapping.
3281H	Invalid data that cannot be converted are input in (s2). • The value specified is out of the range, 1 to 16. • There is no 0 set between the specified device and the last of the assignment range of the relevant device/label.

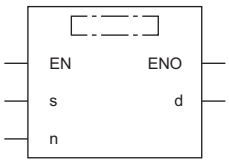
Separating data in units of bytes

WTOB(P)

These instructions separate 16-bit binary data into (n) bytes.

Ladder	ST
	ENO:=WTOB(EN,s,n,d); ENO:=WTOBP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
WTOB	
WTOBP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the separation target data is stored	—	16-bit signed binary	ANY16*1
(d)	Start device for storing the separated data	—	16-bit signed binary	ANY16*1
(n)	Number of data bytes to be separated	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

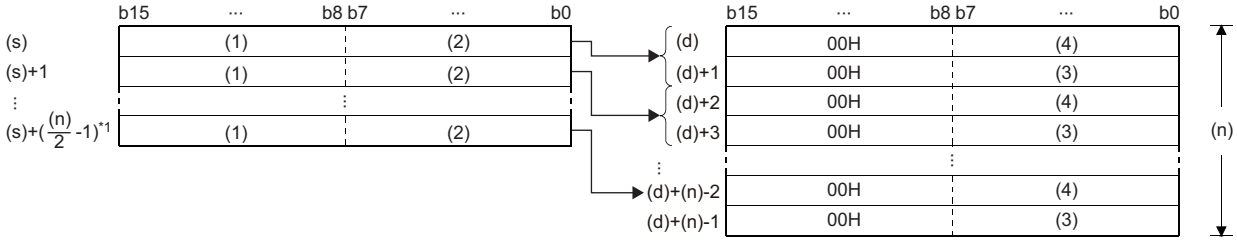
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions separate the 16-bit binary data in the device specified by (s) and later into (n) bytes, and store the separated data in the device specified by (d) and later.

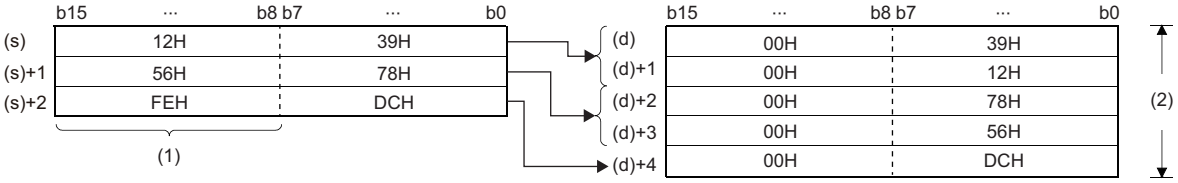


(1) Upper byte
(2) Lower byte
(3) Upper byte data
(4) Lower byte data

*1 Values after the decimal point are rounded up.

Ex.

When (n) is 5, the data in the device specified by (s) (upper 8 bits) to (s)+2 (lower 8 bits) are stored in the device specified by (d) to (d)+4.

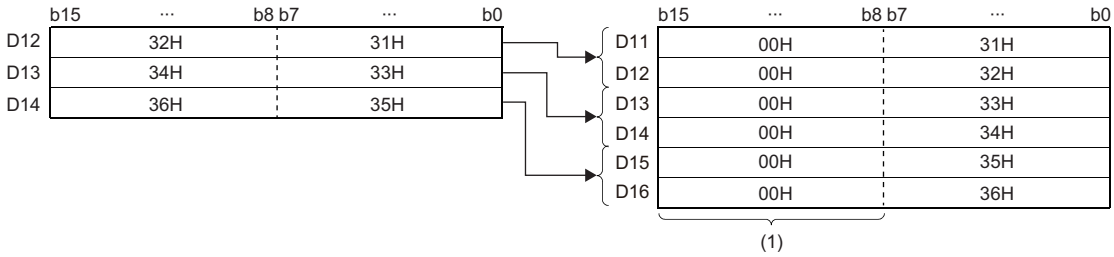


(1) The data, FEH, is ignored when (n) is 5.
(2) (n)=5

- Setting the number of bytes for (n) automatically determines the range of 16-bit binary data specified by (s) and the range of the device specified by (d) for storing the separated data.
- If (n) is 0, no processing is performed.
- The upper 8 bits of the device specified by (d) are automatically filled with 00Hs.

Ex.

When the byte data in D12 to D14 are stored in the lower 8 bits of D11 to D16



(1) 00H is automatically stored.

- Even when the ranges of the device where the separation target data is stored and the device for storing the separated data overlap, the processing is performed normally.

Range of the device where the separation target data is stored	Range of the device for storing the separated data
(s) to (s)+(n/2-1)	(d)+0 to (d)+(n)-1

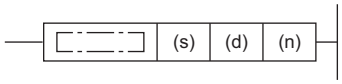
Operation error

There is no operation error.

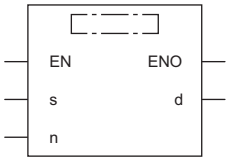
Combining data in units of bytes

BTOW(P)

These instructions combine the lower 8 bits of 16-bit binary data in units of words.

Ladder	ST
	ENO:=BTOW(EN,s,n,d); ENO:=BTOWP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BTOW	
BTOWP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the combination target data is stored	—	16-bit signed binary	ANY16*1
(d)	Start device for storing the combined data	—	16-bit signed binary	ANY16*1
(n)	Number of data bytes to be combined	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

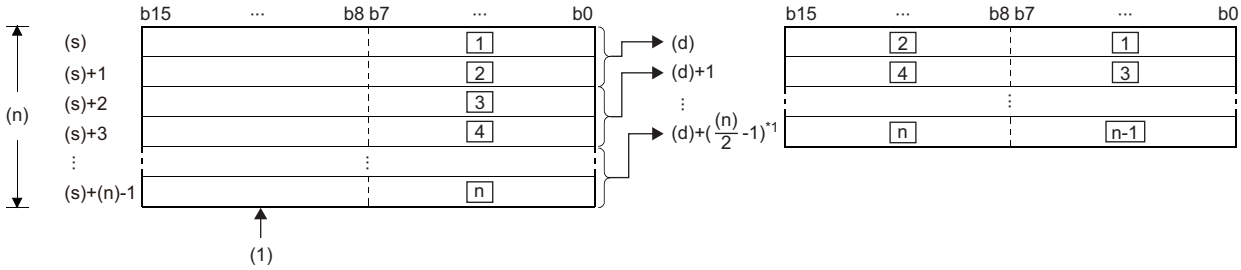
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions combine the (n) bytes of lower 8 bits of 16-bit binary data in the device specified by (s) and later in units of words, and store the combined data in the device specified by (d) and later.
- The (n) bytes of upper 8 bits of 16-bit binary data in the device specified by (s) and later are ignored. When (n) is an odd number, the upper 8 bits of the device where 'n'th-byte data is stored are filled with 0s.



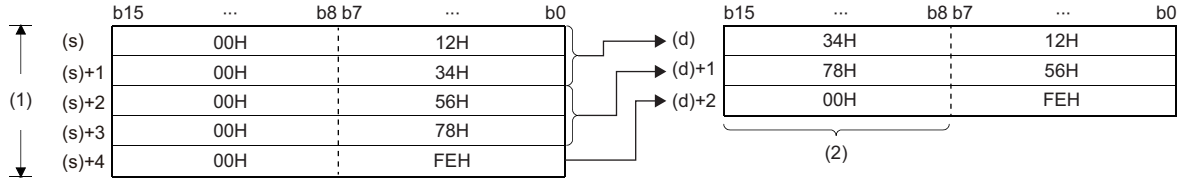
□: □th byte data

(1) The upper byte data are ignored.

*1 Values after the decimal point are rounded up.

Ex.

When (n) is 5, the lower 8 bits of the data in the device specified by (s) to (s)+4 are combined and stored in the device specified by (d) to (d)+2.



(1) (n)=5

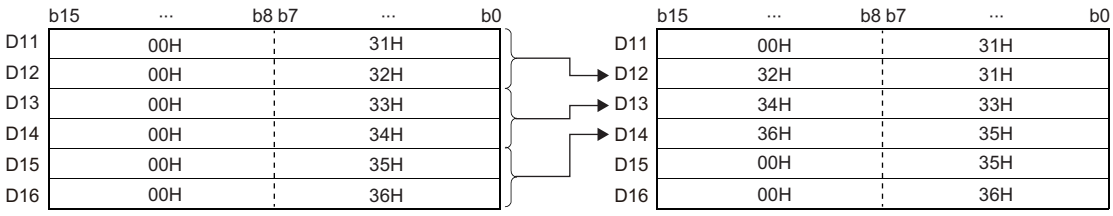
(2) Filled with 00H.

- Setting the number of bytes for (n) automatically determines the range of byte data in the device specified by (s) and the range of the device specified by (d) for storing the combined data.
- If (n) is 0, no processing is performed.
- The upper 8 bits in the device specified by (s) and later are ignored, and only the lower 8 bits are processed.
- Even when the ranges of the device where the combination target data is stored and the device for storing the combined data overlap, the processing is performed normally.

Range of the device where the combination target data is stored	Range of the device for storing the combined data
(s)+0 to (s)+(n)-1	(d) to (d)+(n/2-1)

Ex.

When the lower 8 bits of D11 to D16 are stored in D12 to D14



Operation error

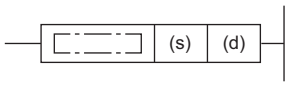
There is no operation error.

12.7 Data Transfer Instructions

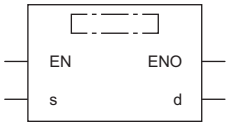
Transferring 16-bit binary data

MOV(P)

These instructions transfer the 16-bit binary data in the device specified.

Ladder	ST
	ENO:=MOV(EN,s,d); ENO:=MOVP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
MOV	
MOVP	

Setting data

■Description, range, data type

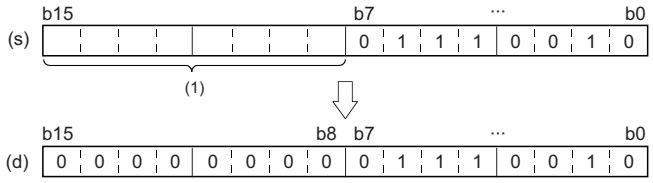
Operand	Description	Range	Data type	Data type (label)
(s)	Transfer source data or the number of the device where the transfer source data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Transfer destination device number	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions transfer the 16-bit binary data in the device specified by (s) to the device specified by (d).
- If (s) is a digit-specified bit device, the digit-specified bits are targeted. If data specified by (s) is less than 16 bits, 0s are added and transferred.



(1) If data specified by (s) is less than 16 bits, 0s are added and transferred.

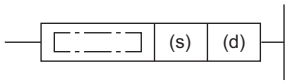
Operation error

There is no operation error.

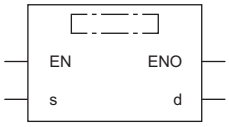
Transferring 32-bit binary data

DMOV(P)

These instructions transfer the 32-bit binary data in the device specified.

Ladder	ST
	ENO:=DMOV(EN,s,d); ENO:=DMOVP(EN,s,d)

FBD/LD



■Execution condition

Instruction	Execution condition
DMOV	
DMOVP	

Setting data

■Description, range, data type

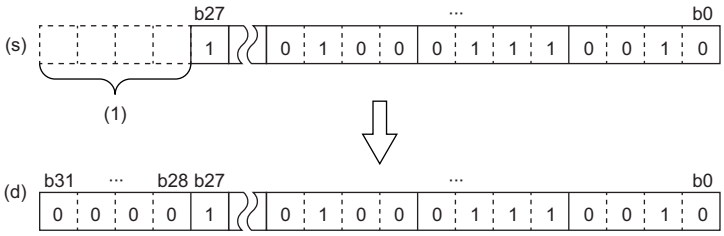
Operand	Description	Range	Data type	Data type (label)
(s)	Transfer source data or the number of the device where the transfer source data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Transfer destination device number	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions transfer the 32-bit binary data in the device specified by (s) to the device specified by (d).
- If (s) is a digit-specified bit device, the digit-specified bits are targeted. If data specified by (s) is less than 16 bits, 0s are added and transferred.



(1) If data specified by (s) is less than 32 bits, 0s are added and transferred.

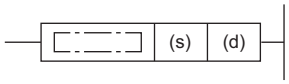
Operation error

There is no operation error.

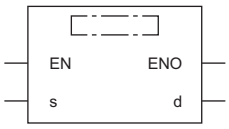
Inverting and transferring 16-bit binary data

CML(P)

These instructions invert the specified 16-bit binary data bit by bit, and transfer the inverted data.

Ladder	ST
	ENO:=CML(EN,s,d); ENO:=CMLP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
CML	
CMLP	

Setting data

■Description, range, data type

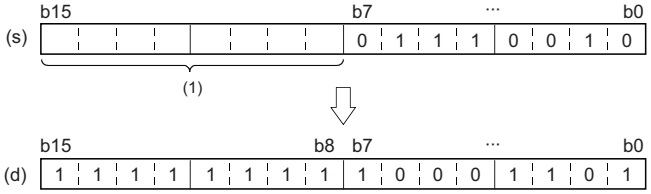
Operand	Description	Range	Data type	Data type (label)
(s)	Inversion target data or the number of the device where the inversion target data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Number of the device for storing the inverted data	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions invert the 16-bit binary data in the device specified by (s) bit by bit, and transfer the inverted data to the device specified by (d).
- If (s) is a digit-specified bit device, the digit-specified bits are targeted. If data specified by (s) is less than 16 bits, 0s are added and inverted.



(1): If data specified by (s) is less than 16 bits, 0s are added and inverted.

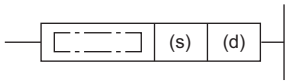
Operation error

There is no operation error.

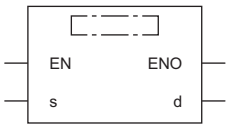
Inverting and transferring 32-bit binary data

DCML(P)

These instructions invert the specified 32-bit binary data bit by bit, and transfer the inverted data.

Ladder	ST
	ENO:=DCML(EN,s,d); ENO:=DCMLP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DCML	
DCMLP	

Setting data

■Description, range, data type

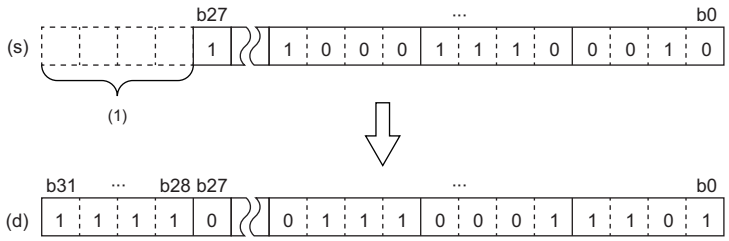
Operand	Description	Range	Data type	Data type (label)
(s)	Inversion target data or the number of the device where the inversion target data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Number of the device for storing the inverted data	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions invert the 32-bit binary data in the device specified by (s) bit by bit, and transfer the inverted data to the device specified by (d).
- If (s) is a digit-specified bit device, the digit-specified bits are targeted. If data specified by (s) is less than 16 bits, 0s are added and inverted.



(1) If data specified by (s) is less than 32 bits, 0s are added and inverted.

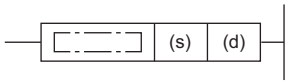
Operation error

There is no operation error.

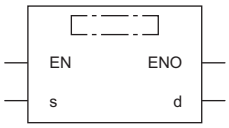
Inverting and transferring 1-bit data

CMLB(P)

These instructions invert the specified bit data, and transfer the inverted data.

Ladder	ST
	ENO:=CMLB(EN,s,d); ENO:=CMLBP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
CMLB	
CMLBP	

Setting data

■Description, range, data type

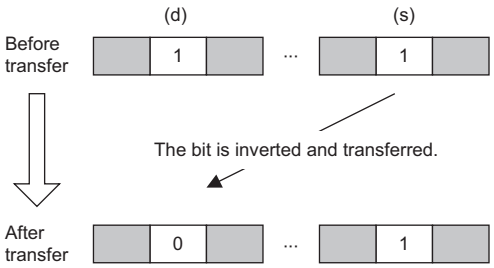
Operand	Description	Range	Data type	Data type (label)
(s)	Inversion target data or the number of the device where the inversion target data is stored	—	Bit	ANY_BOOL
(d)	Transfer destination device number	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	—	—	○	—	○	—	—	—	—
(d)	○	○	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions invert the bit data in the device specified by (s), and transfer the inverted data to the device specified by (d).



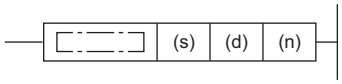
Operation error

There is no operation error.

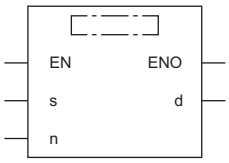
Transferring 16-bit binary data block (16 bits)

BMOV(P)

These instructions batch-transfer the (n) points (0 to 65535) of 16-bit binary data starting from the device specified.

Ladder	ST
	ENO:=BMOV(EN,s,n,d); ENO:=BMOVP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BMOV	
BMOVP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the transfer target data is stored	—	16-bit signed binary	ANY16*1*2
(d)	Transfer destination start device	—	16-bit signed binary	ANY16*1*2
(n)	Number of transfer data points	0 to 65535	16-bit unsigned binary	ANY16*2
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

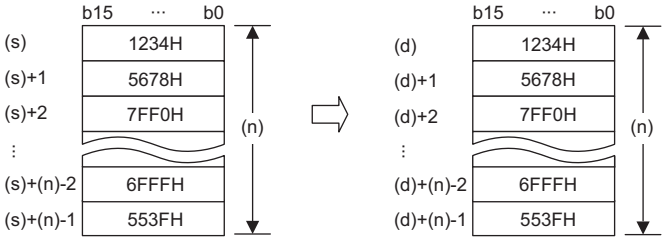
*2 Data types other than INT and WORD are also available. (Except for BOOL, POINTER, and STRUCT data types (structures)). Note that available data types are INT and WORD (in range of ANY16) when "Yes" is selected for "Check the data type of instruction argument" of option settings.
For the option settings, refer to the following.
GX Works3 Operating Manual

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	—	○	—	○	—	—	—	—
(d)	○	○	○	○	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

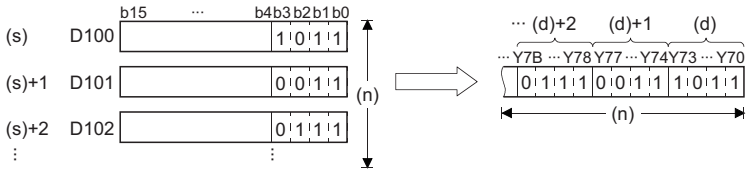
- These instructions batch-transfer the (n) points of 16-bit binary data starting from the device specified by (s) to the device specified by (d).



- Data can be transferred even when the transfer source device and destination device overlap. A transfer to smaller device numbers begins from the device specified by (s), and a transfer to larger device numbers begins from the device specified by (s)+(n)-1.
- When (s) is a word device and (d) is a bit device, the number of digit-specified bits in the word device is transferred.

Ex.

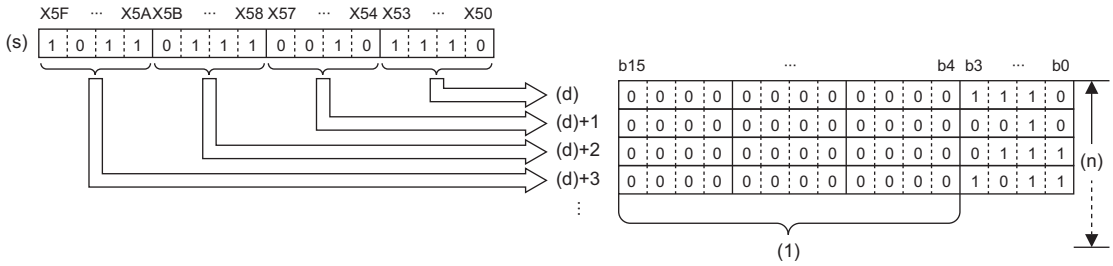
When K1Y70 is specified in (d), the lower 4 bits of the word device specified by (s) are transferred.



- When (s) is a digit-specified bit device and (d) is a word device, the number of digit-specified bits in the word device is transferred.

Ex.

When K1X50 is specified in (s), the data is transferred to the lower 4 bits of the word device specified by (d).



(1) Filled with 0s.

- When both (s) and (d) are bit devices, set the same number of digits for both devices.
- The link direct device, module access device, and CPU buffer memory access device cannot be specified by both (s) and (d) in one instruction. Specify any one of those for either (s) or (d).

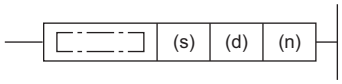
Operation error

Error code (SD0)	Description
3290H	The link direct device, module access device, or CPU buffer memory access device is specified by both (s) and (d).

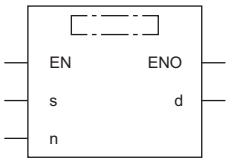
Transferring 16-bit binary data block (32 bits)

BMOVL(P)

These instructions batch-transfer the (n) points (1 to 4294967295) of 16-bit binary data starting from the device specified.

Ladder	ST
	ENO:=BMOVL(EN,s,n,d); ENO:=BMOVLP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BMOVL	
BMOVLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the transfer target data is stored	—	16-bit signed binary	ANY16*1
(d)	Transfer destination start device	—	16-bit signed binary	ANY16*1
(n)	Number of transfer data points	0 to 4294967295	32-bit unsigned binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

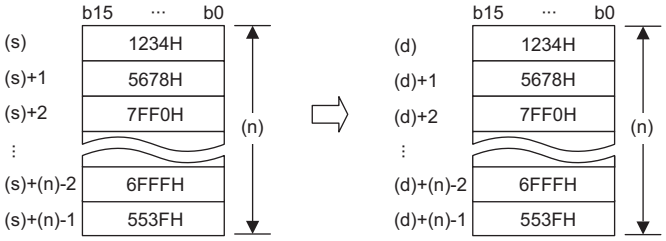
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	—	○	—	○	—	—	—	—
(d)	○	○	○	○	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	○	○	○	○	—	—	—

Processing details

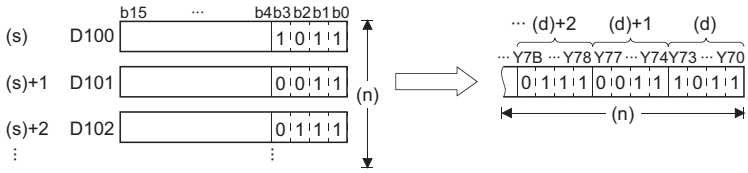
- These instructions batch-transfer the (n) points of 16-bit binary data starting from the device specified by (s) to the device specified by (d).



- Data can be transferred even when the transfer source device and destination device overlap. A transfer to smaller device numbers begins from the device specified by (s), and a transfer to larger device numbers begins from the device specified by (s)+(n)-1.
- When (s) is a word device and (d) is a bit device, the number of digit-specified bits in the word device is transferred.

Ex.

When K1Y70 is specified in (d), the lower 4 bits of the word device specified by (s) are transferred.



- When both (s) and (d) are bit devices, set the same number of digits for both devices.
- To use the link direct device, module access device, or CPU buffer memory access device for (s) or (d), specify it only for one of the devices.

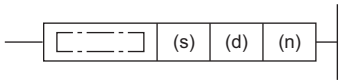
Operation error

Error code (SD0)	Description
3290H	The link direct device, module access device, or CPU buffer memory access device is specified by both (s) and (d).

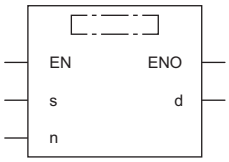
Transferring the same 16-bit binary data block (16 bits)

FMOV(P)

These instructions transfer 16-bit binary data to the (n) points (0 to 65535) starting from the device specified.

Ladder	ST
	ENO:=FMOV(EN,s,n,d); ENO:=FMOVP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FMOV	
FMOVP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Transfer target data or the device where the transfer target data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Transfer destination start device	—	16-bit signed binary	ANY16*1
(n)	Number of transfer data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

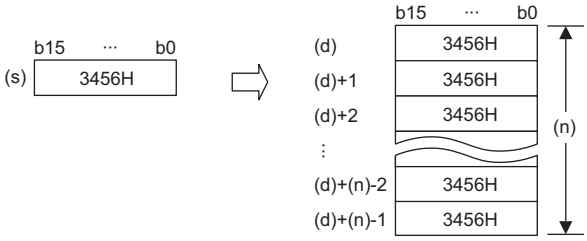
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

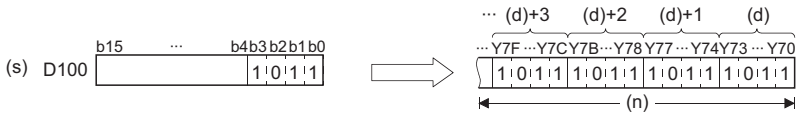
- These instructions transfer the 16-bit binary data in the device specified by (s) to the (n) points of the device specified by (d).



- When (s) is a word device and (d) is a bit device, the number of digit-specified bits in the word device is transferred.

Ex.

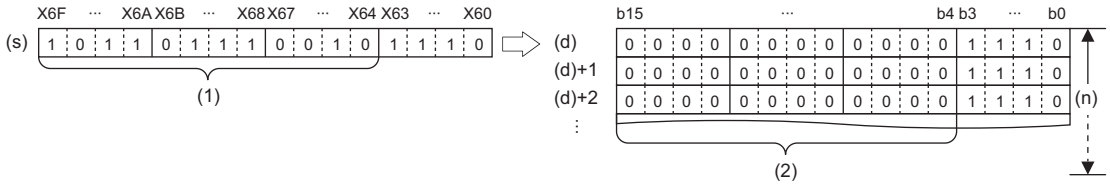
When K1Y70 is specified in (d), the lower 4 bits of the word device specified by (s) are transferred.



- When (s) is a digit-specified bit device and (d) is a word device, the number of digit-specified bits in the word device is transferred.

Ex.

When K1X60 is specified in (s), the data is transferred to the lower 4 bits of the word device specified by (d).



(1) Ignored.
(2) Filled with 0s.

- When both (s) and (d) are bit devices, set the same number of digits for both devices.

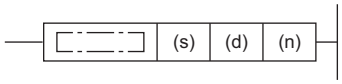
Operation error

There is no operation error.

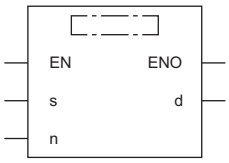
Transferring the same 16-bit binary data block (32 bits)

FMOVL(P)

These instructions transfer 16-bit binary data to the (n) points (1 to 4294967295) starting from the device specified.

Ladder	ST
	ENO:=FMOVL(EN,s,n,d); ENO:=FMOVLP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FMOVL	
FMOVLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Transfer target data or the device where the transfer target data is stored	-32768 to 32767	16-bit signed binary	ANY16
(d)	Transfer destination start device	—	16-bit signed binary	ANY16*1
(n)	Number of transfer data points	0 to 4294967295	32-bit unsigned binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

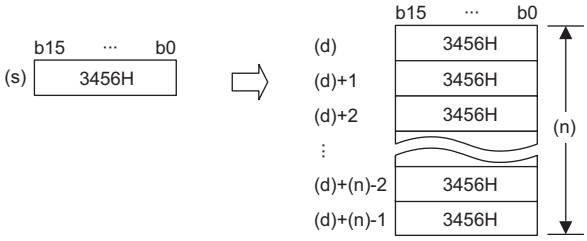
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	○	○	○	○	—	—	—

Processing details

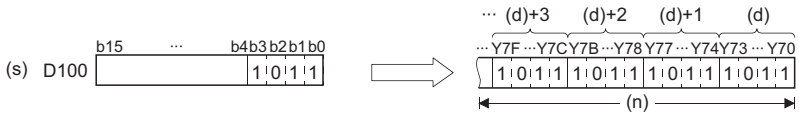
- These instructions transfer the 16-bit binary data in the device specified by (s) to the (n) points of the device specified by (d).



- When (s) is a word device and (d) is a bit device, the number of digit-specified bits in the word device is transferred.

Ex.

When K1Y70 is specified in (d), the lower 4 bits of the word device specified by (s) are transferred.



- When both (s) and (d) are bit devices, set the same number of digits for both devices.

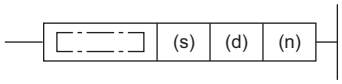
Operation error

There is no operation error.

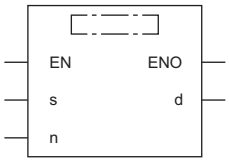
Transferring the same 32-bit binary data block (16 bits)

DFMOV(P)

These instructions transfer 32-bit binary data to the (n) points (1 to 65535) starting from the device specified.

Ladder	ST
	ENO:=DFMOV(EN,s,n,d); ENO:=DFMOVP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DFMOV	
DFMOVP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Transfer target data or the start device where the transfer target data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Transfer destination start device	—	32-bit signed binary	ANY32*1
(n)	Number of transfer data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

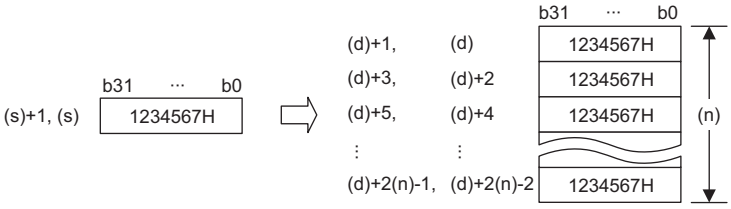
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

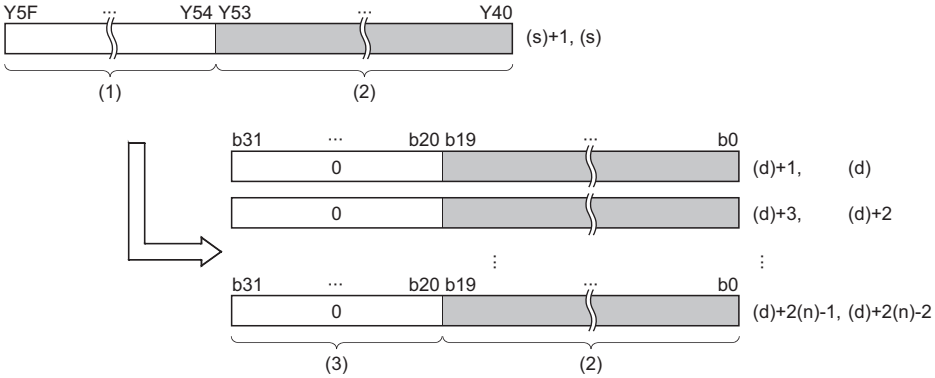
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions transfer the 32-bit binary data in the device specified by (s) to the (n) points of the device specified by (d).

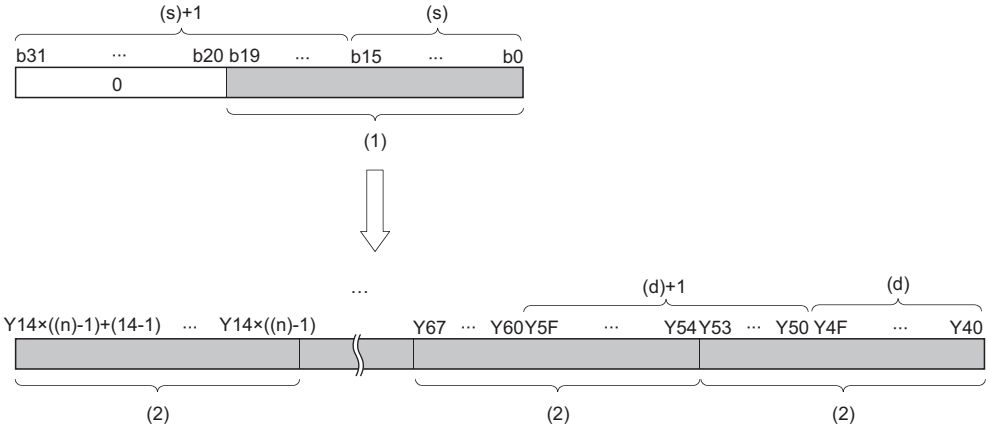


- When the number of digits is specified for the data in the device specified by (s), only the data corresponding to the specified digits are transferred. When K5Y40 is specified in (s), the lower 20 bits (five digits) of the word device specified by (s) are transferred.



(1) Ignored.
(2) Data of 20 bits (5 digits)
(3) Filled with 0s.

- When the number of digits is specified for the data in the device specified by (d), the data corresponding to the specified digits are transferred. When K5Y40 is specified in (d), the lower 20 bits of the device specified by (s) are transferred. When the number of digits is specified for both data in the devices specified by (s) and (d), the data corresponding to the digits specified in (d) are transferred.



(1) Data (20 bits (5 digits)) specified by (d)
(2) Data of 20 bits (5 digits)

- If (n) is 0, no processing is performed.

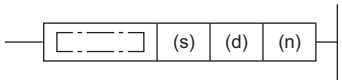
Operation error

There is no operation error.

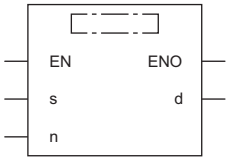
Transferring the same 32-bit binary data block (32 bits)

DFMOVL(P)

These instructions transfer 32-bit binary data to the (n) points (1 to 4294967295) starting from the device specified.

Ladder	ST
	ENO:=DFMOVL(EN,s,n,d); ENO:=DFMOVLP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DFMOVL	
DFMOVLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Transfer target data or the start device where the transfer target data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Transfer destination start device	—	32-bit signed binary	ANY32*1
(n)	Number of transfer data points	0 to 4294967295	32-bit unsigned binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

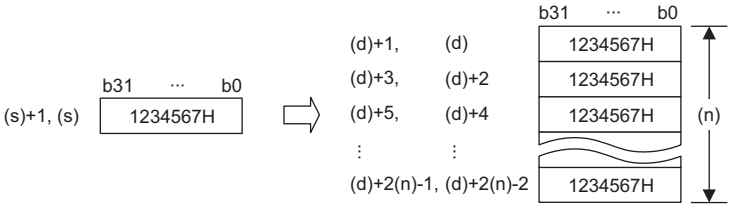
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

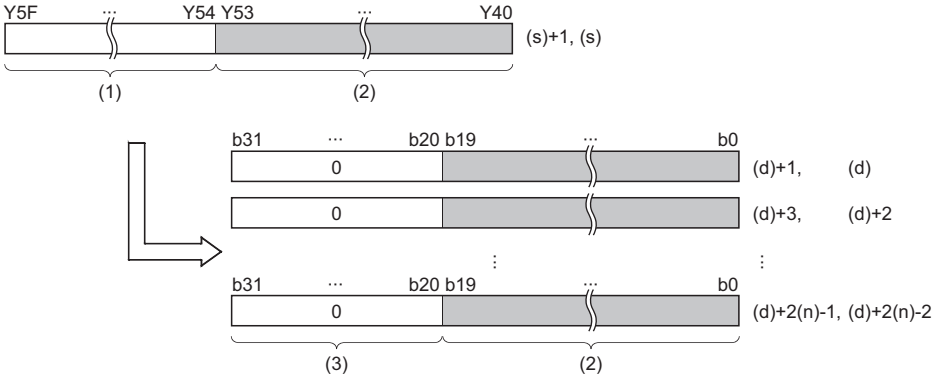
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	○	○	○	○	—	—	—

Processing details

- These instructions transfer the 32-bit binary data in the device specified by (s) to the (n) points of the device specified by (d).

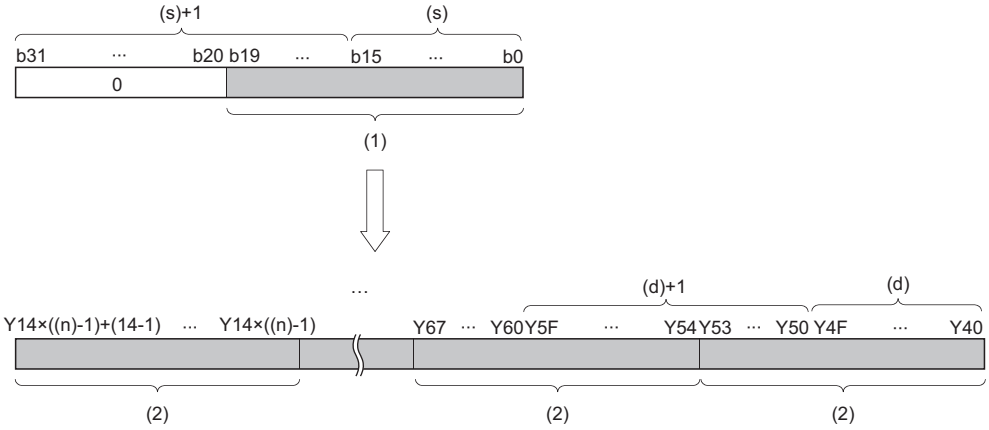


- When the number of digits is specified for the data in the device specified by (s), only the data corresponding to the specified digits are transferred. When K5Y40 is specified in (s), the lower 20 bits (five digits) of the word device specified by (s) are transferred.



(1) Ignored.
(2) Data of 20 bits (5 digits)
(3) Filled with 0s.

- When the number of digits is specified for the data in the device specified by (d), the data corresponding to the specified digits are transferred. When K5Y40 is specified in (d), the lower 20 bits of the device specified by (s) are transferred. When the number of digits is specified for both data in the devices specified by (s) and (d), the data corresponding to the digits specified in (d) are transferred.



(1) Data (20 bits (5 digits)) specified by (d)
(2) Data of 20 bits (5 digits)

- If (n) is 0, no processing is performed.

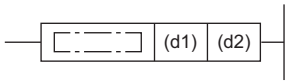
Operation error

There is no operation error.

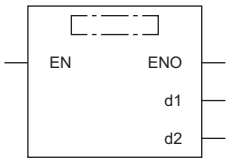
Exchanging 16-bit binary data

XCH(P)

These instructions exchange the 16-bit binary data specified.

Ladder	ST
	ENO:=XCH(EN,d1,d2); ENO:=XCHP(EN,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
XCH	
XCHP	

Setting data

■Description, range, data type

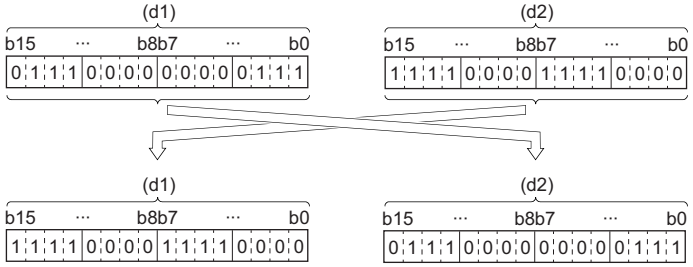
Operand	Description	Range	Data type	Data type (label)
(d1)	Device where the exchange target data is stored	—	16-bit signed binary	ANY16
(d2)	Device where the exchange target data is stored	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d1)	○	○	○	○	○	—	—	○	—	—	—	—
(d2)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instruction exchange the 16-bit binary data between the devices specified by (d1) and (d2).



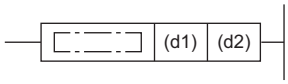
Operation error

There is no operation error.

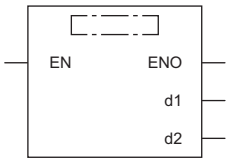
Exchanging 32-bit binary data

DXCH(P)

These instructions exchange the 32-bit binary data specified.

Ladder	ST
	ENO:=DXCH(EN,d1,d2); ENO:=DXCHP(EN,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DXCH	
DXCHP	

Setting data

■Description, range, data type

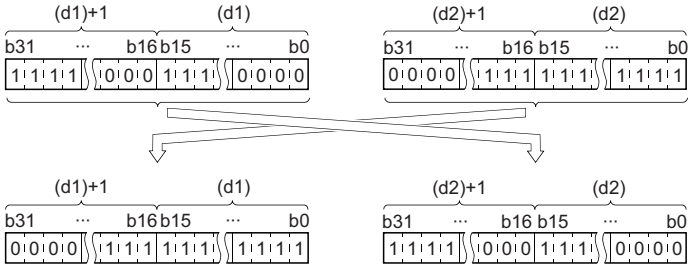
Operand	Description	Range	Data type	Data type (label)
(d1)	Start device where the exchange target data is stored	—	32-bit signed binary	ANY32
(d2)	Start device where the exchange target data is stored	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d1)	○	○	○	○	○	○	○	○	—	—	—	—
(d2)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instruction exchange the 32-bit binary data between the device ranges specified by (d1) and (d2).



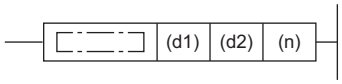
Operation error

There is no operation error.

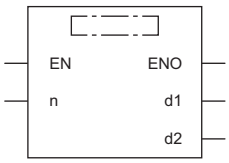
Exchanging 16-bit binary block data

BXCH(P)

These instructions exchange the (n) points of 16-bit binary data starting from the devices specified.

Ladder	ST
	ENO:=BXCH(EN,n,d1,d2); ENO:=BXCHP(EN,n,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
BXCH	
BXCHP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d1)	Start device where the exchange target data is stored	—	Word	ANY16*1
(d2)	Start device where the exchange target data is stored	—	Word	ANY16*1
(n)	Number of exchange data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

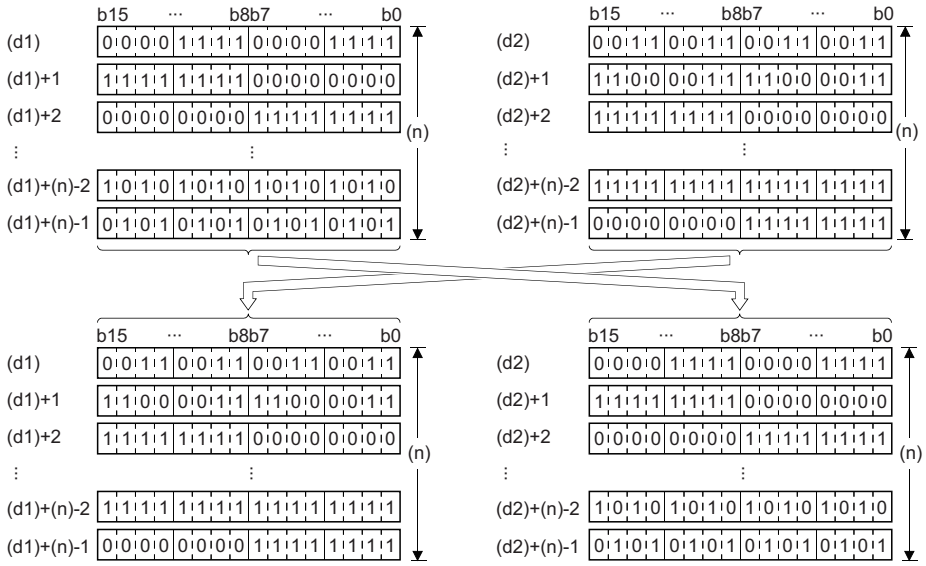
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions exchange the (n) points of 16-bit binary data starting from the device specified by (d1) with the (n) points of 16-bit binary data starting from the device specified by (d2).



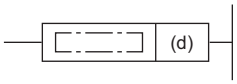
Operation error

Error code (SD0)	Description
2821H	The device ranges specified by (d1) and (d2) are overlapping.

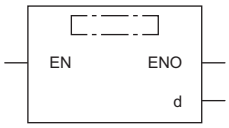
Exchanging the upper and lower bytes of 16-bit binary data

SWAP(P)

These instructions exchange upper and lower 8-bit data in the specified device.

Ladder	ST
	ENO:=SWAP(EN,d); ENO:=SWAPP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SWAP	
SWAPP	

Setting data

■Description, range, data type

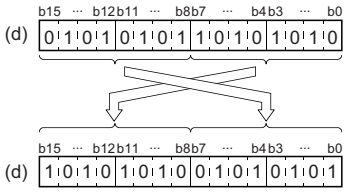
Operand	Description	Range	Data type	Data type (label)
(d)	Start device for storing the data whose upper and lower 8-bit data is exchanged	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- The device specified by (d) exchanges its upper and lower 8-bit data.



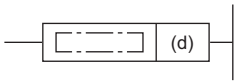
Operation error

There is no operation error.

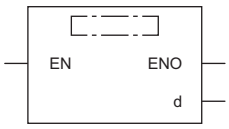
Exchanging the upper and lower bytes of 32-bit binary data

DSWAP(P)

These instructions exchange upper and lower 8-bit data in the specified device.

Ladder	ST
	ENO:=DSWAP(EN,d); ENO:=DSWAPP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSWAP	
DSWAPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device for storing the data whose upper and lower 8-bit data is exchanged	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

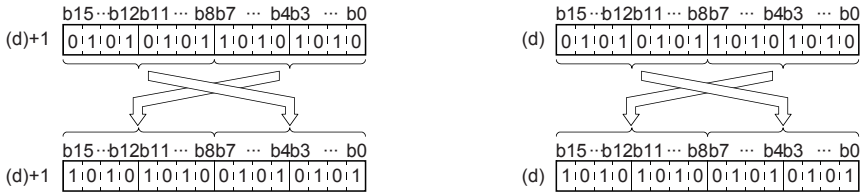
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	○	○	○	○	○	○	○	—	—	—	—

*1 FX and FY cannot be used.

Processing details

- The device specified by (d) exchanges its upper and lower 8-bit data.



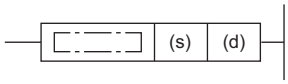
Operation error

There is no operation error.

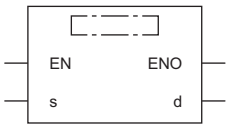
Transferring 1-bit data

MOVB(P)

These instructions transfer the specified 1-bit data.

Ladder	ST
	ENO:=MOVB(EN,s,d); ENO:=MOVBP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
MOVB	
MOVBP	

Setting data

■Description, range, data type

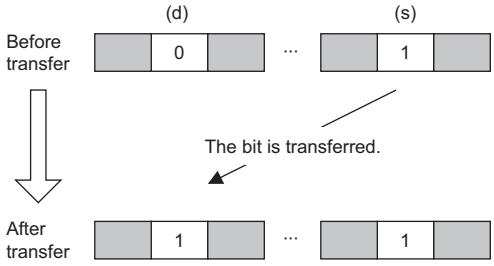
Operand	Description	Range	Data type	Data type (label)
(s)	Number of the device where the transfer target data is stored	—	Bit	ANY_BOOL
(d)	Transfer destination device number	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	—	—	○	—	○	—	—	—	—
(d)	○	○	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions transfer the bit data in the device specified by (s) to the device specified by (d).



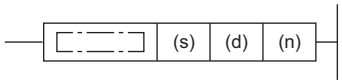
Operation error

There is no operation error.

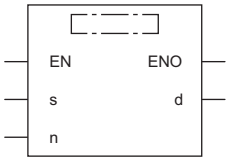
Transferring n-bit data

BLKMOVB(P)

These instructions batch-transfer the (n) points of bit data.

Ladder	ST
	ENO:=BLKMOVB(EN,s,n,d); ENO:=BLKMOVBP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BLKMOVB	
BLKMOVBP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Transfer source block data	—	Bit	ANY_BOOL*1
(d)	Transfer destination block data	—	Bit	ANY_BOOL*1
(n)	Number of transfer data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

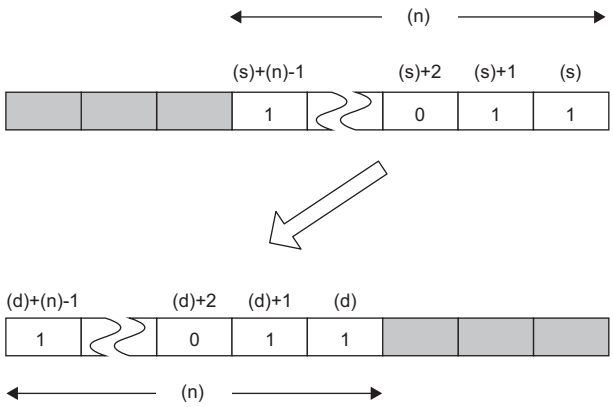
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions batch-transfer the (n) points of bit data starting from the device specified by (s) to the (n) points of bits starting from the device specified by (d).
- Data can be transferred even when the transfer source device and destination device overlap.



Operation error

There is no operation error.

PART 4 APPLICATION INSTRUCTIONS

This part consists of the following chapters.

13 PROGRAM CONTROL

14 DATA PROCESSING

15 DEBUGGING AND FAILURE DIAGNOSTIC

16 STRING PROCESSING

17 REAL VALUE PROCESSING

18 RANDOM NUMBER

19 DEVICE OPERATION

20 TIMER, COUNTER

21 CLOCK

22 MODULE ACCESS

23 PID OPERATION INSTRUCTION

24 PID CONTROL INSTRUCTIONS

25 ADD-ON

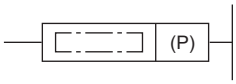
13 PROGRAM CONTROL

13.1 Program Branch Instructions

Pointer branch

CJ, SCJ, JMP

- CJ: This instruction executes the program specified by the pointer number within the same program file.
- SCJ: This instruction executes the program specified by the pointer number within the same program file starting with the next scan.
- JMP: This instruction unconditionally executes the program specified by the pointer number within the same program file.

Ladder	ST
	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
CJ SCJ	
JMP	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(P)	Pointer number of the jump destination	—	Device name	POINTER

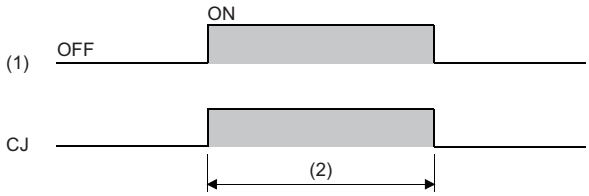
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	(P)
(P)	—	—	—	—	—	—	—	—	—	—	—	○

Processing details

■CJ

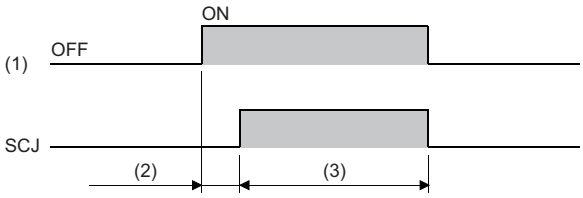
- This instruction executes the program specified by the pointer number within the same program file when the execution command is on.
- When the execution command is off, the program in the next step is executed.



(1) Execution command
(2) Executed in each scan

■SCJ

- This instruction executes the program specified by the pointer number within the same program file starting with the scan immediately after the execution command turns on.
- When the execution command is off or turned off, the program in the next step is executed.



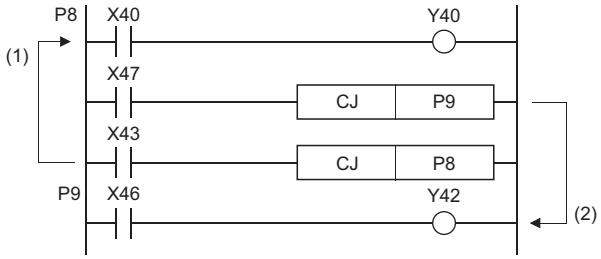
(1) Execution command
(2) One scan
(3) Executed in each scan

■JMP

- This instruction unconditionally executes the program specified by the pointer number within the same program file.

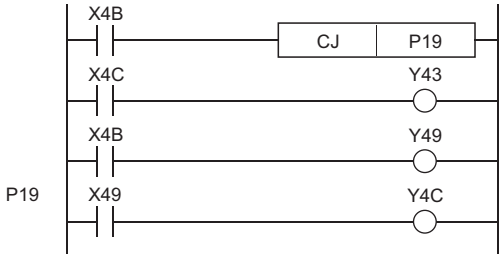
Precautions

- If the timer with its coil on is skipped by these instructions, time cannot be measured correctly.
- If the OUT instruction is skipped by these instructions, the scan time will be shortened.
- If these instructions specify and jump to the program with a bigger step number, the scan time will be shortened.
- These instructions can specify and jump to the program with a smaller step number. In this case, consider a method to exit a loop so that the watchdog timer does not time out.



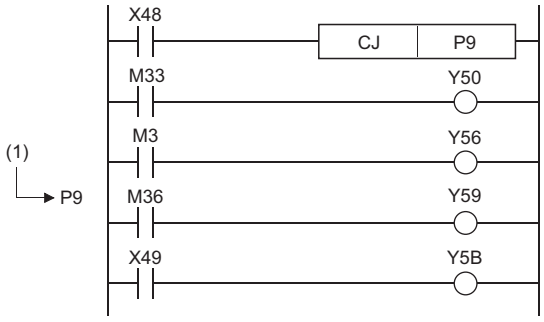
(1) While X43 is on, the loop is repeated.
(2) To exit the loop, turn on X47.

- The value in the device skipped with these instructions remains unchanged.



When X4B turns on, the program jumps to the label, P19.
Y43 and Y49 remain unchanged even if X4B and X4C turn on/off during the execution of the CJ instruction.

- A label (P□) occupies one step.



(1) A label occupies one step.

- Only the pointer numbers within the same program file can be specified.

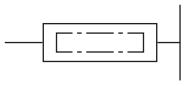
Operation error

Error code (SD0)	Description
3340H	The pointer number specified by (P) is not set before the END instruction.
	The pointer number specified by (P) is not used as a label in the same program.
	The pointer number specified by (P) is a global pointer in another program.

Jumping to END

GOEND

This instruction jumps the program to the FEND or END instruction within the same program file.

Ladder	ST
	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
GOEND	

Processing details

- This instruction jumps the program to the FEND or END instruction within the same program file.

Operation error

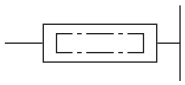
Error code (SD0)	Description
3341H	After execution of the CALL(P) or ECALL(P) instruction, the GOEND instruction is executed before the RET instruction.
3351H	The GOEND instruction is executed before the IRET instruction in the interrupt program specified by the interrupt pointer (I).

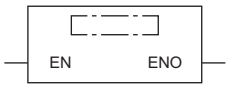
13.2 Program Execution Control Instructions

Disabling/enabling interrupt programs

DI, EI

- DI: This instruction disables execution of interrupt programs.
- EI: This instruction clears the interrupt disabled state.

Ladder	ST
	ENO:=DI(EN); ENO:=EI(EN);

FBD/LD


■Execution condition

Instruction	Execution condition
DI EI	

Processing details

■DI

- This instruction disables execution of interrupt programs until the EI instruction is executed even though they are requested.
- For the operation of the DI (Disabling interrupt programs) instruction used with the DI (Disabling interrupt programs with specified priority or lower) instruction, refer to the following.

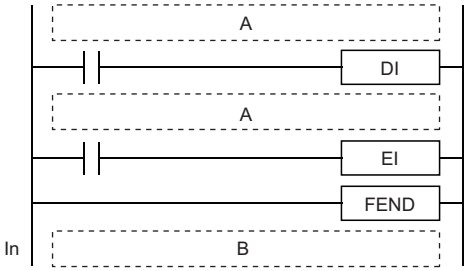
☞ Page 539 DI

- The DI (Disabling interrupt programs) instruction cannot be executed in interrupt programs. If executed, no processing is performed.

■EI

- This instruction clears the interrupt disabled state that has been set by the DI (Disabling interrupt programs) instruction, and enables execution of interrupt programs with the interrupt pointer numbers permitted by the IMASK instruction. When the IMASK instruction is not executed, I32 to I43 are disabled.
- When the system is powered on or the controller is reset and moved to RUN, the system is in the state where the EI instruction has been executed. When the initial execution type program is set, the system is in the state where the DI instruction has been executed during the execution of the initial execution type program in the first scan after RUN, and where the EI instruction has been executed from the execution of the program in the second scan after RUN.
- For the operation of the EI instruction used with the DI (Disabling interrupt programs with specified priority or lower) instruction, refer to the following.

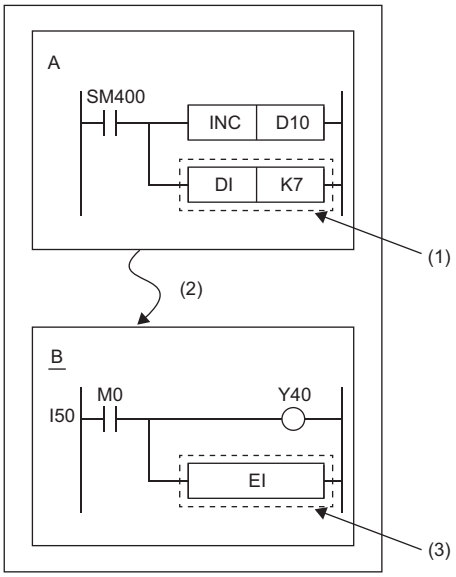
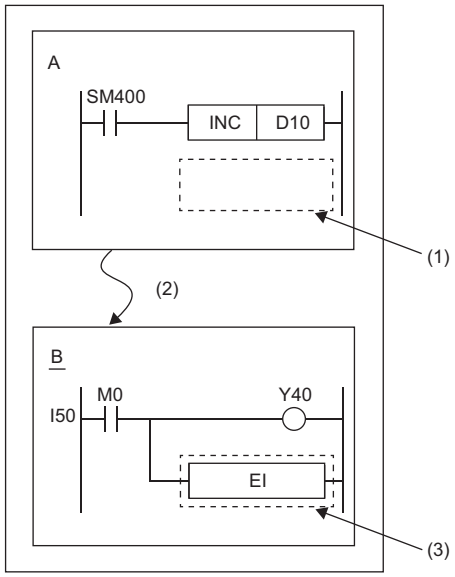
☞ Page 539 DI



Even if an interrupt factor occurs between the DI and EI instructions, the corresponding interrupt program will not be executed until the processing between the DI and EI instructions completes.

A: Sequence program
B: Interrupt program

- The operation of the EI instruction in interrupt programs differs depending on the execution status of the DI (Disabling interrupt programs with specified priority or lower) instruction before the EI instruction. The EI instruction in the interrupt program after the execution of the DI (Disabling interrupt programs with specified priority or lower) instruction can be executed. The following shows whether the EI instruction can be executed in the interrupt program.

When the EI instruction is executed in the interrupt program after the DI (Disabling interrupt programs with specified priority or lower) instruction was executed in the main routine program	When the EI instruction is executed in the interrupt program without executing the DI (Disabling interrupt programs with specified priority or lower) instruction in the main routine program
	
A: Main routine program B: Interrupt program (1) The DI (Disabling interrupt programs with specified priority or lower) instruction is executed. (2) An interrupt has occurred. (3) Execution of the interrupt program is enabled.	A: Main routine program B: Interrupt program (1) The DI (Disabling interrupt programs with specified priority or lower) instruction is not executed. (2) An interrupt has occurred. (3) Execution of the interrupt program is disabled. (No processing is performed.)

- EI and DI instructions in the initial execution type program can be executed. However, if the EI and DI instructions are executed in the initial execution type program, the specified priority setting is valid until the second scan is started. The system is in the state where the EI instruction has been executed from the time when the second scan of the program is executed.

Point

Either create a program by minimizing the interrupt disabled section, or exclude network communication cycle synchronization interrupts by using the DI (Disabling interrupt programs with specified priority or lower) instruction. If the DI instruction is used, interference with network communication cycle synchronization interrupts occurs, causing malfunction in the following functions.

- Motion function
- Network cycle communication function

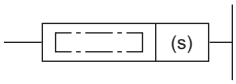
Operation error

Error code (SD0)	Description
3332H	More than 16 DI (Disabling interrupt programs) instructions and DI (Disabling interrupt programs with specified priority or lower) instructions are nested.

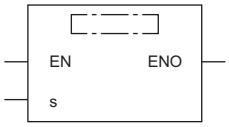
Disabling interrupt programs with specified priority or lower

DI

This instruction disables execution of interrupt programs with the specified priority or lower.

Ladder	ST
	ENO:=DI_1(EN,s);

FBD/LD



(☐ is replaced by DI_1.)

■Execution condition

Instruction	Execution condition
DI	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Interrupt disable priority	1 to 32	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

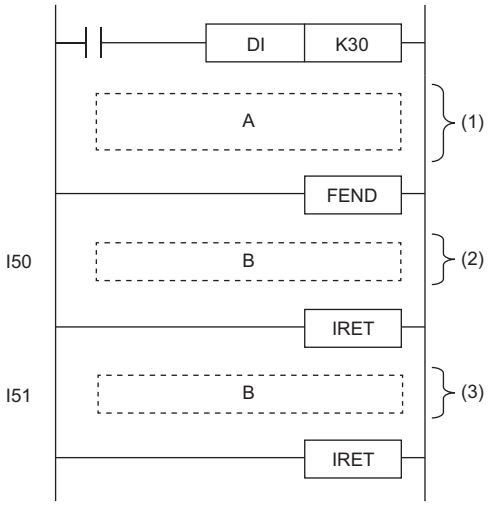
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	○	—	—	○	○	—	—	—

Processing details

• This instruction disables execution of interrupt programs with the interrupt pointer numbers specified by (s) or lower.

Interrupt priority setting

I No.	Priority
I50	29
I51	30



A: Sequence program

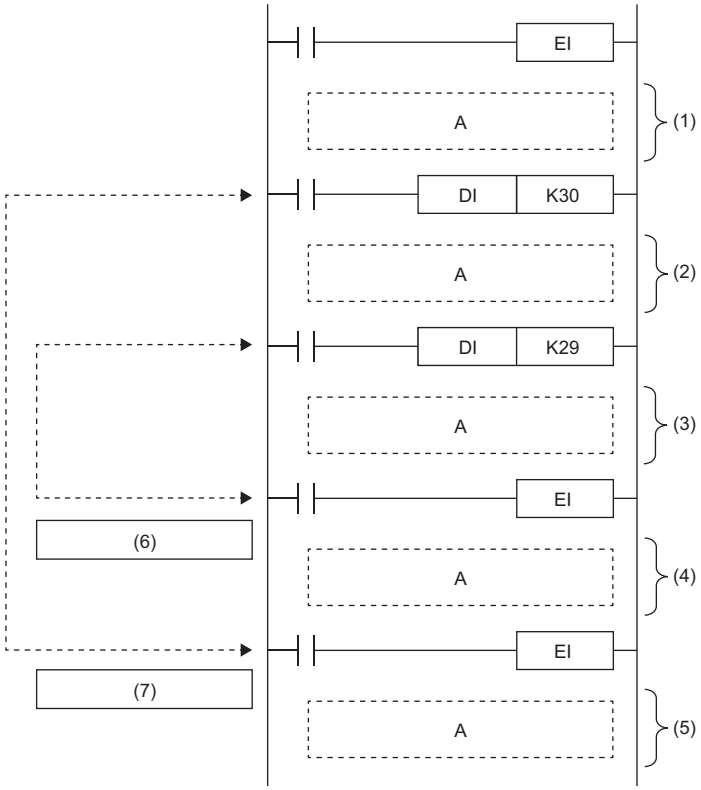
B: Interrupt program

(1) Interrupt disabled section for priority 30 or less (Interrupt enabled section for priority 29 or above)

(2) The interrupt program can be executed because its priority is 29.

(3) The interrupt program cannot be executed because its priority is 30.

- The execution of the EI instruction enables the interrupt that has been disabled by a single DI (Disabling interrupt programs with specified priority or lower) instruction. Note that if interrupts have been disabled only by the DI (Disabling interrupt programs) instruction, executing the EI instruction only once enables interrupts in all priorities.

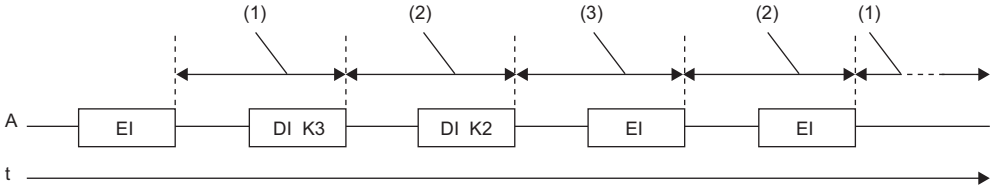


A: Sequence program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 30 or less (Interrupt enabled section for priority 29 or above)
(3) Interrupt disabled section for priority 29 or less (Interrupt enabled section for priority 28 or above)
(4) Interrupt disabled section for priority 30 or less (Interrupt enabled section for priority 29 or above)
(5) Interrupt enabled section for all priorities
(6) EI instruction for [DI K29]
(7) EI instruction for [DI K30]

- When multiple DI (Disabling interrupt programs with specified priority or lower) instructions are executed and the specified interrupt disable priority is lower than the priority of the currently disabled interrupt, the priority of the currently disabled interrupt remains unchanged.
- Up to 16 DI instructions can be nested.
- The priority of the interrupt pointer can be set by parameters. (User's manual for the controller used)
- The interrupt disabled priority currently set can be checked in SD758.
- If the DI (Disabling interrupt programs with specified priority or lower) instruction is executed in the interrupt program and the interrupt disabled priority is changed, the value in SD758 also changes.

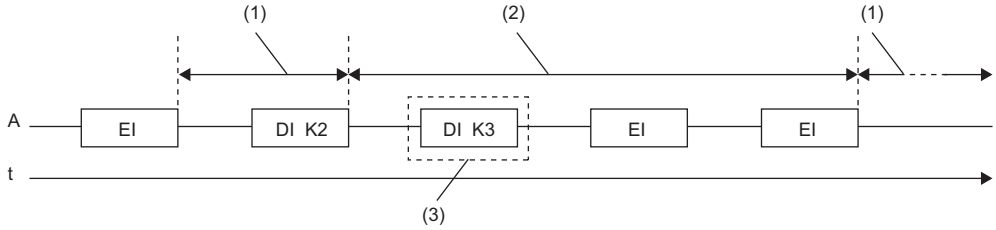
- When the DI (Disabling interrupt programs) instruction, DI (Disabling interrupt programs with specified priority or lower) instruction, and EI instruction are executed, the interrupt disabled sections will be as follows.

  • When another DI (Disabling interrupt programs with specified priority or lower) instruction with a wider priority range is executed during execution of the DI (Disabling interrupt programs with specified priority or lower) instruction



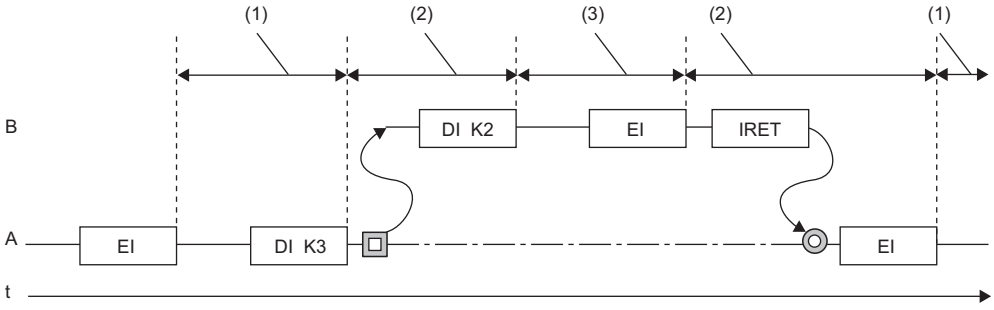
A: Scan execution type program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 3 or less (Interrupt enabled section for priority 2 or above)
(3) Interrupt disabled section for priority 2 or less (Interrupt enabled section for priority 1 or above)

  • When another DI (Disabling interrupt programs with specified priority or lower) instruction with a narrower priority range is executed during execution of the DI (Disabling interrupt programs with specified priority or lower) instruction



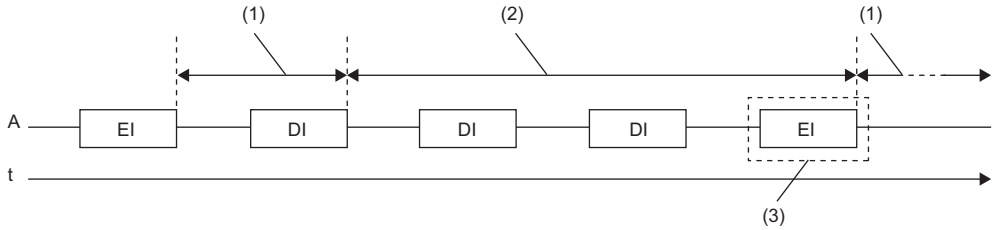
A: Scan execution type program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 2 or less (Interrupt enabled section for priority 1 or above)
(3) The priority of the disabled interrupt remains unchanged because the interrupt with priority 2 or lower is already disabled.

  • When the DI (Disabling interrupt programs with specified priority or lower) instruction is executed in the interrupt program



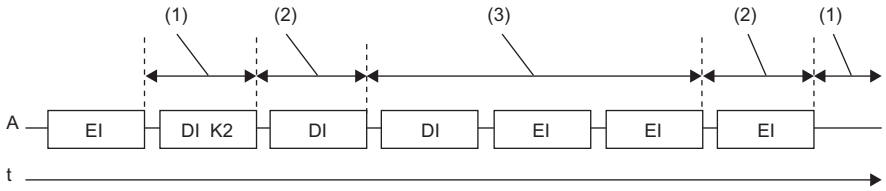
A: Scan execution type program
B: Interrupt program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 3 or less (Interrupt enabled section for priority 2 or above)
(3) Interrupt disabled section for priority 2 or less (Interrupt enabled section for priority 1 or above)

  • When only the DI (Disabling interrupt programs) instruction is executed



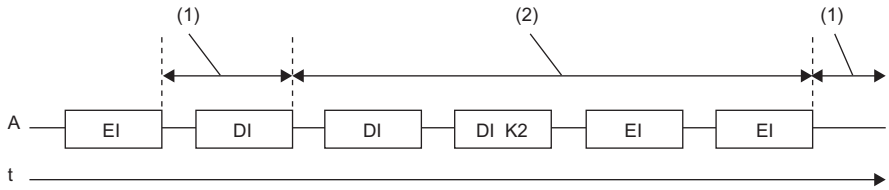
A: Scan execution type program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 1 or less (Interrupt disabled section for all priorities)
(3) Executing the EI instruction only once enables interrupts with all priorities.

• When the DI (Disabling interrupt programs with specified priority or lower) instruction and the DI (Disabling interrupt programs) instruction are executed in this order



A: Scan execution type program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 2 or less (Interrupt enabled section for priority 1 or above)
(3) Interrupt disabled section for priority 1 or less (Interrupt disabled section for all priorities)

• When the DI (Disabling interrupt programs) instruction and the DI (Disabling interrupt programs with specified priority or lower) instruction are executed in this order



A: Scan execution type program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 1 or less (Interrupt disabled section for all priorities)

• The DI instruction in the initial execution type program can be executed. However, if the DI instruction is executed in the initial execution type program, the specified priority setting is valid until the second scan is started. The system is in the state where the EI instruction has been executed from the time when the second scan of the program is executed.

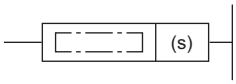
Operation error

Error code (SD0)	Description
3285H	The priority specified by (s) is out of range.
3332H	More than 16 DI (Disabling interrupt programs) instructions and DI (Disabling interrupt programs with specified priority or lower) instructions are nested.

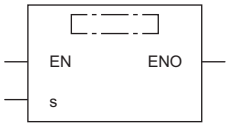
Interrupt program mask

IMASK

This instruction enables or disables the execution of the interrupt program with the specified interrupt pointer number.

Ladder	ST
	ENO:=IMASK(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
IMASK	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Interrupt mask data or the start device where the interrupt mask data is stored	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 16)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- This instruction enables or disables the execution of the interrupt program with the interrupt pointer number specified by the 16 points of bit pattern starting from the device specified by (s).
  - 1 (on): Enables the execution of the interrupt program.
  - 0 (off): Disables the execution of the interrupt program.
- The interrupt pointer numbers correspond to individual bits as shown below.

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
(s)	I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
(s)+1	I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
(s)+2	I47	I46	I45	I44	I43	I42	I41	I40	I39	I38	I37	I36	I35	I34	I33	I32
(s)+3	I63	I62	I61	I60	I59	I58	I57	I56	I55	I54	I53	I52	I51	I50	I49	I48
(s)+4	I79	I78	I77	I76	I75	I74	I73	I72	I71	I70	I69	I68	I67	I66	I65	I64
(s)+5	I95	I94	I93	I92	I91	I90	I89	I88	I87	I86	I85	I84	I83	I82	I81	I80
(s)+6	I111	I110	I109	I108	I107	I106	I105	I104	I103	I102	I101	I100	I99	I98	I97	I96
(s)+7	I127	I126	I125	I124	I123	I122	I121	I120	I119	I118	I117	I116	I115	I114	I113	I112
(s)+8	I143	I142	I141	I140	I139	I138	I137	I136	I135	I134	I133	I132	I131	I130	I129	I128
(s)+9	I159	I158	I157	I156	I155	I154	I153	I152	I151	I150	I149	I148	I147	I146	I145	I144
(s)+10	I175	I174	I173	I172	I171	I170	I169	I168	I167	I166	I165	I164	I163	I162	I161	I160
(s)+11	I191	I190	I189	I188	I187	I186	I185	I184	I183	I182	I181	I180	I179	I178	I177	I176
(s)+12	I207	I206	I205	I204	I203	I202	I201	I200	I199	I198	I197	I196	I195	I194	I193	I192
(s)+13	I223	I222	I221	I220	I219	I218	I217	I216	I215	I214	I213	I212	I211	I210	I209	I208
(s)+14	I239	I238	I237	I236	I235	I234	I233	I232	I231	I230	I229	I228	I227	I226	I225	I224
(s)+15	I255	I254	I253	I252	I251	I250	I249	I248	I247	I246	I245	I244	I243	I242	I241	I240

- When the system is powered on or the controller is reset, the execution of interrupt programs I0 to I31 and I44 to I1023 is enabled, and the execution of interrupt programs I32 to I43 is disabled.
- The states of the device range (s) to (s)+15 are stored in SD1400 to SD1463 (IMASK instruction mask pattern).

Point

The IMASK instruction can enable or disable the execution of interrupt pointers I0 to I255 altogether.
To enable or disable the execution of interrupt pointers I256 to I1023, substitute the SIMASK instruction for the IMASK instruction in the program.
For details on the SIMASK instruction, refer to the following.
☞ Page 546 SIMASK

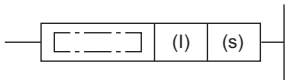
Operation error

There is no operation error.

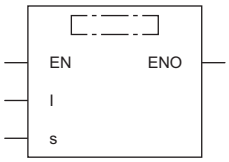
Disabling/enabling the specified interrupt pointer

SIMASK

This instruction enables or disables the execution of the interrupt program with the specified interrupt pointer number.

Ladder	ST
	ENO:=SIMASK(EN,I,s);

FBD/LD



■Execution condition

Instruction	Execution condition
SIMASK	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(I)	Interrupt pointer number to be interrupt-enabled or -disabled	I0 to I1023	Device name	POINTER*1
(s)	Execution status of the specified interrupt pointer	0: Disabled 1: Enabled	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to the device (I) can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	(I)
(I)	—	—	—	—	—	—	—	—	—	—	—	○
(s)	—	—	○	—	○	—	—	○	○	—	—	—

Processing details

- This instruction, according to the data specified by (s), enables or disables the execution of the interrupt program with the interrupt pointer number specified by (I).
  - When (s) is 1: The execution of the interrupt program is enabled.
  - When (s) is 0: The execution of the interrupt program is disabled.
- When the system is powered on or the controller is reset, the execution of interrupt programs I0 to I31 and I44 to I1023 is enabled, and the execution of interrupt programs I32 to I43 is disabled.
- The execution status of the interrupt pointers is stored in SD1400 to SD1463 (IMASK instruction mask pattern).

Point

The device (I) can be index modified. Using the SIMASK instruction specifying the index-modified device can enable or disable the execution of interrupt pointers I0 to I1023.

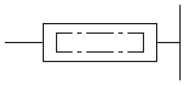
Operation error

Error code (SD0)	Description
3285H	The interrupt pointer number specified by (I) is out of the valid range.
	The value in the device specified by (s) is neither 0 (disabled) nor 1 (enabled).

Returning from the interrupt program

IRET

This instruction indicates the end of the processing of an interrupt program.

Ladder	ST
	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
IRET	Every scan

Processing details

- This instruction indicates the end of the processing of the interrupt program specified by the interrupt pointer (I).
- The instruction returns control to the sequence program after execution.

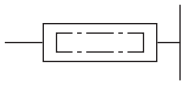
Operation error

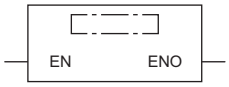
Error code (SD0)	Description
3350H	There is no pointer corresponding to the interrupt number.
3351H	After an interrupt occurs, the END, FEND, GOEND, or STOP instruction is executed before the IRET instruction.
3352H	The IRET instruction is executed in a main routine program or subroutine program.
3353H	The IRET instruction is executed in a fixed scan execution type program.
3354H	The IRET instruction is executed in an event execution type program.

Resetting the watchdog timer

WDT(P)

These instructions reset the watchdog timer.

Ladder	ST
	ENO:=WDT(EN); ENO:=WDTP(EN);

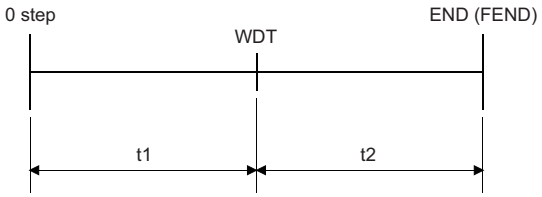
FBD/LD


■Execution condition

Instruction	Execution condition
WDT	
WDTP	

Processing details

- These instructions reset the watchdog timer in a program.
- These instructions are used when the scan time exceeds the value set for the watchdog timer depending on the condition. If the scan time exceeds the value set for the watchdog timer every scan, change the watchdog timer value in parameter using the engineering tool.
- The time values required for t1 from step 0 to the WDT(P) instruction and t2 from the WDT(P) instruction to the END(FEND) instruction must not exceed the value set for the watchdog timer.



- The instruction can be used twice or more in a single scan, but it takes time to turn off the output when an error occurs.
- Executing the instruction does not clear the scan time value stored in the special register. For this reason, the scan time value in the special register may be greater than the value set for the watchdog timer in parameter.

Operation error

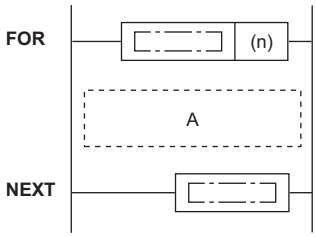
There is no operation error.

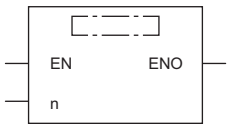
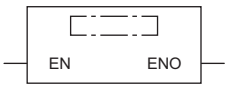
13.3 Structure Creation Instructions

Performing the FOR to NEXT instruction loop

FOR, NEXT

These instructions execute the processing between the FOR instruction and the NEXT instruction (n) times.

Ladder	ST
 A: Repeat program	Not supported

FBD/LD	
[FOR] 	[NEXT] 

■Execution condition

Instruction	Execution condition
FOR NEXT	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(n)	Number of FOR to NEXT instruction loops	1 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

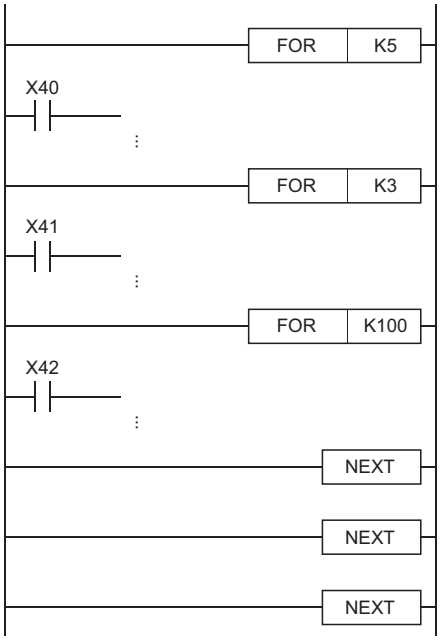
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- After the FOR to NEXT instruction loop is executed unconditionally (n) times, the step following the NEXT instruction is executed.
- Specify a value 1 to 32767 for (n). If a value -32768 to 0 is specified, the processing is performed regarding (n) as 1.
- To not to execute the FOR to NEXT instruction loop, use the CJ or SCJ instruction to jump the instruction loop.

- Up to 16 FOR instructions can be nested.

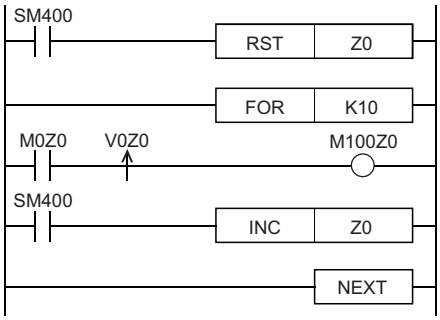


Precautions

- To terminate the FOR to NEXT instruction loop being executed, use the BREAK instruction.

☞ Page 553 BREAK(P)

- To perform pulse operations of an index-modified program within the FOR to NEXT instruction loop, use the EGP or EGF instruction. Note, however, that no rising edge or falling edge instruction can be used on the operation output side.



- The JMP instruction cannot be used to branch from outside the FOR to NEXT instruction loop to inside the FOR to NEXT instruction loop, and from inside the FOR to NEXT instruction loop to outside the FOR to NEXT instruction loop, resulting in a conversion error.
- If the pointer device/label specified by the JMP instruction inside the FOR to NEXT instruction loop is index-modified, the branch cannot be executed, resulting in a conversion error.
- The CALL(P) instruction cannot be used to branch from outside the FOR to NEXT instruction loop to inside the FOR to NEXT instruction loop, resulting in a conversion error. However, if the pointer device/label specified as the branch destination in the CALL(P) instruction is index-modified and a pointer that exists in the FOR to NEXT instruction loop is specified after resolving the index modification, an error (error code: 3340H) occurs when the CALL(P) instruction is executed. For details on the CALL(P) instruction, refer to the following.

☞ Page 555 CALL(P)

- For a nested structure within FOR to NEXT instruction loop, the pointer device/label specified by the JMP and other instructions can be described only if it is in the same hierarchy as the JMP and other instructions, which is the execution origin. Describing in a different hierarchy results in a convention error.
- If the FOR and NEXT instructions are used in the FBD/LD languages, specifying a variable or constant which is not always on (TRUE/1/SM400) in EN (execution condition) will cause a conversion error.

Operation error

There is no operation error.

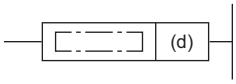
Point

To create an easy-to-understand program, use a pair of instructions, the FOR and NEXT instructions, within a single program block.

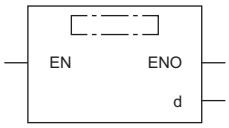
Forcibly terminating the FOR to NEXT instruction loop

BREAK(P)

These instructions forcibly terminate the loop processing between the FOR and NEXT instructions, and pass the control to the processing of FOR to NEXT instruction or later.

Ladder	ST
	Not supported

FBD/LD



■Execution condition

Instruction	Execution condition
BREAK	
BREAKP	

Setting data

■Description, range, data type

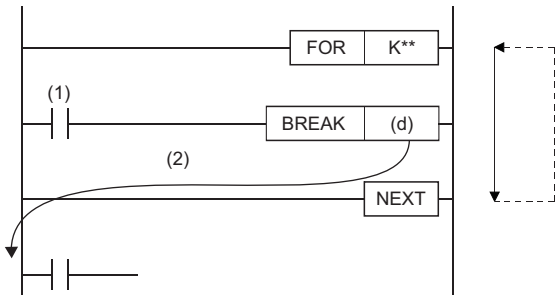
Operand	Description	Range	Data type	Data type (label)
(d)	Number of the device for storing the remaining number of loops	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions forcibly terminate the loop processing between the FOR and NEXT instructions, and pass the control to the processing of FOR to NEXT instruction or later.



If no BREAK instruction is executed, the program performs loop processing as many times as specified by the FOR instruction.

(1) Forced termination condition

(2) When the condition is met

- The remaining number of FOR to NEXT instruction loops at the time of forced termination is stored in (d). The remaining number of loops includes the processing when the BREAK(P) instruction is executed.
- The BREAK(P) instruction can be used only within the FOR to NEXT instruction loop.
- The BREAK(P) instruction is valid only for one level of nesting. To forcibly terminate multiple levels of nesting, execute as many BREAK(P) instructions as nesting levels.

Point

When applying a program from a different series, immediately after the NEXT instruction ends, write the process to be performed when the forced termination condition written to the branch destination pointer is met.

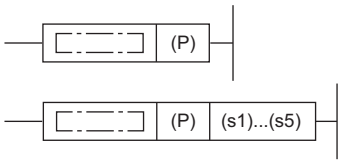
Operation error

There is no operation error.

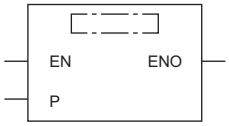
Calling a subroutine program

CALL(P)

These instructions execute the subroutine program of the specified pointer.

Ladder	ST
	Not supported

FBD/LD



■Execution condition

Instruction	Execution condition
CALL	
CALLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(P)	Start pointer number of subroutine program	—	Device name	POINTER
(s1) to (s5)*1	Device number to be passed to the subroutine program as an argument	-2147483648 to 2147483647	Bit/16-bit signed binary/32-bit signed binary	ANY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Cannot be specified in FBD/LD.

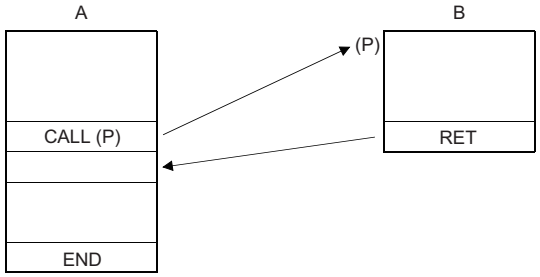
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	(P)
(P)	—	—	—	—	—	—	—	—	—	—	—	○
(s1) to (s5)	○*1	○	○	○	○	○	○	○	○	—	—	—

*1 Devices other than F can be used.

Processing details

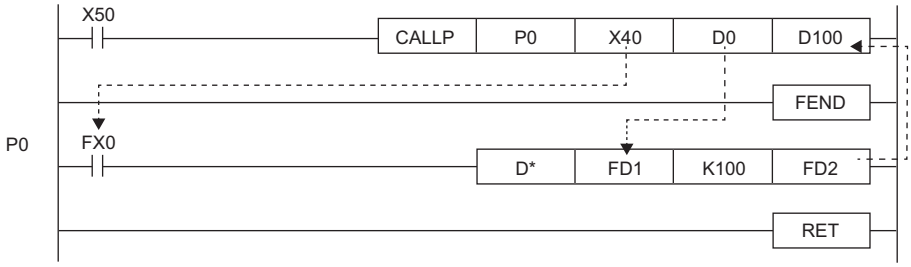
- These instructions execute the subroutine program of the pointer specified by (P). The instructions can execute the subroutine program specified by a pointer in the same program file or the subroutine program specified by a global pointer.



A: Main routine program

B: Subroutine program

- When the function device (FX, FY, or FD) is used in the subroutine program, specify the device corresponding to the function device in (s1) to (s5). The following figure shows the data of the device areas specified by (s1) to (s5).



- Before execution of the subroutine program, bit data is transferred to FX and word data is transferred to FD.
- After execution of the subroutine program, the data in FY and FD are transferred to the corresponding devices.

- The data transfer size for FX, FY, and FD differs depending on the data type specified in (s1) to (s5).

■When a device is specified

Function devices FX and FY are transferred in units of bits. Function device FD is transferred in units of 4 words. The size of data to be processed varies depending on the type of the device specified by an argument.

The device specified as a function device should be secured for the data size. An error occurs if an appropriate data area cannot be secured.

The following table lists the data sizes of each function device.

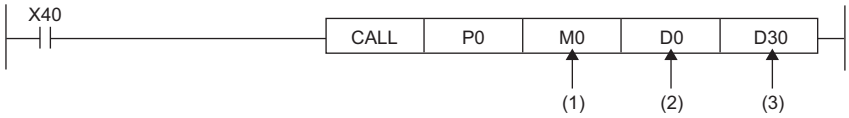
Function device	Device to be used	Sizes of data when transferred to function devices from (s1) to (s5)	Sizes of data when transferred from function devices to (s1) to (s5)
FX, FY	Bit device	1 bit	1 bit
	Bit-specified word device	1 bit	1 bit
FD	Digit-specified bit device*1	4 words	Digit-specified size or 4 words*2
	Word device	4 words	4 words

*1 It can be specified even if the device number specified by (s1) to (s5) is not a multiple of 16 in bit device digit specification mode.

*2 If the device number is a multiple of 16 and the digit specification is K4 or K8, size of 4 words will be transferred.

Ex.

The following operation occurs when a device is specified.



(1) M0 is transferred to FX0.

(2) D0 to D3 are transferred to FD1.

(3) D30 to D33 are transferred to FD2.

■When a label is specified

When data is transferred to function devices from (s1) to (s5), function devices FX and FY are transferred in units of bits. Function device FD is transferred in units of 4 words.

When data is transferred from function devices to (s1) to (s5), function devices FX and FY are transferred in units of bits. The unit of transfer for function device FD is the label-assigned or digit specification size unit.

The following table lists the data sizes of each function device.

Function device	Label-assigned size	Sizes of data when transferred to function devices from (s1) to (s5)	Sizes of data when transferred from function devices to (s1) to (s5)
FX, FY	1 bit	1 bit	1 bit
FD	2 bits or more	4 words*1	Label-assigned size*2*3/digit specification size*4

*1 If the assigned size of the specified label/structure is less than 4 words, 0 is transferred for the area that is less than 4 words.

*2 If the assigned size of the specified label/structure is 4 words or more, 4 words are transferred.

*3 When the following labels are specified, the size to be sent differs. If a label other than the following is specified, the data is transferred with the label-assignment size.

(1) When an array label (including when an element is specified) or a structure type label is specified, the entire size of the specified label will be transferred.

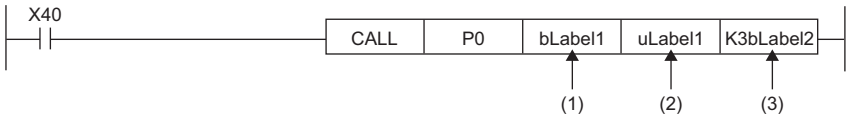
(2) When a member of the structure type label is specified, the size of the specified member will be transferred.

*4 When the digit specification is K1 to K3, 1 word will be transferred; when it is K5 to K7, 2 words will be transferred; if it is K4 or K8, the data will be transferred with the label-assignment size.

Ex.

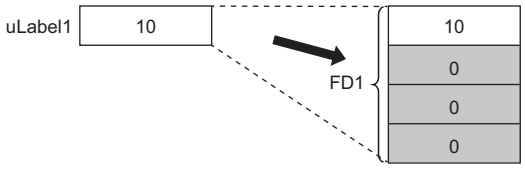
The following table lists the label names and label types when data is transferred to function devices from (s1) to (s5).

Label name	Label type
bLabel1	Bit type label
uLabel1	Word type label
bLabel2	Array bit type label

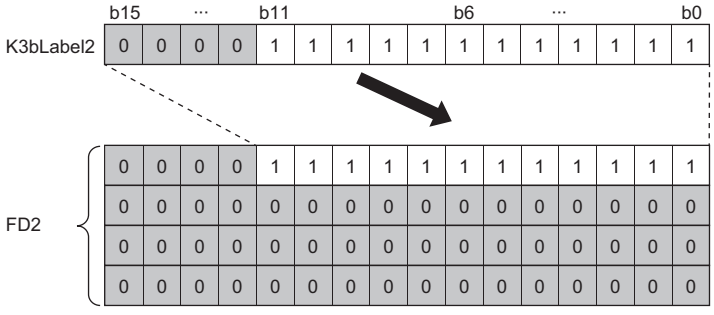


(1) bLabel1 (1 bit) is transferred to FX0.

(2) uLabel1 (1 word) is transferred to FD1. Since the assigned size is less than 4 words, 0 is transferred for the area that is less than 4 words.



(3) 1 word of K3bLabel2 is transferred to FD2. Since the assigned size is less than 4 words, 0 is transferred for the area that is less than 4 words.

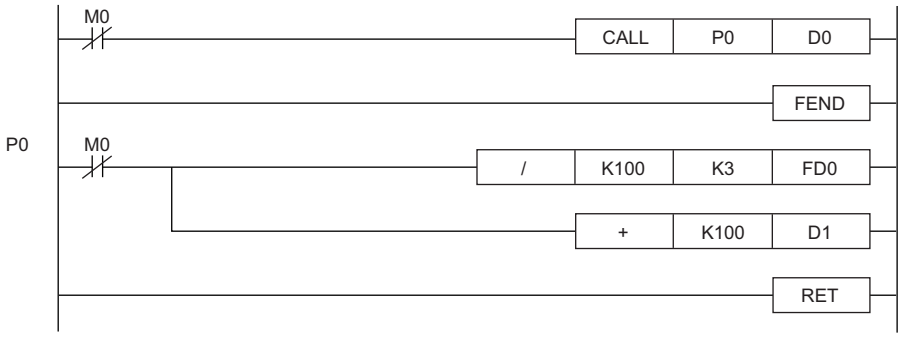


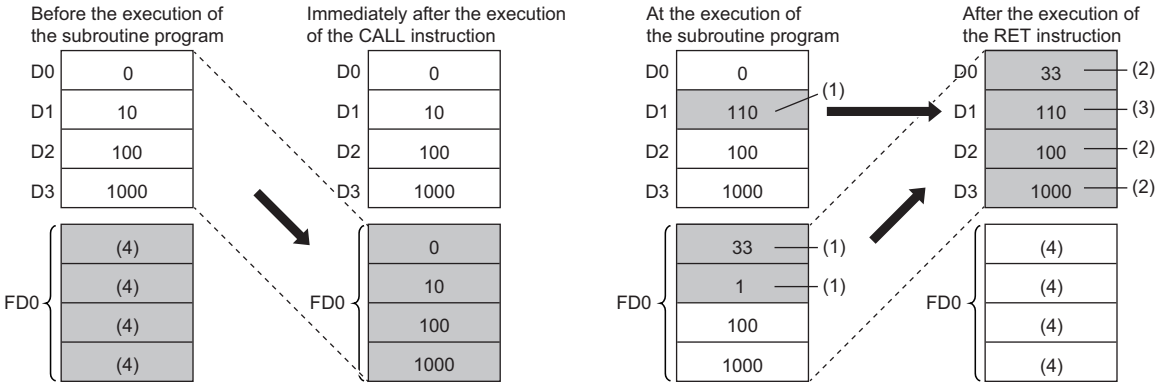
- The CALL(P) instruction can use (s1) to (s5).
- The number of function devices used in the subroutine program must be identical to the number of arguments of the CALL(P) instruction. Also, the types of function devices and CALL(P) instruction arguments must be the same.
- If the timer or counter is specified as a device in the argument of the CALL(P) instruction, only the current value is transmitted/received.
- Set the device numbers in the argument of the CALL(P) instruction so that they do not overlap. If they overlap, normal operation cannot be performed.

• Do not use any device used in the argument of the CALL(P) instruction in the subroutine program. If used, normal operation cannot be performed.

Ex.

Incorrect operation: While D0 is specified for FD0 in the subroutine program, D1 is used in the subroutine program.

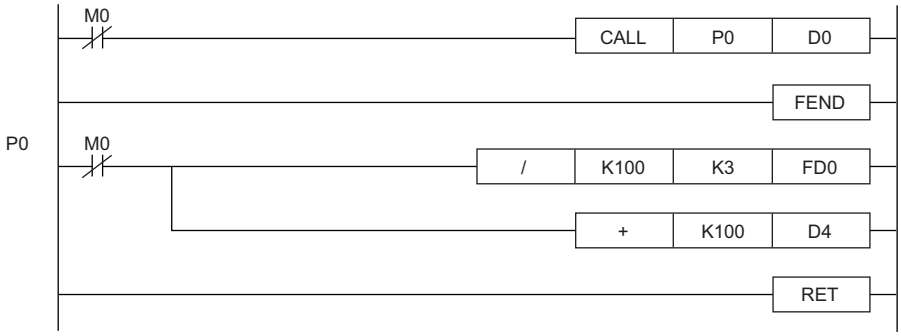


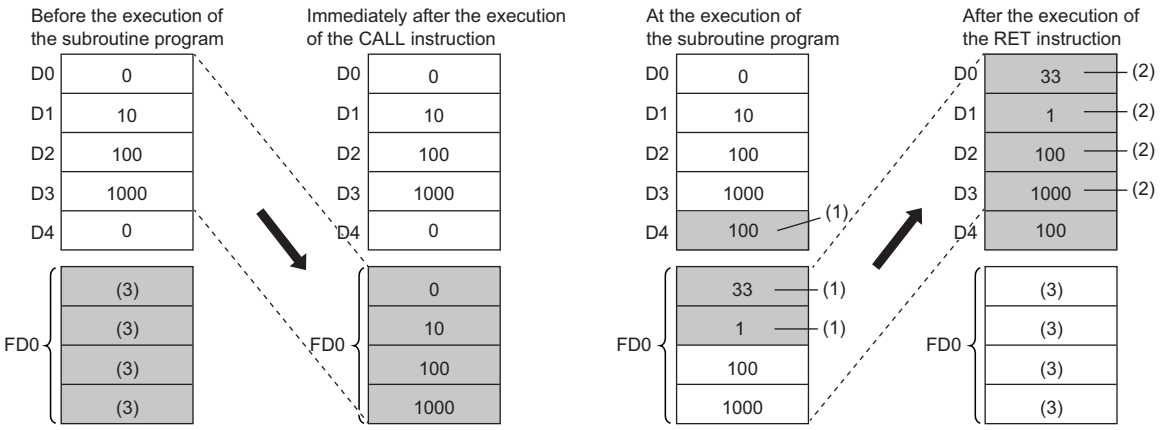


(1) The operation result of the subroutine program is stored.
(2) These values are replaced with the function device values.
(3) The value of D1 is not replaced with the function device value.
(4) Undefined values are stored.

Ex.

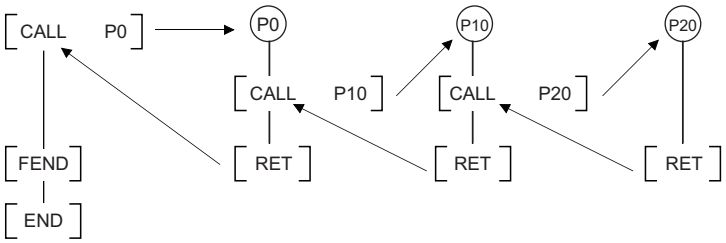
Correct operation: While D0 is specified for FD0 in the subroutine program, D4 is used in the subroutine program.





(1) The operation result of the subroutine program is stored.
(2) These values are replaced with the function device values.
(3) Undefined values are stored.

- Up to 16 CALL(P) instructions can be nested. Note that the 16-level nesting is the total of the CALL(P), FCALL(P), ECALL(P), EFCALL(P), and XCALL instructions.



- Devices which are turned on in the subroutine program hold the on state even when the subroutine program is not executed. The on state can be changed to off by executing the FCALL(P) instruction.

Precautions

- An FBD/LD program cannot be created as a subroutine program.
- FBD/LD does not support the execution of a subroutine program with an argument.
- When the pointer specified by (P) is index-modified, an error (error code: 3340H) occurs if a pointer that exists in FOR to NEXT is specified after the index modification is resolved. If a pointer that exists in FOR to NEXT is specified directly without index modification, a conversion error occurs.

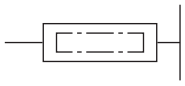
Operation error

Error code (SD0)	Description
2820H	The device specified in an argument from (s1) to (s5) cannot be secured for the data size.
3330H	The 17th level of the nesting is executed.
3340H	The subroutine program corresponding to the pointer specified by (P) does not exist. Alternatively, the pointer specified by (P) is index-modified, and the pointer that exists in FOR to NEXT is specified after the index modification is resolved.
3341H	After execution of the CALL(P) instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3342H	The RET instruction is executed before the CALL(P) instruction.

Returning from the subroutine program called

RET

This instruction indicates the end of a subroutine program.

Ladder	ST
	Not supported

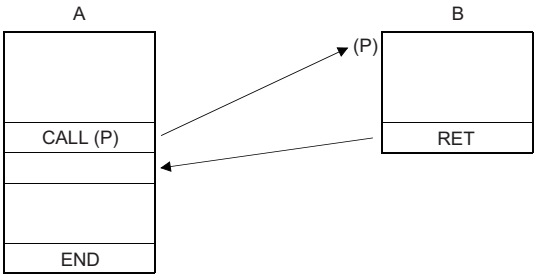
FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
RET	Every scan

Processing details

- This instruction indicates the end of a subroutine program.
- When the instruction is executed, the program returns to the next step where the CALL(P), FCALL(P), ECALL(P), EFCALL(P), or XCALL instruction that called the subroutine program is executed.



A: Main routine program

B: Subroutine program

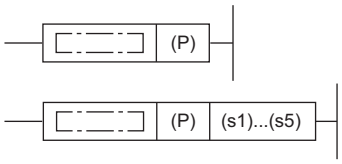
Operation error

Error code (SD0)	Description
3341H	After execution of the CALL(P), FCALL(P), ECALL(P), EFCALL(P), or XCALL instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3342H	The RET instruction is executed before the CALL(P), FCALL(P), ECALL(P), EFCALL(P), or XCALL instruction.

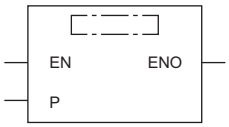
Calling a subroutine program and turning the output off

FCALL(P)

These instructions perform non-execution processing of the subroutine program of the specified pointer.

Ladder	ST
	Not supported

FBD/LD



■Execution condition

Instruction	Execution condition
FCALL	
FCALLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(P)	Start pointer number of subroutine program	—	Device name	POINTER
(s1)···(s5)*1	Device number to be passed to the subroutine program as an argument	-2147483648 to 2147483647	Bit/16-bit signed binary/32-bit signed binary	ANY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Cannot be specified in FBD/LD.

■Applicable devices

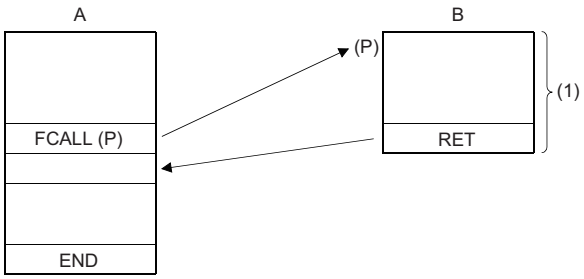
Operand	Bit		Word			Double word		Indirect specification	Constant			Others (P)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(P)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)···(s5)	○*1	○	○	○	○	○	○	○	○	—	—	—

*1 Devices other than F can be used.

Processing details

- When the FCALL (P) instruction is executed, this instruction executes non-execution processing*2 of the subroutine program of the pointer (P). The FCALL(P) instruction can disable the execution of the subroutine program specified by a pointer in the same program file or the subroutine program specified by a global pointer.

*2 Non-execution processing is the same as the processing performed by each coil instruction with the condition contact set to off.



A: Main routine program
B: Subroutine program
(1) Non-execution processing is performed when the command of the FCALL(P) instruction changes from on to off.

- The operation results of individual coil instructions after the end of non-execution processing are as follows regardless of on/off of the condition contact.

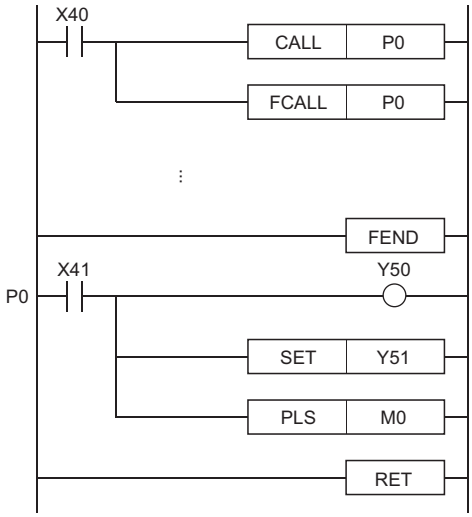
Device used for operation	Operation result (device status)
High-speed timer, low-speed timer	Set to 0
High-speed retentive timer, low-speed retentive timer, counter	Maintains the current status.
Device in OUT instruction	Forcibly turned off.
Device in the SET, RST, SFT(P), or basic/application instruction	Maintains the current status.
PLS or pulse instruction (□P)	Performs the same processing as when the condition contact is off.

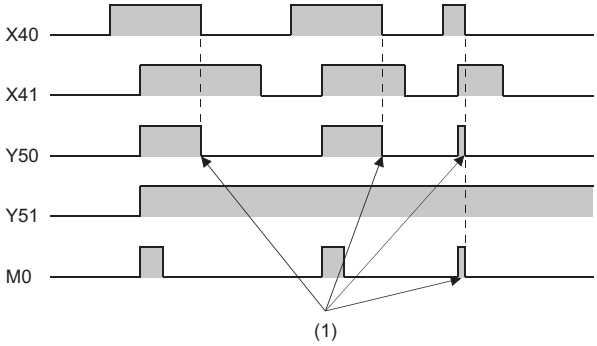
- The FCALL(P) instruction is used in combination with the CALL(P) instruction. If the FCALL(P) instruction is not used in combination with the CALL(P) instruction, non-execution processing of the subroutine program is not performed even if the execution command is turned off, and therefore the output status of each coil instruction is retained.

• When the execution command is turned off, non-execution processing of the subroutine program is performed, enabling the OUT instruction and PLS instruction (including pulse conversion instructions) to be forcibly turned off.

Ex.

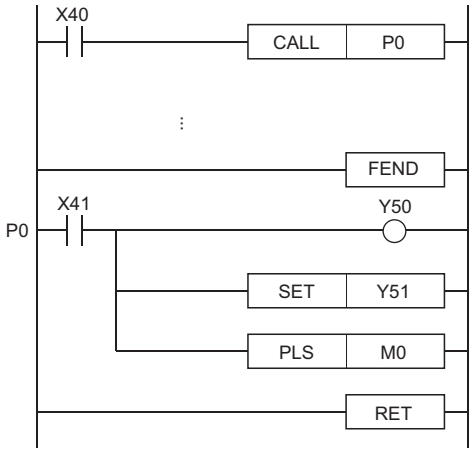
When the FCALL(P) instruction is used

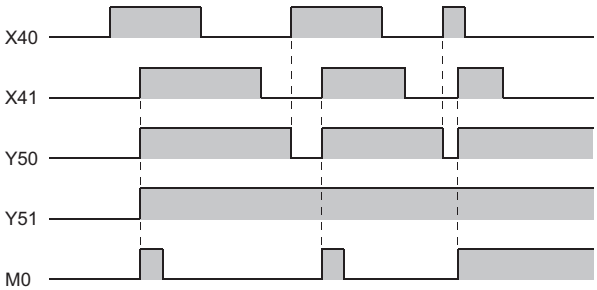




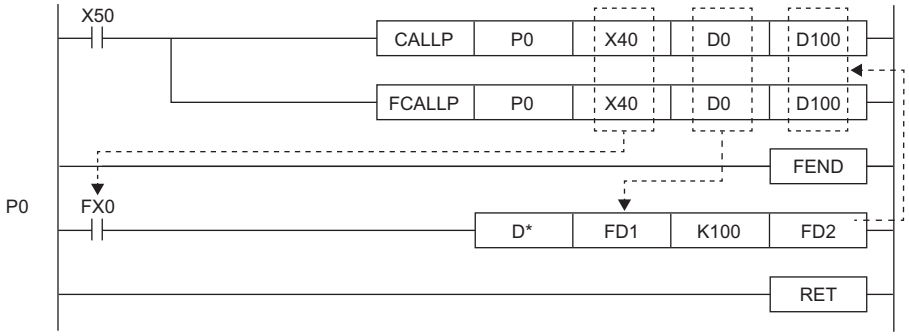
(1) Forced off by the FCALL instruction

When the FCALL(P) instruction is not used





• When the subroutine program uses function devices (FX, FY, FD), specify the devices corresponding to the function devices in (s1) to (s5). The following figure the contents of the devices specified by (s1) to (s5).



• Before execution of the subroutine program, bit data is transferred to FX and word data is transferred to FD.
• After execution of the subroutine program, the contents of FY and FD are transferred to the corresponding devices.

• The data transfer size for FX, FY, and FD differs depending on the data type specified in (s1) to (s5).

■When a device is specified

Function devices FX and FY are transferred in units of bits. Function device FD is transferred in units of 4 words. The size of data to be processed varies depending on the type of the device specified by an argument.

The device specified as a function device should be secured for the data size. An error occurs if an appropriate data area cannot be secured.

The following table lists the data sizes of each function device.

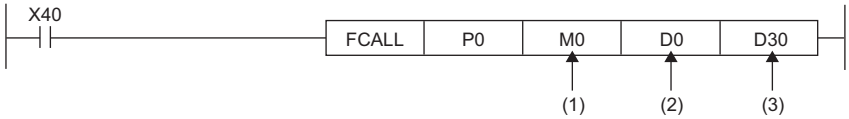
Function device	Device to be used	Sizes of data when transferred to function devices from (s1) to (s5)	Sizes of data when transferred from function devices to (s1) to (s5)
FX, FY	Bit device	1 bit	1 bit
	Bit-specified word device	1 bit	1 bit
FD	Digit-specified bit device*1	4 words	Digit-specified size or 4 words*2
	Word device	4 words	4 words

*1 It can be specified even if the device number specified by (s1) to (s5) is not a multiple of 16 in bit device digit specification mode.

*2 If the device number is a multiple of 16 and the digit specification is K4 or K8, size of 4 words will be transferred.

Ex.

The following operation occurs when a device is specified.



(1) M0 is transferred to FX0.

(2) D0 to D3 are transferred to FD1.

(3) D30 to D33 are transferred to FD2.

■When a label is specified

When data is transferred to function devices from (s1) to (s5), function devices FX and FY are transferred in units of bits. Function device FD is transferred in units of 4 words.

When data is transferred from function devices to (s1) to (s5), function devices FX and FY are transferred in units of bits. The unit of transfer for function device FD is the label-assigned or digit specification size unit.

The following table lists the data sizes of each function device.

Function device	Label-assigned size	Sizes of data when transferred to function devices from (s1) to (s5)	Sizes of data when transferred from function devices to (s1) to (s5)
FX, FY	1 bit	1 bit	1 bit
FD	2 bits or more	4 words*1	Label-assigned size*2*3/digit specification size*4

*1 If the assigned size of the specified label/structure is less than 4 words, 0 is transferred for the area that is less than 4 words.

*2 If the assigned size of the specified label/structure is 4 words or more, 4 words are transferred.

*3 When the following labels are specified, the size to be sent differs. If a label other than the following is specified, the data is transferred with the label-assignment size.

(1) When an array label (including when an element is specified) or a structure type label is specified, the entire size of the specified label will be transferred.

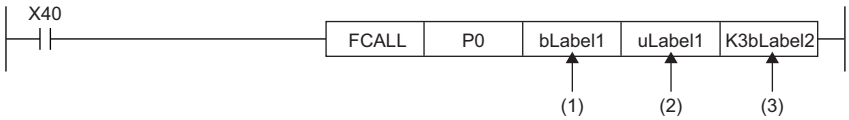
(2) When a member of the structure type label is specified, the size of the specified member will be transferred.

*4 When the digit specification is K1 to K3, 1 word will be transferred; when it is K5 to K7, 2 words will be transferred; if it is K4 or K8, the data will be transferred with the label-assignment size.

Ex.

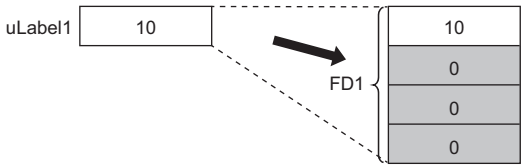
The following table lists the label names and label types when data is transferred to function devices from (s1) to (s5).

Label name	Label type
bLabel1	Bit type label
uLabel1	Word type label
bLabel2	Array bit type label

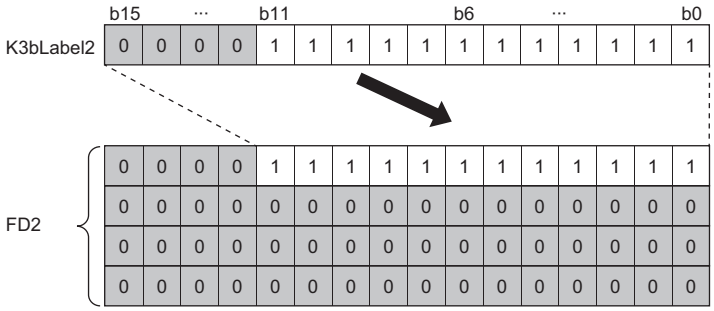


(1) bLabel1 (1 bit) is transferred to FX0.

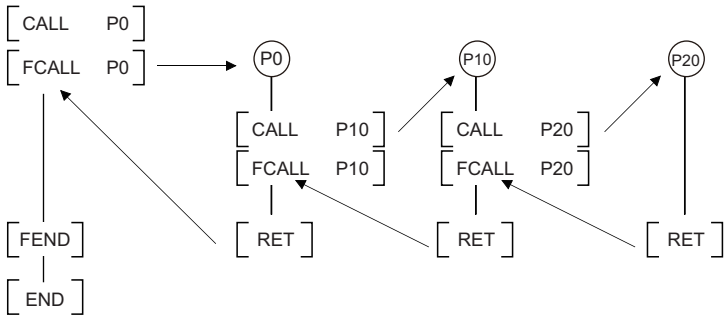
(2) uLabel1 (1 word) is transferred to FD1. Since the assigned size is less than 4 words, 0 is transferred for the area that is less than 4 words.



(3) 1 word of K3bLabel2 is transferred to FD2. Since the assigned size is less than 4 words, 0 is transferred for the area that is less than 4 words.



- The FCALL(P) instruction can use (s1) to (s5).
- Up to 16 FCALL(P) instructions can be nested. Note that the 16-level nesting is the total of the CALL(P), FCALL(P), ECALL(P), EFCALL(P), and XCALL instructions.



Precautions

- An FBD/LD program cannot be created as a subroutine program.
- FBD/LD does not support the execution of a subroutine program with an argument.

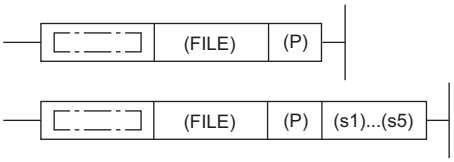
Operation error

Error code (SD0)	Description
2820H	The device specified in an argument from (s1) to (s5) cannot be secured for the data size.
3330H	The 17th level of the nesting is executed.
3340H	The subroutine program corresponding to the pointer specified by the FCALL(P) instruction does not exist. Alternatively, the pointer specified by (P) is index-modified, and the pointer that exists in FOR to NEXT is specified after the index modification is resolved.
3341H	After execution of the FCALL(P) instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3342H	The RET instruction is executed before the FCALL(P) instruction.

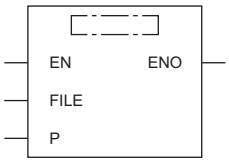
Calling a subroutine program in the specified program file

ECALL(P)

These instructions execute the subroutine program corresponding the specified pointer of the specified program file name.

Ladder	ST
 FILE: File name	Not supported

FBD/LD



FILE: File name

■Execution condition

Instruction	Execution condition
ECALL	
ECALLP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(File name)	Program file name to be called	—	Unicode string	ANYSTRING_DOUBLE
(P)	Start pointer number of subroutine program	—	Device name	POINTER
(s1)···(s5)*1	Device number to be passed to the subroutine program as an argument	-2147483648 to 2147483647	Bit/16-bit signed binary/32-bit signed binary	ANY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Cannot be specified in FBD/LD.

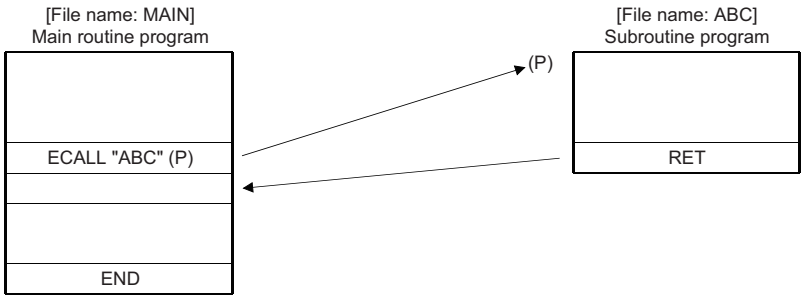
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (P)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(File name)	—	—	○	—	—	—	—	○	—	—	○	—
(P)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)···(s5)	○*1	○	○	○	○	○	○	○	○	—	—	—

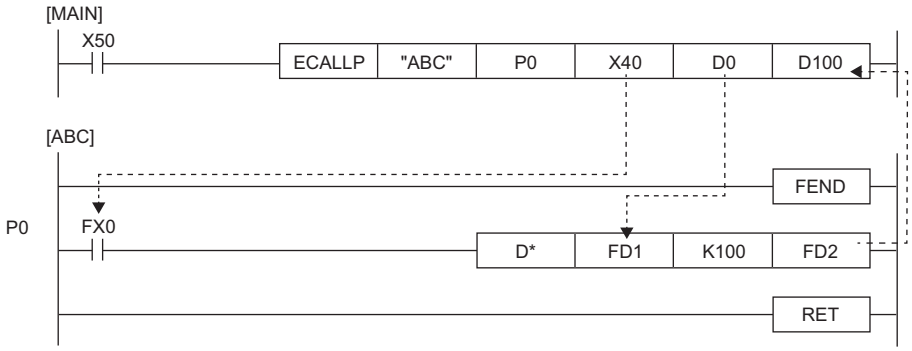
*1 Any value other than F can be used.

Processing details

- When the ECALL(P) instruction is executed, these instructions execute the subroutine program corresponding to the pointer specified by (P) of the specified program file name. The ECALL(P) instructions can also call the subroutine program using the local pointer of another program file.



- Only a program file stored in program memory can be specified for the file name.
- Extension ".PRG" need not be specified for the file name. (Only .PRG files can be processed by these instructions.)
- When the subroutine program uses function devices (FX, FY, FD), specify the devices corresponding to the function devices in (s1) to (s5). The following figure the contents of the devices specified by (s1) to (s5).



- Before execution of the subroutine program, bit data is transferred to FX and word data is transferred to FD.
- After execution of the subroutine program, the contents of FY and FD are transferred to the corresponding devices.

- The data transfer size for FX, FY, and FD differs depending on the data type specified in (s1) to (s5).

■When a device is specified

Function devices FX and FY are transferred in units of bits. Function device FD is transferred in units of 4 words. The size of data to be processed varies depending on the type of the device specified by an argument.

The device specified as a function device should be secured for the data size. An error occurs if an appropriate data area cannot be secured.

The following table lists the data sizes of each function device.

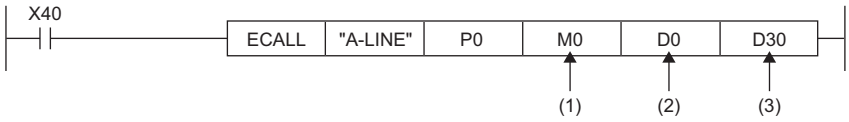
Function device	Device to be used	Sizes of data when transferred to function devices from (s1) to (s5)	Sizes of data when transferred from function devices to (s1) to (s5)
FX, FY	Bit device	1 bit	1 bit
	Bit-specified word device	1 bit	1 bit
FD	Digit-specified bit device*1	4 words	Digit-specified size or 4 words*2
	Word device	4 words	4 words

*1 It can be specified even if the device number specified by (s1) to (s5) is not a multiple of 16 in bit device digit specification mode.

*2 If the device number is a multiple of 16 and the digit specification is K4 or K8, size of 4 words will be transferred.

Ex.

The following operation occurs when a device is specified.



(1) M0 is transferred to FX0.

(2) D0 to D3 are transferred to FD1.

(3) D30 to D33 are transferred to FD2.

■When a label is specified

When data is transferred to function devices from (s1) to (s5), function devices FX and FY are transferred in units of bits. Function device FD is transferred in units of 4 words.

When data is transferred from function devices to (s1) to (s5), function devices FX and FY are transferred in units of bits. The unit of transfer for function device FD is the label-assigned or digit specification size unit.

The following table lists the data sizes of each function device.

Function device	Label-assigned size	Sizes of data when transferred to function devices from (s1) to (s5)	Sizes of data when transferred from function devices to (s1) to (s5)
FX, FY	1 bit	1 bit	1 bit
FD	2 bits or more	4 words*1	Label-assigned size*2*3/digit specification size*4

*1 If the assigned size of the specified label/structure is less than 4 words, 0 is transferred for the area that is less than 4 words.

*2 If the assigned size of the specified label/structure is 4 words or more, 4 words are transferred.

*3 When the following labels are specified, the size to be sent differs. If a label other than the following is specified, the data is transferred with the label-assignment size.

(1) When an array label (including when an element is specified) or a structure type label is specified, the entire size of the specified label will be transferred.

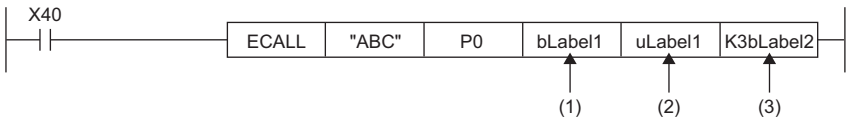
(2) When a member of the structure type label is specified, the size of the specified member will be transferred.

*4 When the digit specification is K1 to K3, 1 word will be transferred; when it is K5 to K7, 2 words will be transferred; if it is K4 or K8, the data will be transferred with the label-assignment size.

Ex.

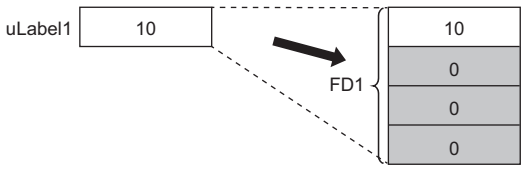
The following table lists the label names and label types when data is transferred to function devices from (s1) to (s5).

Label name	Label type
bLabel1	Bit type label
uLabel1	Word type label
bLabel2	Array bit type label

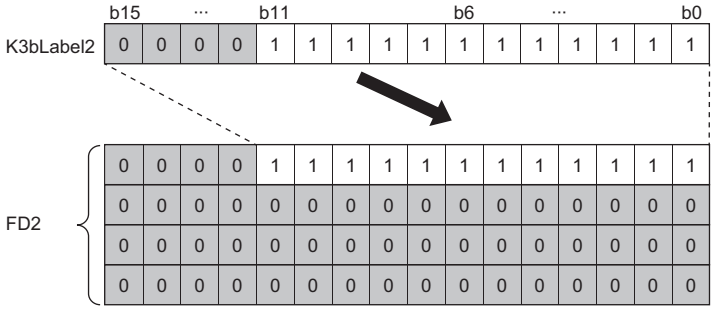


(1) bLabel1 (1 bit) is transferred to FX0.

(2) uLabel1 (1 word) is transferred to FD1. Since the assigned size is less than 4 words, 0 is transferred for the area that is less than 4 words.



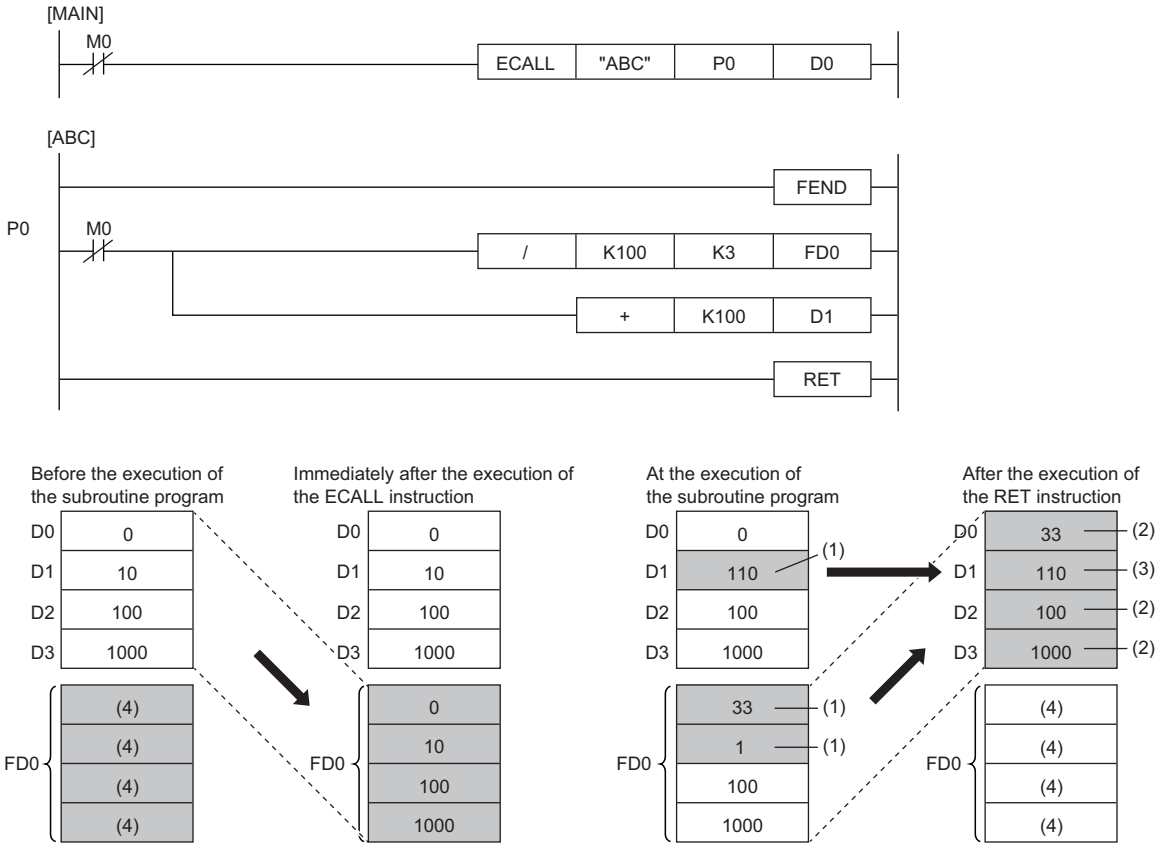
(3) 1 word of K3bLabel2 is transferred to FD2. Since the assigned size is less than 4 words, 0 is transferred for the area that is less than 4 words.



- The ECALL(P) instruction can use (s1) to (s5).
- Any device used in the argument of the ECALL(P) instruction must not be used in the subroutine program. Otherwise, normal operation cannot be performed.

Ex.

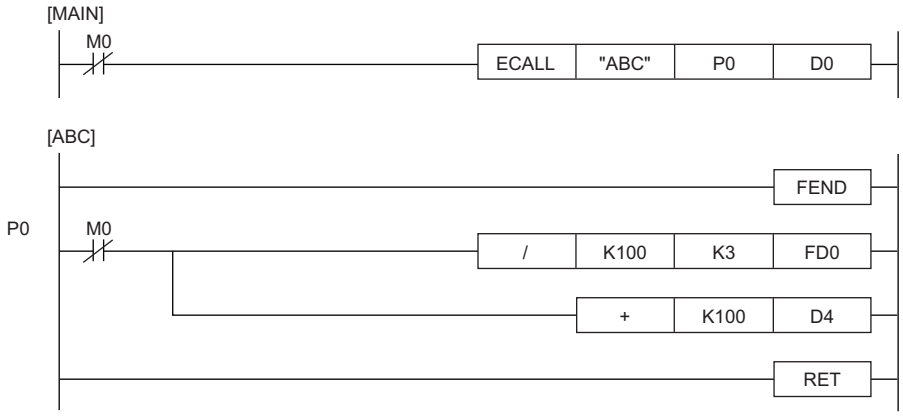
Incorrect operation: While D0 is specified for FD0 in the subroutine program, D1 is used in the subroutine program.

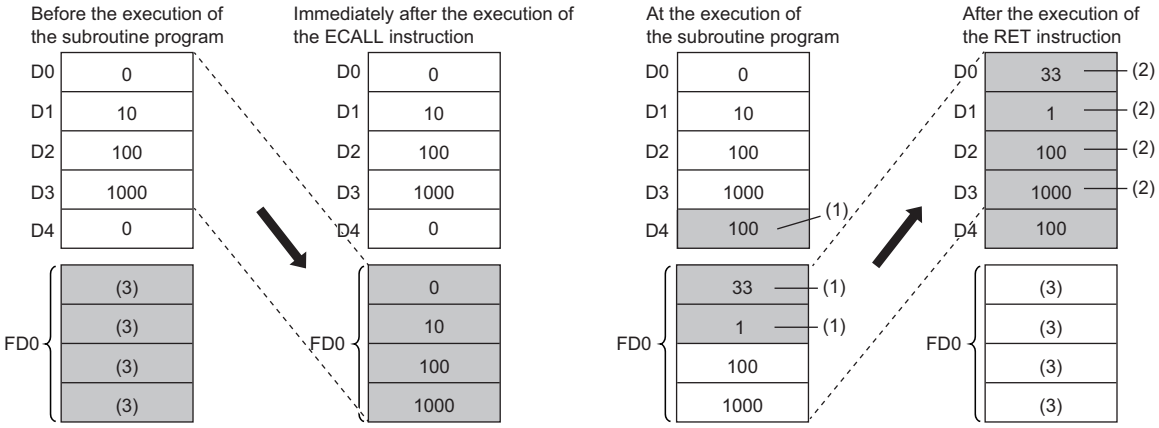


(1) The operation result of the subroutine program is stored.
(2) These values are replaced with the function device values.
(3) The value of D1 is not replaced with the function device value.
(4) Undefined values are stored.

Ex.

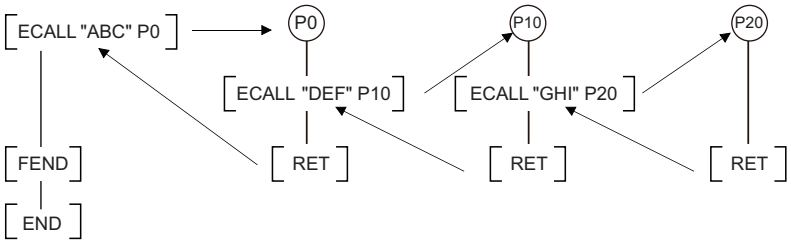
Correct operation: While D0 is specified for FD0 in the subroutine program, D4 is used in the subroutine program.





(1) The operation result of the subroutine program is stored.
(2) These values are replaced with the function device values.
(3) Undefined values are stored.

- The device numbers specified by the ECALL(P) instruction arguments must not be overlapping. If they are overlapping, normal operation cannot be performed.
- Up to 16 ECALL(P) instructions can be nested. Note that the 16-level nesting is the total of the CALL(P), FCALL(P), ECALL(P), EFCALL(P), and XCALL instructions.



- Devices which are turned on in the subroutine program are retained even while the subroutine program is not executed. Devices which are turned on during execution of the subroutine program can be turned off by the EFCALL(P) instruction.

Precautions

- An FBD/LD program cannot be created as a subroutine program.
- FBD/LD does not support the execution of a subroutine program with an argument.
- When the pointer specified by (P) is index-modified, an error (error code: 3340H) occurs if a pointer that exists in FOR to NEXT is specified after the index modification is resolved. If a pointer that exists in FOR to NEXT is specified directly without index modification, a conversion error occurs.

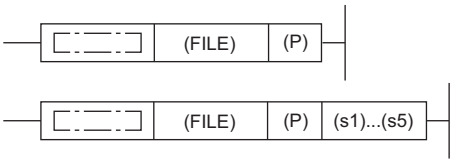
Operation error

Error code (SD0)	Description
2820H	The device specified in an argument from (s1) to (s5) cannot be secured for the data size.
2840H	The file specified by (file name) does not exist.
2841H	The file specified by (file name) cannot be executed.
2842H	The type of the program with the file name specified by (file name) is not supported.
3330H	The 17th level of the nesting is executed.
3340H	The subroutine program corresponding to the pointer specified by (P) does not exist. Alternatively, the pointer specified by (P) is index-modified, and the pointer that exists in FOR to NEXT is specified after the index modification is resolved.
3341H	After execution of the ECALL(P) instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3342H	The RET instruction is executed before the ECALL(P) instruction.

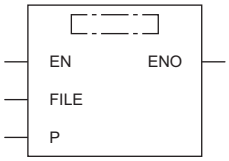
Calling a subroutine program in the specified program file and turning the output off

EFCALL(P)

These instructions perform non-execution processing of the subroutine program corresponding the specified pointer of the specified program file name.

Ladder	ST
 FILE: File name	Not supported

FBD/LD



FILE: File name

■Execution condition

Instruction	Execution condition
EFCALL	
EFCALLP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(File name)	Program file name subject to non-execution processing	—	Unicode string	ANYSTRING_DOUBLE
(P)	Start pointer number of subroutine program	—	Device name	POINTER
(s1) to (s5)*1	Device number to be passed to the subroutine program as an argument	-2147483648 to 2147483647	Bit/16-bit signed binary/32-bit signed binary	ANY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Cannot be specified in FBD/LD.

■Applicable devices

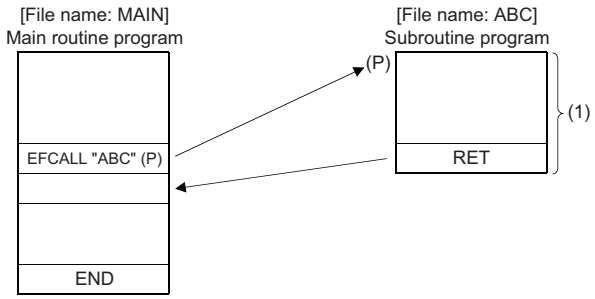
Operand	Bit		Word			Double word		Indirect specification	Constant			Others (P)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(File name)	—	—	○	—	—	—	—	○	—	—	○	—
(P)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)···(s5)	○*1	○	○	○	○	○	○	○	○	—	—	—

*1 Any value other than F can be used.

Processing details

- When the EFCALL(P) instruction is executed, this instruction executes non-execution processing*2 of the subroutine program of the pointer specified by (P). The EFCALL(P) instructions can also call the subroutine program using the local pointer of another program file.

*2 Non-execution processing is the same as the processing performed by each coil instruction with the condition contact set to off.



(1) Non-execution processing is performed when the command of the EFCALL(P) instruction changes from on to off.

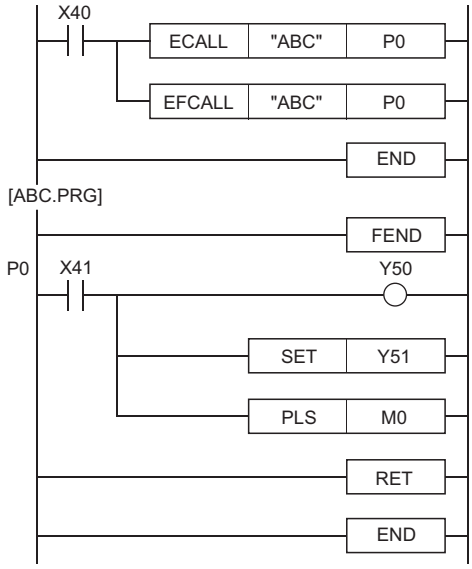
- The operation results of individual coil instructions after the end of non-execution processing are as follows regardless of on/off of the condition contact.

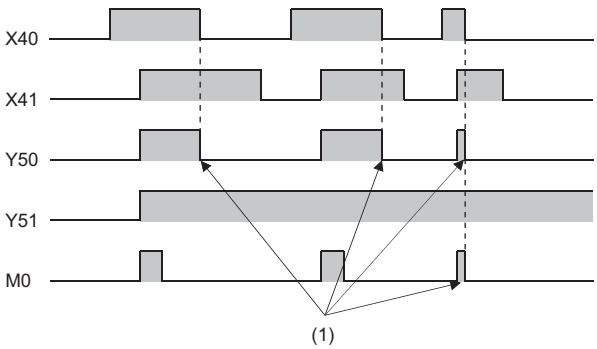
Device used for operation	Operation result (device status)
High-speed timer, low-speed timer	Set to 0
High-speed retentive timer, low-speed retentive timer, counter	Maintains the current status.
Device in OUT instruction	Forcibly turned off.
Device in the SET, RST, SFT(P), or basic/application instruction	Maintains the current status.
PLS or pulse instruction (□P)	Performs the same processing as when the condition contact is off.

- The EFCALL(P) instruction is used in combination with the ECALL(P) instruction. If the EFCALL(P) instruction is not used in combination with the ECALL(P) instruction, non-execution processing of the subroutine program is not performed even if the execution command is turned off, and therefore the output status of each coil instruction is retained.
- When the execution command is turned off, non-execution processing of the subroutine program is performed, enabling the OUT instruction and PLS instruction (including pulse conversion instructions) to be forcibly turned off.

Ex.

When the EFCALL (P) instruction is used

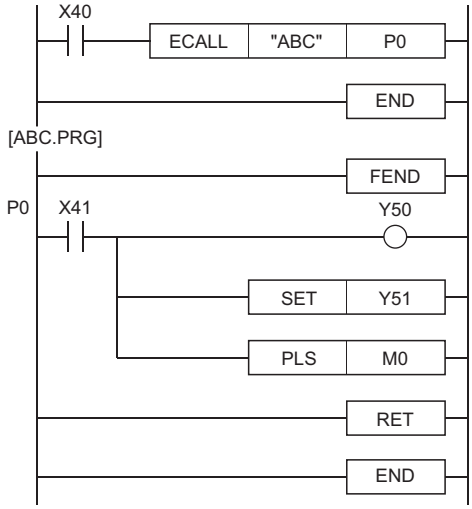


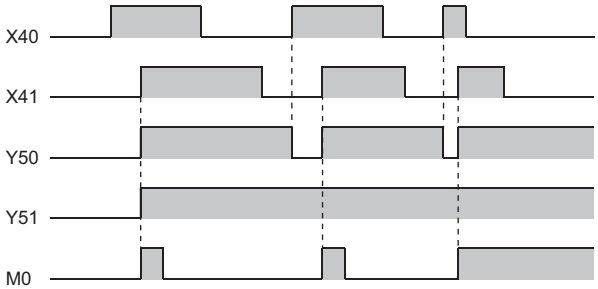


(1) Forced off by the EFCALL instruction

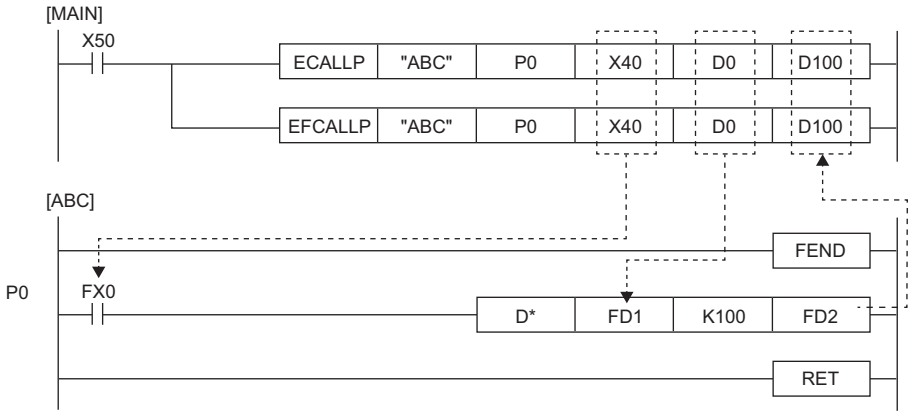
Ex.

When the EFCALL (P) instruction is not used





- Only a program file stored in program memory can be specified for the file name.
- Extension ".PRG" need not be specified for the file name. (Only .PRG files can be processed by these instructions.)
- When the subroutine program uses function devices (FX, FY, FD), specify the devices corresponding to the function devices in (s1) to (s5). The following figure the contents of the devices specified by (s1) to (s5).



- Before execution of the subroutine program, bit data is transferred to FX and word data is transferred to FD.
- After execution of the subroutine program, the contents of FY and FD are transferred to the corresponding devices.

- The data transfer size for FX, FY, and FD differs depending on the data type specified in (s1) to (s5).

■When a device is specified

Function devices FX and FY are transferred in units of bits. Function device FD is transferred in units of 4 words. The size of data to be processed varies depending on the type of the device specified by an argument.

The device specified as a function device should be secured for the data size. An error occurs if an appropriate data area cannot be secured.

The following table lists the data sizes of each function device.

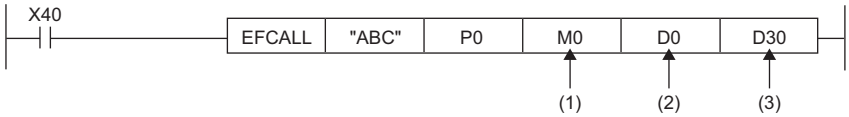
Function device	Device to be used	Sizes of data when transferred to function devices from (s1) to (s5)	Sizes of data when transferred from function devices to (s1) to (s5)
FX, FY	Bit device	1 bit	1 bit
	Bit-specified word device	1 bit	1 bit
FD	Digit-specified bit device*1	4 words	Digit-specified size or 4 words*2
	Word device	4 words	4 words

*1 It can be specified even if the device number specified by (s1) to (s5) is not a multiple of 16 in bit device digit specification mode.

*2 If the device number is a multiple of 16 and the digit specification is K4 or K8, size of 4 words will be transferred.

Ex.

The following operation occurs when a device is specified.



(1) M0 is transferred to FX0.
(2) D0 to D3 are transferred to FD1.
(3) D30 to D33 are transferred to FD2.

■When a label is specified

When data is transferred to function devices from (s1) to (s5), function devices FX and FY are transferred in units of bits. Function device FD is transferred in units of 4 words.

When data is transferred from function devices to (s1) to (s5), function devices FX and FY are transferred in units of bits. The unit of transfer for function device FD is the label-assigned or digit specification size unit.

The following table lists the data sizes of each function device.

Function device	Label-assigned size	Sizes of data when transferred to function devices from (s1) to (s5)	Sizes of data when transferred from function devices to (s1) to (s5)
FX, FY	1 bit	1 bit	1 bit
FD	2 bits or more	4 words*1	Label-assigned size*2*3/digit specification size*4

*1 If the assigned size of the specified label/structure is less than 4 words, 0 is transferred for the area that is less than 4 words.

*2 If the assigned size of the specified label/structure is 4 words or more, 4 words are transferred.

*3 When the following labels are specified, the size to be sent differs. If a label other than the following is specified, the data is transferred with the label-assignment size.

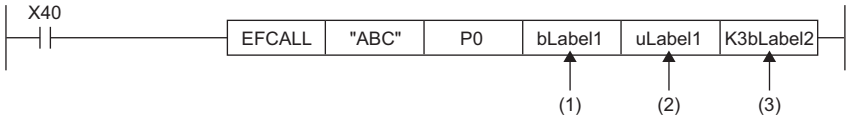
(1) When an array label (including when an element is specified) or a structure type label is specified, the entire size of the specified label will be transferred.
(2) When a member of the structure type label is specified, the size of the specified member will be transferred.

*4 When the digit specification is K1 to K3, 1 word will be transferred; when it is K5 to K7, 2 words will be transferred; if it is K4 or K8, the data will be transferred with the label-assignment size.

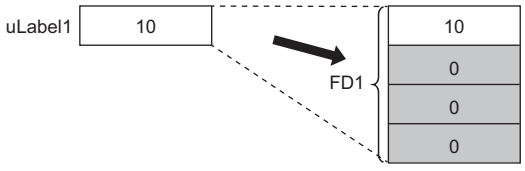
Ex.

The following table lists the label names and label types when data is transferred to function devices from (s1) to (s5).

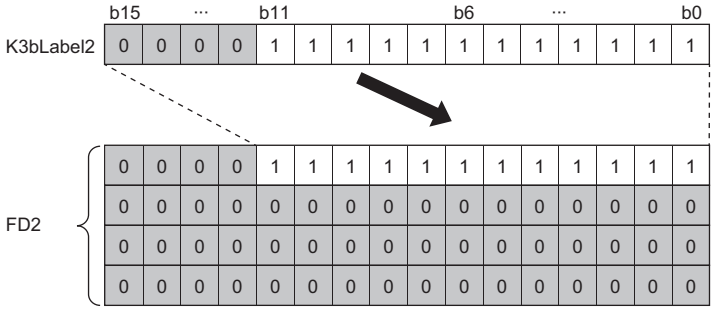
Label name	Label type
bLabel1	Bit type label
uLabel1	Word type label
bLabel2	Array bit type label



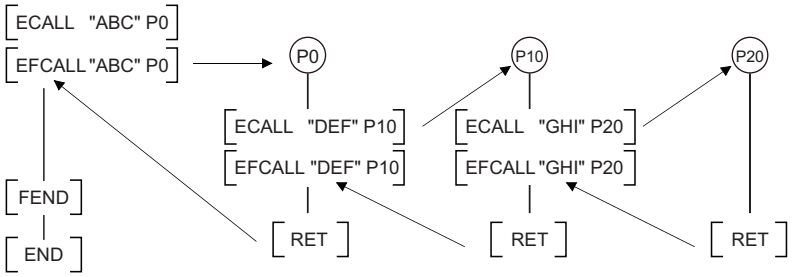
(1) bLabel1 (1 bit) is transferred to FX0.
(2) uLabel1 (1 word) is transferred to FD1. Since the assigned size is less than 4 words, 0 is transferred for the area that is less than 4 words.



(3) 1 word of K3bLabel2 is transferred to FD2. Since the assigned size is less than 4 words, 0 is transferred for the area that is less than 4 words.



- The EFCALL(P) instruction can use (s1) to (s5).
- The number of function devices used in the subroutine program must be identical to the number of arguments of the EFCALL(P) instruction. Also, the types of function devices and EFCALL(P) instruction arguments must be the same.
- Up to 16 EFCALL(P) instructions can be nested. Note that the 16-level nesting is the total of the CALL(P), FCALL(P), ECALL(P), EFCALL(P), and XCALL instructions.



Precautions

- An FBD/LD program cannot be created as a subroutine program.
- FBD/LD does not support the execution of a subroutine program with an argument.
- When the pointer specified by (P) is index-modified, an error (error code: 3340H) occurs if a pointer that exists in FOR to NEXT is specified after the index modification is resolved. If a pointer that exists in FOR to NEXT is specified directly without index modification, a conversion error occurs.

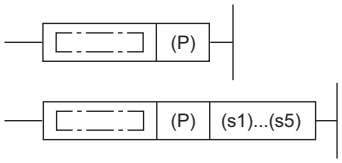
Operation error

Error code (SD0)	Description
2820H	The device specified in an argument from (s1) to (s5) cannot be secured for the data size.
2840H	The file specified by (file name) does not exist.
2841H	The file specified by (file name) cannot be executed.
3330H	The 17th level of the nesting is executed.
3340H	The subroutine program corresponding to the pointer specified by (P) does not exist. Alternatively, the pointer specified by (P) is index-modified, and the pointer that exists in FOR to NEXT is specified after the index modification is resolved.
3341H	After execution of the EFCALL(P) instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3342H	The RET instruction is executed before the EFCALL(P) instruction.

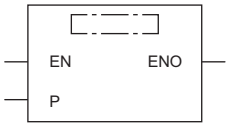
Calling a subroutine program

XCALL

This instruction performs execution or non-execution processing of a subroutine program. When the condition is satisfied, the instruction triggers CALL for the subroutine. When the condition turns off, the instruction triggers FCALL for the subroutine.

Ladder	ST
	Not supported

FBD/LD



■Execution condition

Instruction	Execution condition
XCALL	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(P)	Start pointer number of subroutine program	—	Device name	POINTER
(s1)···(s5)*1	Device number to be passed to the subroutine program as an argument	-2147483648 to 2147483647	Bit/16-bit signed binary/32-bit signed binary	ANY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Cannot be specified in FBD/LD.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	(P)
(P)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)···(s5)	○*1	○	○	○	○	○	○	○	○	—	—	—

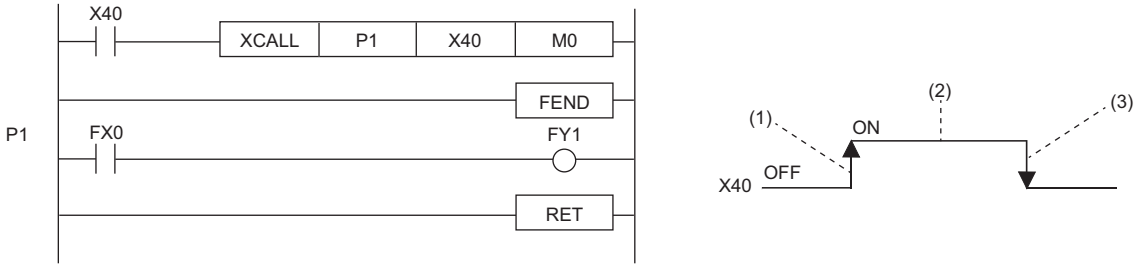
*1 Devices other than F can be used.

Processing details

- This instruction performs execution or non-execution processing of a subroutine program.
  - For execution of the subroutine program, it operates each coil instruction according to the on/off status of condition contacts.
  - For non-execution processing of the subroutine program, it operates each coil instruction in the same way as when condition contacts are off.
- The operation results of individual coil instructions after the end of non-execution processing are as follows regardless of on/off of the condition contact.

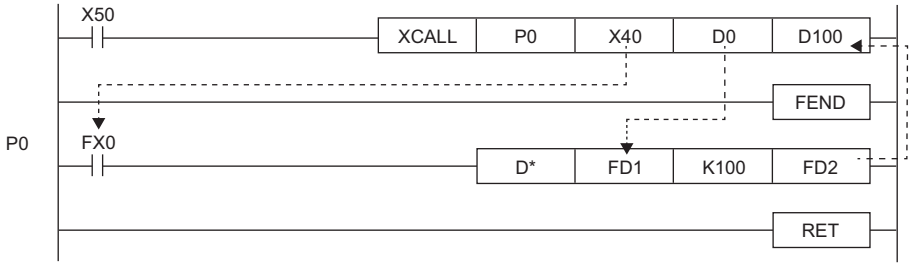
Device used for operation	Operation result (device status)
High-speed timer, low-speed timer	Set to 0
High-speed retentive timer, low-speed retentive timer, counter	Maintains the current status.
Device in OUT instruction	Forcibly turned off.
Device in the SET, RST, SFT(P), or basic/application instruction	Maintains the current status.
PLS or pulse instruction (□P)	Performs the same processing as when the condition contact is off.

- The following figure shows the operation of the XCALL instruction.



(1) On the rising edge of X40 (off→on): Executes the subroutine program specified by P1.
(2) During X40 is on: Executes the subroutine program specified by P1. ("During X40 is on" does not include the rising edge of X40.)
(3) On the falling edge of X40 (on→off): Performs non-execution processing of a subroutine program specified by P1.

- When the subroutine program uses function devices (FX, FY, FD), specify the devices corresponding to the function devices in (s1) to (s5). The following figure the contents of the devices specified by (s1) to (s5).



- Before execution of the subroutine program, bit data is transferred to FX and word data is transferred to FD.
- After execution of the subroutine program, the contents of FY and FD are transferred to the corresponding devices.

- The data transfer size for FX, FY, and FD differs depending on the data type specified in (s1) to (s5).

■When a device is specified

Function devices FX and FY are transferred in units of bits. Function device FD is transferred in units of 4 words. The size of data to be processed varies depending on the type of the device specified by an argument.

The device specified as a function device should be secured for the data size. An error occurs if an appropriate data area cannot be secured.

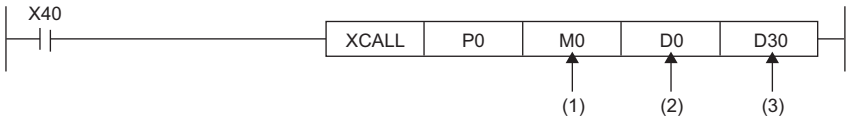
The following table lists the data sizes of each function device.

Function device	Device to be used	Sizes of data when transferred to function devices from (s1) to (s5)	Sizes of data when transferred from function devices to (s1) to (s5)
FX, FY	Bit device	1 bit	1 bit
	Bit-specified word device	1 bit	1 bit
FD	Digit-specified bit device*1	4 words	Digit-specified size or 4 words*2
	Word device	4 words	4 words

*1 It can be specified even if the device number specified by (s1) to (s5) is not a multiple of 16 in bit device digit specification mode.
*2 If the device number is a multiple of 16 and the digit specification is K4 or K8, size of 4 words will be transferred.

Ex.

The following operation occurs when a device is specified.



(1) M0 is transferred to FX0.
(2) D0 to D3 are transferred to FD1.
(3) D30 to D33 are transferred to FD2.

■When a label is specified

When data is transferred to function devices from (s1) to (s5), function devices FX and FY are transferred in units of bits. Function device FD is transferred in units of 4 words.

When data is transferred from function devices to (s1) to (s5), function devices FX and FY are transferred in units of bits. The unit of transfer for function device FD is the label-assigned or digit specification size unit.

The following table lists the data sizes of each function device.

Function device	Label-assigned size	Sizes of data when transferred to function devices from (s1) to (s5)	Sizes of data when transferred from function devices to (s1) to (s5)
FX, FY	1 bit	1 bit	1 bit
FD	2 bits or more	4 words*1	Label-assigned size*2*3/digit specification size*4

*1 If the assigned size of the specified label/structure is less than 4 words, 0 is transferred for the area that is less than 4 words.

*2 If the assigned size of the specified label/structure is 4 words or more, 4 words are transferred.

*3 When the following labels are specified, the size to be sent differs. If a label other than the following is specified, the data is transferred with the label-assignment size.

(1) When an array label (including when an element is specified) or a structure type label is specified, the entire size of the specified label will be transferred.

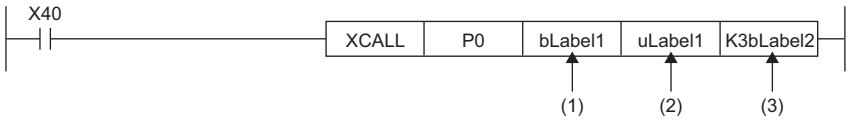
(2) When a member of the structure type label is specified, the size of the specified member will be transferred.

*4 When the digit specification is K1 to K3, 1 word will be transferred; when it is K5 to K7, 2 words will be transferred; if it is K4 or K8, the data will be transferred with the label-assignment size.

Ex.

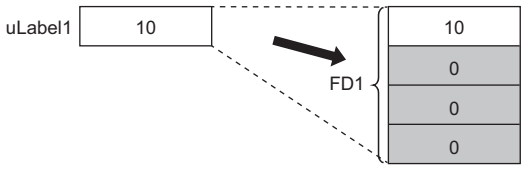
The following table lists the label names and label types when data is transferred to function devices from (s1) to (s5).

Label name	Label type
bLabel1	Bit type label
uLabel1	Word type label
bLabel2	Array bit type label

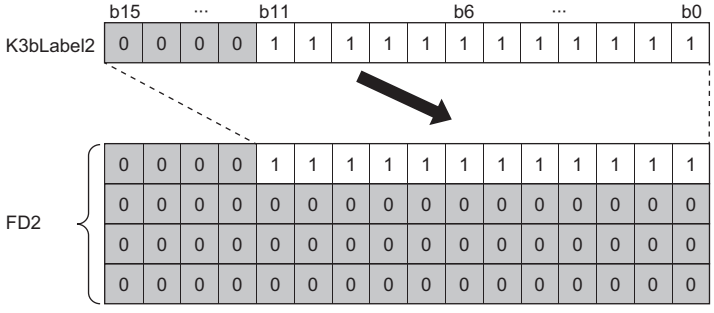


(1) bLabel1 (1 bit) is transferred to FX0.

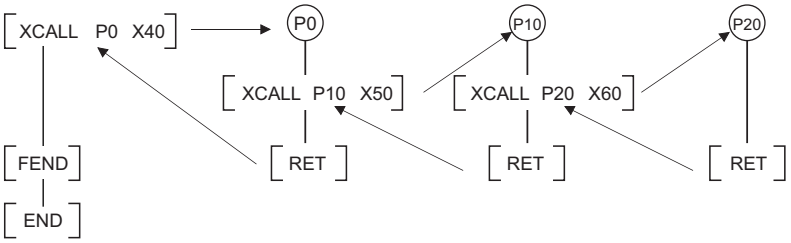
(2) uLabel1 (1 word) is transferred to FD1. Since the assigned size is less than 4 words, 0 is transferred for the area that is less than 4 words.



(3) 1 word of K3bLabel2 is transferred to FD2. Since the assigned size is less than 4 words, 0 is transferred for the area that is less than 4 words.



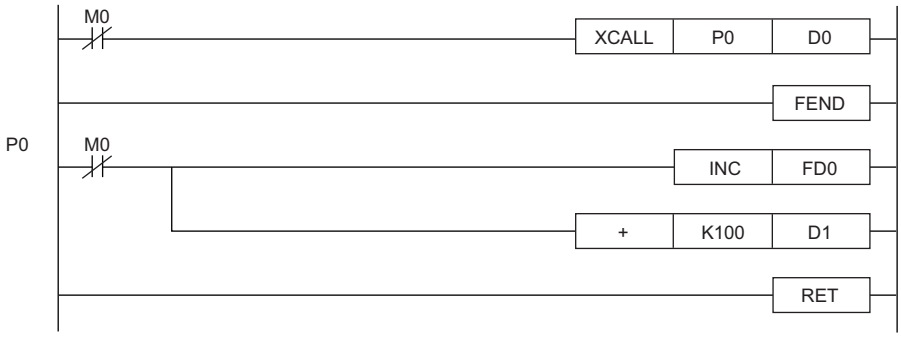
- The XCALL instruction can use (s1) to (s5).
- The number of function devices used in the subroutine program must be identical to the number of arguments of the XCALL instruction. Also, the types of function devices and XCALL instruction arguments must be the same.
- The device numbers specified by the XCALL instruction arguments must not be overlapping. If they are overlapping, normal operation cannot be performed.
- Up to 16 XCALL instructions can be nested. Note that the 16-level nesting is the total of the CALL(P), FCALL(P), ECALL(P), EFCALL(P), and XCALL instructions.

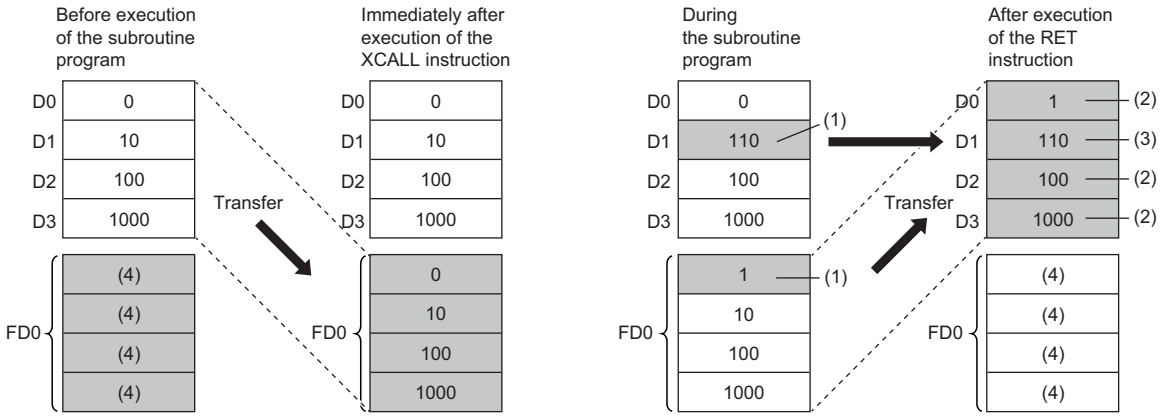


- Any device used in the argument of the XCALL instruction must not be used in the subroutine program. Otherwise, normal operation cannot be performed.

Ex.

Incorrect operation: While D0 is specified for FD0 in the subroutine program, D1 is used in the subroutine program.

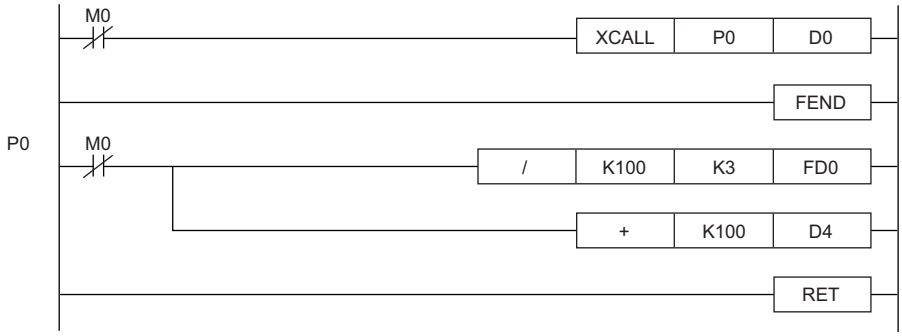


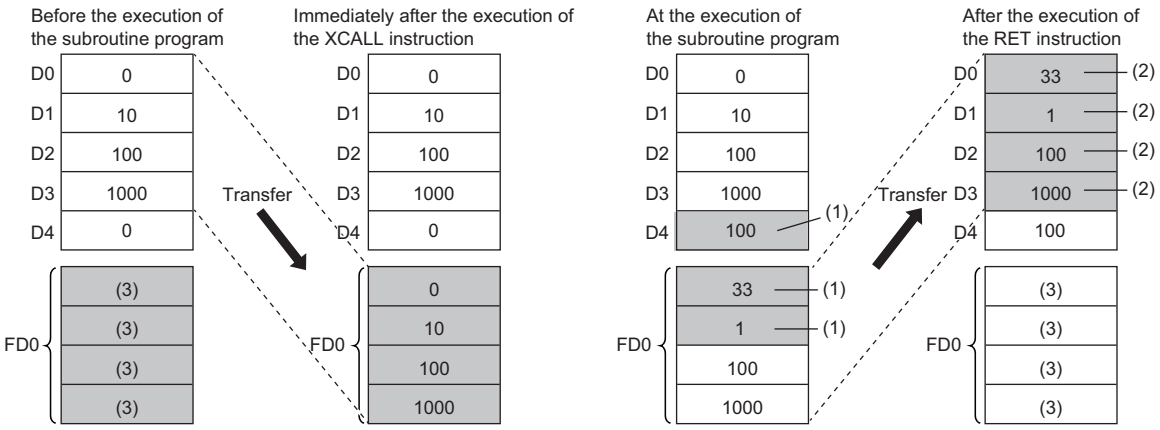


(1) The operation result of the subroutine program is stored.
(2) These values are replaced with the function device values.
(3) The value of D1 is not replaced with the function device value.
(4) Undefined values are stored.

Ex.

Correct operation: While D0 is specified for FD0 in the subroutine program, D4 is used in the subroutine program.





(1) The operation result of the subroutine program is stored.
(2) These values are replaced with the function device values.
(3) Undefined values are stored.

Precautions

- An FBD/LD program cannot be created as a subroutine program.
- FBD/LD does not support the execution of a subroutine program with an argument.
- When the pointer specified by (P) is index-modified, an error (error code: 3340H) occurs if a pointer that exists in FOR to NEXT is specified after the index modification is resolved. If a pointer that exists in FOR to NEXT is specified directly without index modification, a conversion error occurs.

Operation error

Error code (SD0)	Description
2820H	The device specified in an argument from (s1) to (s5) cannot be secured for the data size.
3330H	The 17th level of the nesting is executed.
3340H	The subroutine program corresponding to the pointer specified by (P) does not exist. Alternatively, the pointer specified by (P) is index-modified, and the pointer that exists in FOR to NEXT is specified after the index modification is resolved.
3341H	After execution of the XCALL instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3342H	The RET instruction is executed before the XCALL instruction.

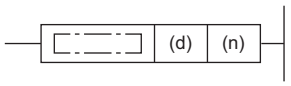
14 DATA PROCESSING

14.1 Rotation Instructions

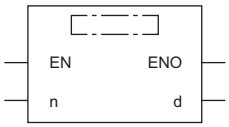
Rotating 16-bit binary data to the right

ROR(P), RCR(P)

- ROR(P): These instructions rotate the 16-bit binary data to the right by (n) bit(s), excluding the carry flag.
- RCR(P): These instructions rotate the 16-bit binary data to the right by (n) bit(s), including the carry flag.

Ladder	ST*1
	ENO:=RORP(EN,n,d); ENO:=RCR(EN,n,d); ENO:=RCRP(EN,n,d);

FBD/LD*1



*1 The ROR instruction does not support the ST and FBD/LD. Use the standard function, ROR.
☞ Page 1308 ROR(_E)

■Execution condition

Instruction	Execution condition
ROR RCR	
RORP RCRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Rotation target device	—	16-bit signed binary	ANY16
(n)	Number of bits to be rotated	0 to 15	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

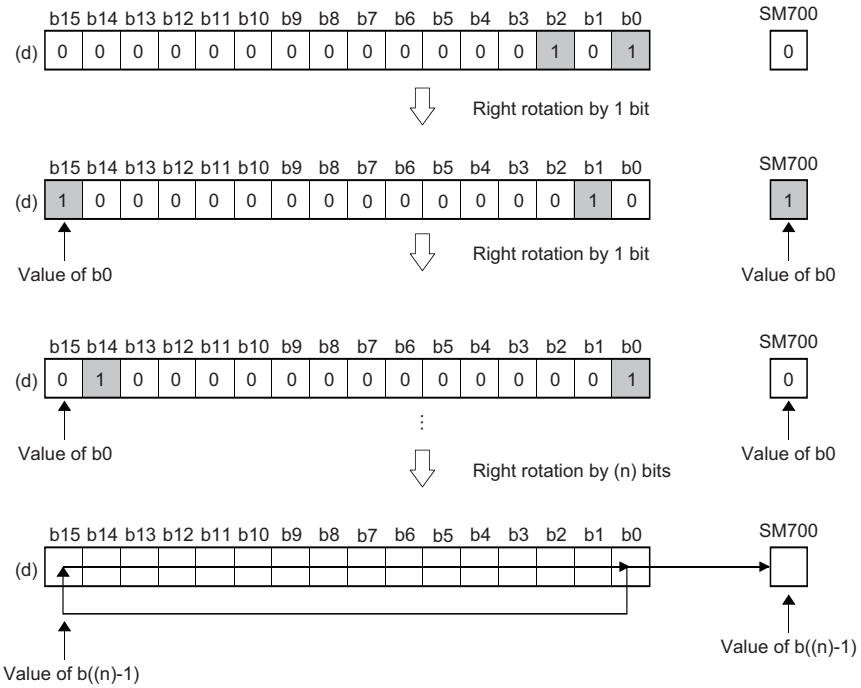
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

■ROR(P)

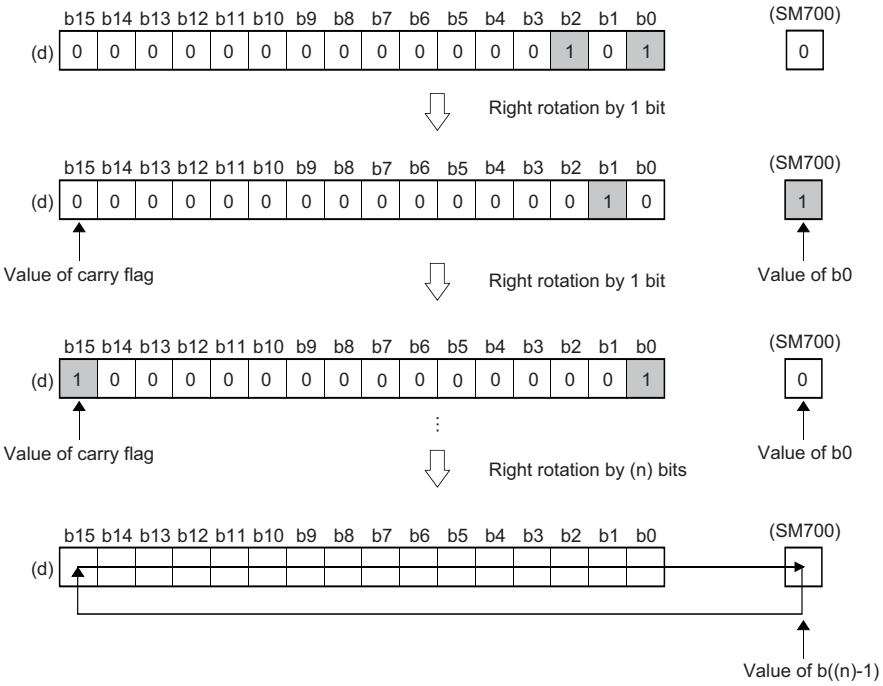
- These instructions rotate the 16-bit binary data in the device specified by (d) to the right by (n) bit(s), excluding the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the ROR(P) instruction.



- When (d) is a bit device, bits are rotated to the right within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 12, 3 bits are rotated because 15 divided by 12 equals 1 with a remainder of 3.
- Specify any value between 0 and 15 for (n). If a value 16 or bigger is specified, bits are rotated by the remainder value of n÷16. For example, when (n) is 18, 2 bits are rotated because 18 divided by 16 equals 1 with a remainder of 2.

■RCR(P)

- These instructions rotate the 16-bit binary data in the device specified by (d) to the right by (n) bit(s), including the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the RCR(P) instruction.



- When (d) is a bit device, bits are rotated to the right within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 12, 3 bits are rotated because 15 divided by 12 equals 1 with a remainder of 3.
- Specify any value between 0 and 15 for (n). If a value 16 or bigger is specified, bits are rotated by the remainder value of n÷16. For example, when (n) is 18, 2 bits are rotated because 18 divided by 16 equals 1 with a remainder of 2.

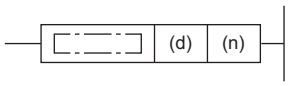
Operation error

There is no operation error.

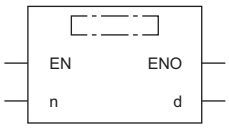
Rotating 16-bit binary data to the left

ROL(P), RCL(P)

- ROL(P): These instructions rotate the 16-bit binary data to the left by (n) bit(s), excluding the carry flag.
- RCL(P): These instructions rotate the 16-bit binary data to the left by (n) bit(s), including the carry flag.

Ladder	ST*1
	ENO:=ROLP(EN,n,d); ENO:=RCL(EN,n,d); ENO:=RCLP(EN,n,d);

FBD/LD*1



*1 The ROL instruction does not support the ST and FBD/LD. Use the standard function, ROL.
☞ Page 1306 ROL(_E)

■Execution condition

Instruction	Execution condition
ROL RCL	
ROLP RCLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Rotation target device	—	16-bit signed binary	ANY16
(n)	Number of bits to be rotated	0 to 15	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

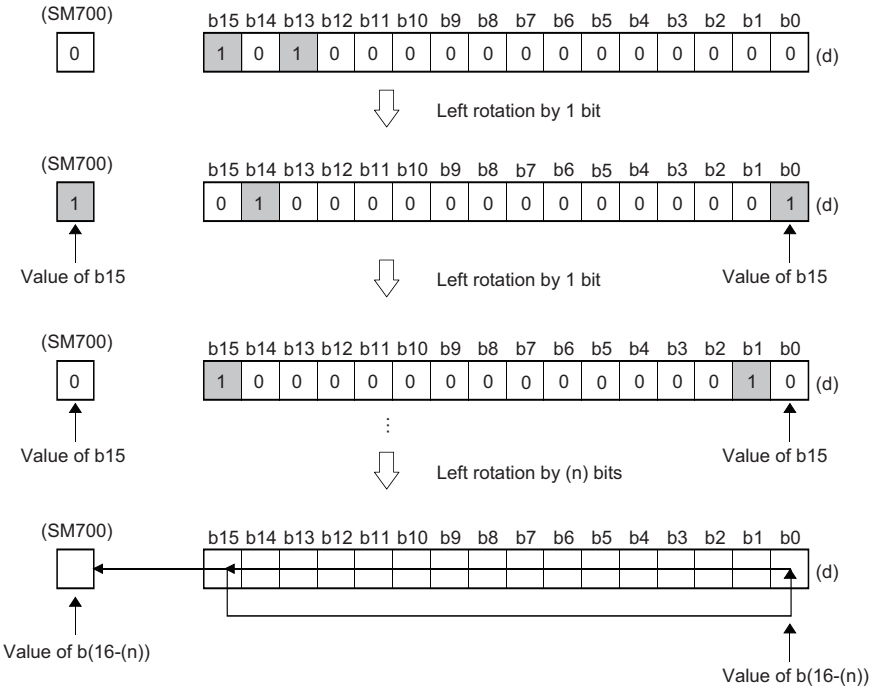
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

■ROL(P)

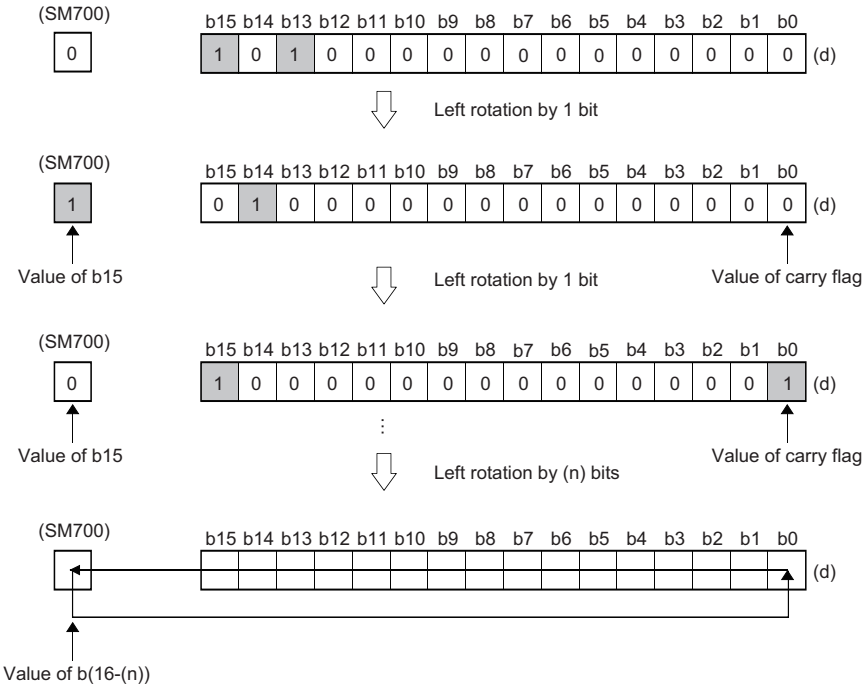
- These instructions rotate the 16-bit binary data in the device specified by (d) to the left by (n) bit(s), excluding the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the ROL(P) instruction.



- When (d) is a bit device, bits are rotated to the left within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 12, 3 bits are rotated because 15 divided by 12 equals 1 with a remainder of 3.
- Specify any value between 0 and 15 for (n). If a value 16 or bigger is specified, bits are rotated by the remainder value of n÷16. For example, when (n) is 18, 2 bits are rotated to the left because 18 divided by 16 equals 1 with a remainder of 2.

■RCL(P)

- These instructions rotate the 16-bit binary data in the device specified by (d) to the left by (n) bit(s), including the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the RCL(P) instruction.



- When (d) is a bit device, bits are rotated to the left within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 15 and the specified number of bits is 12, 3 bits are rotated because 15 divided by 12 equals 1 with a remainder of 3.
- Specify any value between 0 and 15 for (n). If a value 16 or bigger is specified, bits are rotated by the remainder value of n÷16. For example, when (n) is 18, 2 bits are rotated to the left because 18 divided by 16 equals 1 with a remainder of 2.

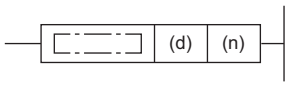
Operation error

There is no operation error.

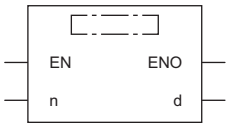
Rotating 32-bit binary data to the right

DROR(P), DRCR(P)

- DROR(P): These instructions rotate the 32-bit binary data to the right by (n) bit(s), excluding the carry flag.
- DRCR(P): These instructions rotate the 32-bit binary data to the right by (n) bit(s), including the carry flag.

Ladder	ST*1
	ENO:=DRORP(EN,n,d); ENO:=DRCR(EN,n,d); ENO:=DRCRP(EN,n,d);

FBD/LD*1



*1 The DROR instruction does not support the ST and FBD/LD. Use the standard function, ROR.
☞ Page 1308 ROR(_E)

■Execution condition

Instruction	Execution condition
DROR DRCR	
DRORP DRCRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device where the rotation target data is stored	—	32-bit signed binary	ANY32
(n)	Number of bits to be rotated	0 to 31	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

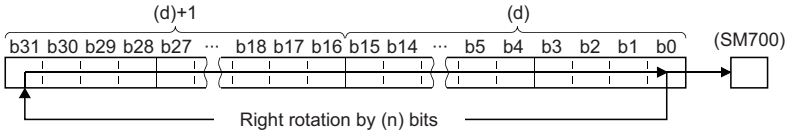
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	○	○	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

■DROR(P)

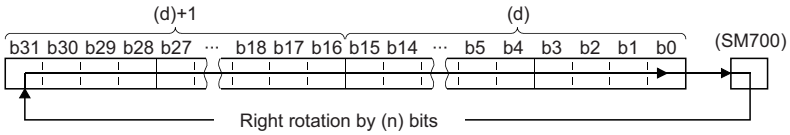
- These instructions rotate the 32-bit binary data in the device specified by (d) to the right by (n) bit(s), excluding the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the DROR(P) instruction.



- When (d) is a bit device, bits are rotated to the right within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 31 and the specified number of bits is 24, 7 bits are rotated because 31 divided by 24 equals 1 with a remainder of 7.
- Specify any value between 0 and 31 for (n). If a value 32 or bigger is specified, bits are rotated by the remainder value of n÷32. For example, when (n) is 34, 2 bits are rotated because 34 divided by 32 equals 1 with a remainder of 2.

■DRCR(P)

- These instructions rotate the 32-bit binary data in the device specified by (d) to the right by (n) bit(s), including the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the DRCR(P) instruction.



- When (d) is a bit device, bits are rotated to the right within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 31 and the specified number of bits is 24, 7 bits are rotated because 31 divided by 24 equals 1 with a remainder of 7.
- Specify any value between 0 and 31 for (n). If a value 32 or bigger is specified, bits are rotated by the remainder value of n÷32. For example, when (n) is 34, 2 bits are rotated because 34 divided by 32 equals 1 with a remainder of 2.

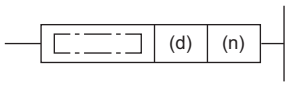
Operation error

There is no operation error.

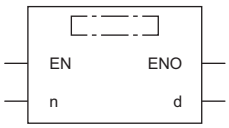
Rotating 32-bit binary data to the left

DROL(P), DRCL(P)

- DROL(P): These instructions rotate the 32-bit binary data to the left by (n) bit(s), excluding the carry flag.
- DRCL(P): These instructions rotate the 32-bit binary data to the left by (n) bit(s), including the carry flag.

Ladder	ST*1
	ENO:=DROLP(EN,n,d); ENO:=DRCL(EN,n,d); ENO:=DRCLP(EN,n,d);

FBD/LD*1



*1 The DROL instruction does not support the ST and FBD/LD. Use the standard function, ROL.
☞ Page 1306 ROL(_E)

■Execution condition

Instruction	Execution condition
DROL DRCL	
DROLP DRCLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device where the rotation target data is stored	—	32-bit signed binary	ANY32
(n)	Number of bits to be rotated	0 to 31	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

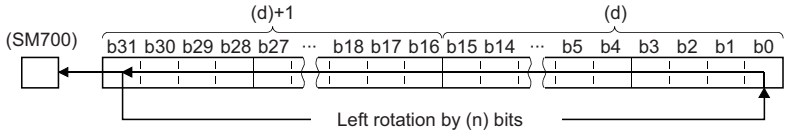
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	○	○	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

■DROL(P)

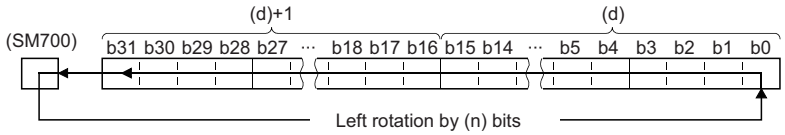
- These instructions rotate the 32-bit binary data in the device specified by (d) to the left by (n) bit(s), excluding the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the DROL(P) instruction.



- When (d) is a bit device, bits are rotated to the left within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 31 and the specified number of bits is 24, 7 bits are rotated because 31 divided by 24 equals 1 with a remainder of 7.
- Specify any value between 0 and 31 for (n). If a value 32 or bigger is specified, bits are rotated by the remainder value of n÷32. For example, when (n) is 34, 2 bits are rotated to the left because 34 divided by 32 equals 1 with a remainder of 2.

■DRCL(P)

- These instructions rotate the 32-bit binary data in the device specified by (d) to the left by (n) bit(s), including the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the DRCL(P) instruction.



- When (d) is a bit device, bits are rotated to the left within the device range specified by digit specification. The number of bits actually to be rotated is the remainder of (n)÷(specified number of bits). For example, when (n) is 31 and the specified number of bits is 24, 7 bits are rotated because 31 divided by 24 equals 1 with a remainder of 7.
- Specify any value between 0 and 31 for (n). If a value 32 or bigger is specified, bits are rotated by the remainder value of n÷32. For example, when (n) is 34, 2 bits are rotated to the left because 34 divided by 32 equals 1 with a remainder of 2.

Operation error

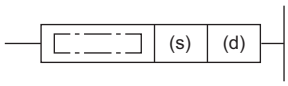
There is no operation error.

14.2 Data Table Operation Instructions

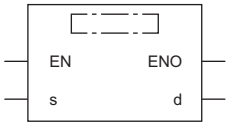
Reading the oldest data from the data table

FIFR(P)

These instructions store the data first stored in the table in the specified device.

Ladder	ST
	ENO:=FIFR(EN,s,d); ENO:=FIFRP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FIFR	
FIFRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device for storing the data read from the table	—	16-bit signed binary	ANY16
(d)	Start device of table	—	Word	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

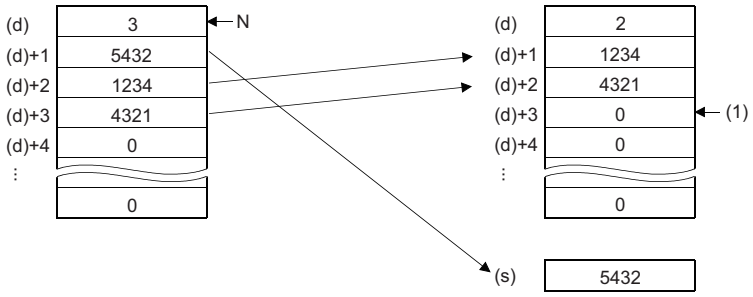
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions store the first-in data in the device specified by (d)+1 in the table specified by (d) in the device specified by (s). After execution of the FIFR(P) instruction, the data in the data table is moved forward one by one.



N: Number of data
(1) Filled with 0.

- An interlock mechanism should be used to prevent the FIFR(P) instruction from being executed when the value stored in the device specified by (d) is 0.
- A number from 0 to 65535 is stored in the number of data (d).

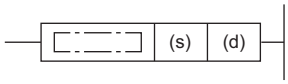
Operation error

Error code (SD0)	Description
3285H	The FIFR(P) instruction is executed when the value in the device specified by (d) is 0.

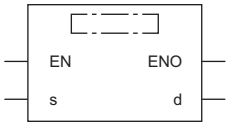
Reading the newest data from the data table

FPOP(P)

These instructions store the data last stored in the table in the specified device.

Ladder	ST
	ENO:=FPOP(EN,s,d); ENO:=FPOPP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FPOP	
FPOPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device for storing the data read from the table	—	16-bit signed binary	ANY16
(d)	Start device of table	—	Word	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

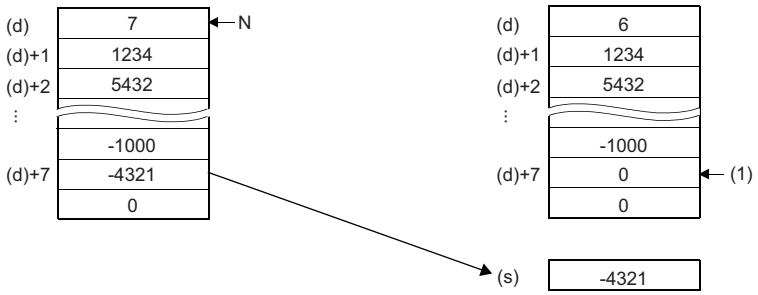
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions store the data last stored in the table in the device specified by (d) in the device specified by (s). After execution of the FPOP(P) instruction, the device in which the data read by the instruction has been stored is cleared to 0.



N: Number of data
(1) Filled with 0.

- An interlock mechanism should be used to prevent the FPOP(P) instruction from being executed when the value stored in the device specified by (d) is 0.
- A number from 0 to 65535 is stored in the number of data (d).

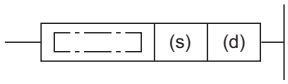
Operation error

Error code (SD0)	Description
3285H	The FPOP(P) instruction is executed when the value in the device specified by (d) is 0.

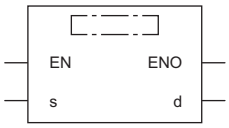
Writing data to the data table

FIFW(P)

These instructions transfer 16-bit binary data to the specified data table.

Ladder	ST
	ENO:=FIFW(EN,s,d); ENO:=FIFWP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FIFW	
FIFWP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Data to be written to the table, or the device number where the write data is stored	—	16-bit signed binary	ANY16
(d)	Start device of table	—	Word	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

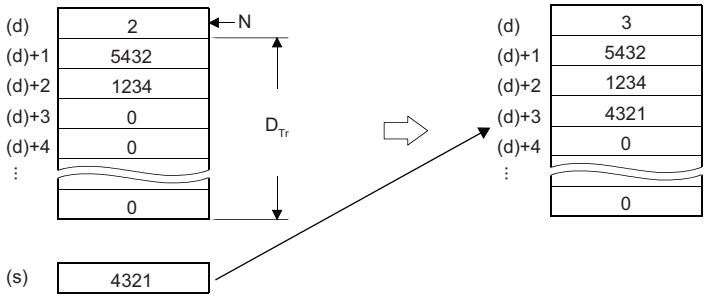
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions store the 16-bit binary data in the device specified by (s) in the data table in the device specified by (d). The number of data stored in the table is stored in (d), and the data in the device specified by (s) is stored in order in the device specified by (d)+1 and later.



N: Number of data

D_{Tr}: Data table range (managed by users)

- When the FIFW(P) is executed for the first time, the value in the device specified by (d) must be cleared.
- A care must be taken for the data table range because data is stored sequentially in the device specified by (d)+1 and later.
- A number from 0 to 65535 is stored in the number of data (d).

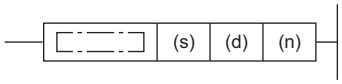
Operation error

Error code (SD0)	Description
3285H	The FIFW(P) instruction is executed when the value in the device specified by (d) is FFFFH.

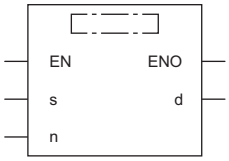
Inserting data to the data table

FINS(P)

These instructions insert 16-bit binary data to the (n)th position in the specified data table.

Ladder	ST
	ENO:=FINS(EN,s,n,d); ENO:=FINSP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FINS	
FINSP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device where the insertion data is stored	—	16-bit signed binary	ANY16
(d)	Start device of table	—	Word	ANY16*1
(n)	Insertion position in the table	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

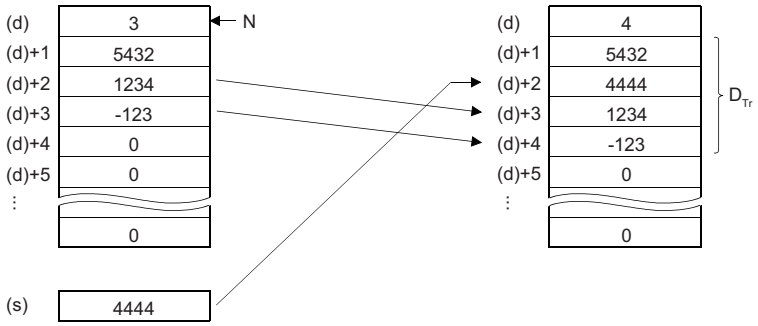
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions store the 16-bit binary data in the device specified by (s) in the (n)th position in the data table in the device specified by (d). After execution of the FINS(P) instruction, the data after the (n)th position in the data table is moved down one by one.



N: Number of data

D_{Tr}: Data table range (managed by users)

When (n)=2, data is inserted to the device specified by (d)+2.

- A number from 0 to 65535 is stored in the number of data (d).

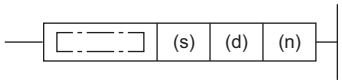
Operation error

Error code (SD0)	Description
3285H	The FINS(P) instruction is executed when the value specified by (n) is 0.
	The FINS(P) instruction is executed when the value in the device specified by (d) is FFFFH.
	The FINS(P) instruction is executed when the value specified by (n) exceeds the number of data specified by (d) +1.

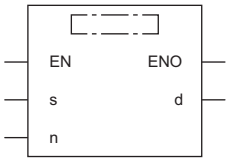
Deleting data from data table

FDEL(P)

These instructions delete the data at the (n)th position in the data table.

Ladder	ST
	ENO:=FDEL(EN,s,n,d); ENO:=FDELP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FDEL	
FDELP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device where the data to be deleted is stored	—	16-bit signed binary	ANY16
(d)	Start device of table	—	Word	ANY16*1
(n)	Deletion position in the table	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

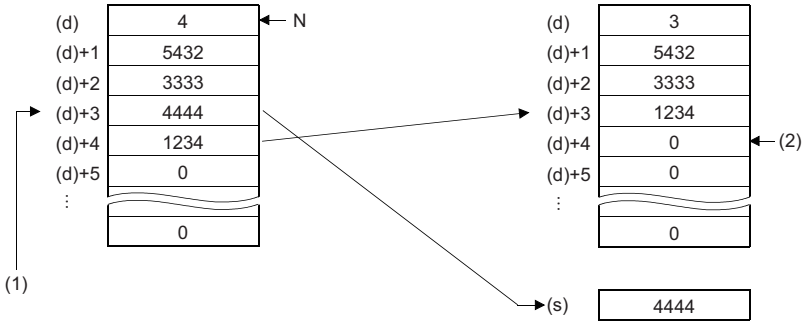
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions delete the (n)th data in the data table in the device specified by (d) and store it in the device specified by (s). After execution of the FDEL(P) instruction, the data at the (n)+1th position and later in the data table is moved forward one by one.



N: Number of data
(1) When (n)=3, data in (d)+3 is deleted.
(2) Filled with 0.

- A number from 0 to 65535 is stored in the number of data (d).

Operation error

Error code (SD0)	Description
3285H	The FDEL(P) instruction is executed when the value specified by (n) is 0.
	The FDEL(P) instruction is executed when the value in the device specified by (d) is 0.
	The FDEL(P) instruction is executed when the value specified by (n) exceeds the number of data specified by (d) +1.

14.3 Reading/Writing Data Instructions

Reading/Writing data to data memory

The data write instruction is an instruction to write arbitrary device data to data memory.
Writing the fixed values used for operation and operation results to data memory can prevent data loss when the battery is low.
The data that has been written to data memory can be read at any time using the data memory read instruction.

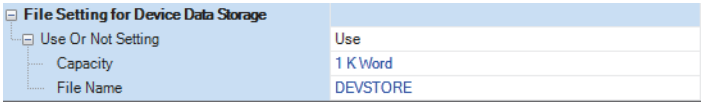
■Execution method

Use the SP.DEVST instruction to write device data to data memory.
Use the S(P).DEVLD instruction to read device data from data memory to any specified device.

■Setting method

When the SP.DEVST and S(P).DEVLD instructions are used, a device data storage file must be set up in advance.

[CPU Parameter] ⇨ [File Setting] ⇨ [File Setting for Device Data Storage]



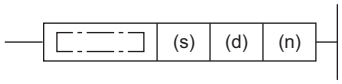
Item	Description
Capacity	1K to 512K (words)
File name	DEVSTORE (fixed)

If data memory does not have enough free space for creating a device data storage file, an error (error code: 21A1H) occurs.
When the controller is switched from STOP to RUN, it is checked to see whether the actual device data storage file matches the parameter setting. If they do not match, an error (error code: 21A0H) occurs.

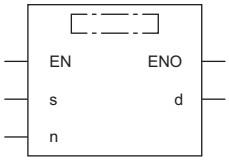
Reading data from the data memory

S(P).DEVLD

These instructions read data from the device data storage file in data memory.

Ladder	ST
	ENO:=S_DEVLD(EN,s,n,d); ENO:=SP_DEVLD(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
S.DEVLD	
SP.DEVLD	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Read offset of device data storage file (specified in units of 16-bit words)	0 to 524287	32-bit unsigned binary	ANY32
(d)	Device for storing the data that has been read	—	Word	ANY16*1
(n)	Number of read points	1 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

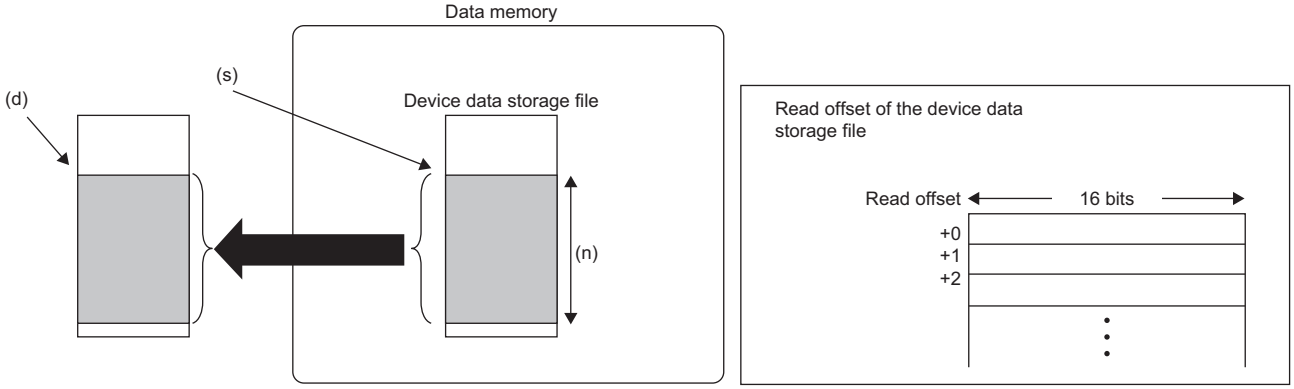
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	—	○	—	—	—	—	○	○	—	—	—

Processing details

- These instructions read device data by the number of points specified by (n) from the read offset specified by (s) in the device data storage file in data memory, and store it in the device specified by (d). (s) indicates the offset from the start of the device data storage file and can be specified by word offsets (incremented by 1 every 16 bits).



- When the S(P).DEVLD instruction is used, a device data storage file must be set up in advance. (☞ Page 605 Setting method)
- Use the SP.DEVST instruction to write device data to the device data storage file. (☞ Page 608 SP.DEVST)

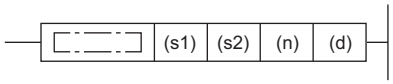
Operation error

Error code (SD0)	Description
2840H	The device data storage file is not set in parameter.
3285H	The value specified by (n) is 0.

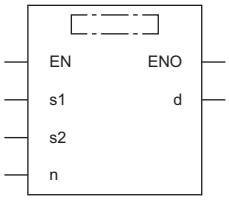
Writing data to the data memory

SP.DEVST

This instruction writes the specified number of points of data to the device data storage file in data memory.

Ladder	ST
	ENO:=SP_DEVST(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.DEVST	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Write offset of device data storage file (specified in units of 16-bit words)	0 to 524287	32-bit unsigned binary	ANY32
(s2)	Start device to which data is to be written	—	Word	ANY16*1
(n)	Number of write points	1 to 65535	16-bit unsigned binary	ANY16
(d)	(d): Completion device, (d)+1: Error completion device	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

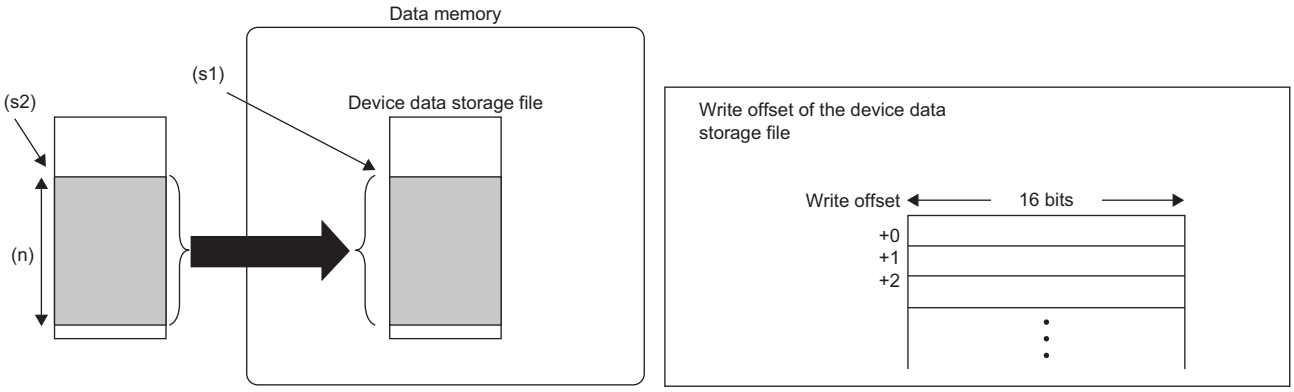
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	○	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	—	○	—	—	—	—	○	○	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions retrieves the specified number of points of data specified by (n) from the device specified by (s2) and write it to the write offset in the device specified by (s1) in the device data storage file in data memory. (s1) indicates the offset from the start of the device data storage file and can be specified by word offsets (incremented by 1 every 16 bits).



- The completion device specified by (d) automatically turns on upon execution of the END instruction following the detection of processing completion of the SP.DEVST instruction and turns off upon execution of the END instruction in the next scan, so it is used as the execution complete flag of the SP.DEVST instruction.
- If the SP.DEVST instruction completes with an error, the error completion device specified by (d)+1 turns on or off at the same time as the completion device specified by (d). Therefore, the device is used as the error completion flag of the SP.DEVST instruction.
- SM753 (File being accessed) turns on while the SP.DEVST instruction is being executed. If SM753 has already been on, the SP.DEVST instruction cannot be executed. (If the instruction is executed, no processing is performed.)
- If an error is detected during execution of the SP.DEVST instruction, the completion device (d), error completion device (d)+1, and SM753 do not turn on.
- When the SP.DEVST instruction is used, a device data storage file must be set up in advance. (☞ Page 605 Setting method)
- Use the S(P).DEVLD instruction to read device data from the device data storage file to any specified device. (☞ Page 606 S(P).DEVLD)

Precautions

- The value written to data memory is the one at execution of the SP.DEVST instruction.
- To prevent the data memory write count from being increased by careless instruction execution, SD771 can be set to limit the write count per day. The maximum number of writes is 36 by default. Change the maximum number of writes by using SD771 as needed. If the specified write count is exceeded, an error occurs with error code 3291H. The number of executions of the instruction to write to data memory per day is initialized to 0 at the following timing.
  - When power off→on, or when reset→reset canceled
  - The date (year, month, day) in clock data is changed by time advancement.
  - Controller internal clock data (year, month, day) is changed by the clock data change function.
- Data is written to the device data storage file when the END instruction is executed. Data is written to the device data storage file when the END instruction is executed immediately after the SP.DEVST instruction is executed. Thus, depending on the number of write points, writing to the device data storage file may involve multiple scans. Check the completion device to see whether the writing is completed.

Operation error

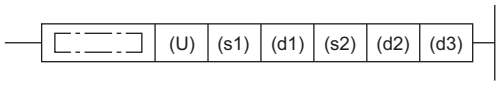
Error code (SD0)	Description
2840H	The device data storage file is not set in parameter.
3285H	The value specified by (n) is 0.
3291H	When the SP.DEVST instruction is executed, the write count of the day exceeds the value specified in SD771.
	When the SP.DEVST instruction is executed, a value out of the range (1 to 32767) is set in SD771.

14.4 File Operation Instructions

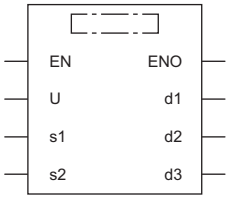
Reading data from the specified file

SP.FREAD

This instruction reads device data from the specified file in an SD memory card.

Ladder	ST
	ENO:=SP_FREAD(EN,U,s1,s2,d1,d2,d3);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.FREAD	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	Device name	ANY16
(s1)	Drive specification	2 (fixed)*1	Word	ANY16
(d1)	Start device where the control data is stored	☞ Page 611 Control data (d1)	Word	ANY16_ARRAY (Number of elements: 8)
(s2)	Start device where the file name is stored	—	Unicode string	ANYSTRING_DOUBLE
(d2)	Start device for storing the data that has been read	—	Word	ANY16*2
(d3)	Bit device that turns on upon completion of the processing (In the case of an error completion, the device specified by (d3)+1 also turns on.)	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only drive 2 (for the SD memory card) can be set.

*2 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(d3)	○	—	○	—	—	—	—	—	—	—	—	—

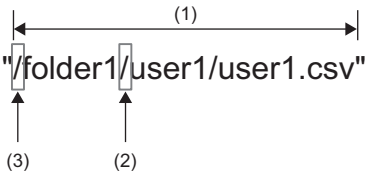
■Control data (d1)

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution/completion type	Specify the execution type. ■00**H: Reading binary data • 0000H: 16-bit binary data • 0001H: 32-bit binary data ■01**H: Reading data after converted to CSV format • 0100H: Decimal (16-bit data) • 0110H: Decimal (32-bit data) • 0120H: Hexadecimal (16-bit data) • 0121H: Hexadecimal (32-bit data) • 0130H: String (ASCII data) • 0140H: Floating point real number (single-precision real number) • 0141H: Floating point real number (double-precision real number)	0000H 0001H 0100H 0110H 0120H 0121H 0130H 0140H 0141H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code) (☞ Page 660 Error codes generated for file operation instructions)	—	System
+2	Number of read-target data	Specify the number of read-target data (in units of words). The processing unit and setting range depend on (d1)+0. ■When "Reading binary data" is specified by (d1)+0 • When 16-bit binary data is specified: In units of words (1 to 65535)*1*2 • When 32-bit binary data is specified: In units of double words (1 to 32767) ■When "Reading data after converted to CSV format" is specified by (d1)+0 • When decimal (16-bit data) is specified: Number of elements (1 to 65535)*1*2 • When decimal (32-bit data) is specified: Number of elements (1 to 32767) • When hexadecimal (16-bit data) is specified: Number of elements (1 to 65535) • When hexadecimal (32-bit data) is specified: Number of elements (1 to 32767) • When a string (ASCII data) is specified: Number of elements (1 to 1023) • When a floating point real number (single-precision real number) is specified: Number of elements (1 to 32767) • When a floating point real number (double-precision real number) is specified: Number of elements (1 to 16383)	1 to 65535	User
+3	Maximum number of read data	■When "0130H: String (ASCII data)" is specified by (d1)+0 • Total size of the characters in the element (in units of words) ■When data other than "0130H: String (ASCII data)" is specified by (d1)+0 • Fixed to 0	0, 1 to 65535	User
+4 +5	File position	■When "Reading binary data" is specified by (d1)+0 • 00000000H: From the beginning of the file • 00000001H to FFFFFFFEH: From the specified position (The data unit is determined by (d1)+7.) • FFFFFFFFH: Cannot be specified. ■When "Reading data after converted to CSV format" is specified by (d1)+0 • 00000000H: From the beginning of the file • 00000001H to FFFFFFFEH: From the specified row • FFFFFFFFH: From the position where the previous reading ends	00000000H to FFFFFFFFH	User
+6	Number of columns	When "Reading binary data" is specified by (d1)+0, set 0. When "Reading data after converted to CSV format" is specified by (d1)+0, set the number of read-target columns. • 0: No column (Regarded as one row.) • Other than 0: Number of specified columns	0000H to FFFFH (0 to 65535)	User
+7	Data type specification	• 0: Word • 1: Even number of bytes*1 • 2: Unit of the data type specified by (d1)+0 • 3: Odd number of bytes*1*2 "0: Word", "1: Even number of bytes", and "3: Odd number of bytes" can be specified only when "0000H: 16-bit binary data" or "0100H: Decimal (16-bit data)" is specified by (d1)+0.	0, 1, 2, 3	User

*1 When "1: Even number of bytes" or "3: Odd number of bytes" is specified by (d1)+7 (Data type specification), the setting range of (d1)+2 (Number of read-target data) is 1 to 32767.

*2 Even when "3: Odd number of bytes" is specified by (d1)+7 (Data type specification), specify the number in units of words. Add one byte to the odd number of bytes to be read and set the number in (d1)+2 (Number of read-target data).

■File name (s2)

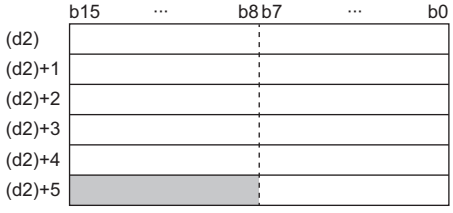
Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +☐	File name string	Specify the folder path where the files are stored and the file name. • The folder path and file name (including an extension) must be within 253 characters in total. • The folder path must be within 244 characters. (Delimiters are not included.) • The number of folder path hierarchies must be within 10 levels. • When omitting an extension in the file name, omit the ". (period)" as well. • The file name (not including the extension) must be between 1 and 60 characters, and the extension must be within a period + 3 characters. If the file name (not including the extension) is 61 characters or more, the extension is ignored and replaced with ".BIN" or ".CSV".  (1): Up to 253 characters (2): Use "/" or "\" as delimiters for the folder path and file. (3): Can be omitted. When it is omitted, (1) is up to 252 characters.	Unicode string	User

■Read data (d2)

Operand: (d2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Number of data actually read	The number of data actually read is set. The data unit is determined by (d1)+7 (Data type specification).	—	System
+1 to +☐	Read data	The read data is stored.	—	System

Processing details

- This instruction reads data from the specified file. Set the execution/completion type in the control data to specify the file read-target format.
- The read target is the data in the SD memory card only.
- The processing completion bit device (d3) automatically turns on at the execution of the END instruction in the scan in which the completion of processing of the SP.FREAD instruction is detected. The bit device (d3) turns off at the execution of the END instruction in the next scan.
- If the SP.FREAD instruction completes with an error, the error completion device (d3)+1 turns on or off in synchronization with (d3).
- SM753 (File being accessed) turns on while the SP.FREAD instruction is being executed.
- While SM753 is on, the SP.FREAD instruction cannot be executed. (If the instruction is executed, no processing is performed.)
- If an error is detected during the execution of the instruction, Processing Complete (d3), Error Completion (d3)+1, and SM753 do not turn on.
- When "Odd number of bytes" is specified by (d1)+7, the read data is stored as shown below. (In the shaded area shown below, the data before execution of the instruction is kept.)



• Specify data in (d1)+2, (d1)+4, (d1)+5, and (d2)+0 depending on the combination of (d1)+0 and (d1)+7.

Execution/completion type (d1)+0		Data type specification (d1)+7	Processing unit and setting range		
			Number of read-target data (d1)+2	File position (d1)+4, (d1)+5	Number of data actually read (d2)+0
Reading binary data	0000H: 16-bit binary data	0: Word	Word (1 to 65535)	Word (00000000H to 7FFFFFFFH)	Word
		1: Even number of bytes	Word (1 to 32767)	Byte (00000000H to FFFFFFFEH)	Byte
		2: Unit of the data type specified by the execution/completion type	Word (1 to 65535)	Word (00000000H to 7FFFFFFFH)	Word
		3: Odd number of bytes	Word (1 to 32767)	Byte (00000000H to FFFFFFFEH)	Byte
	0001H: 32-bit binary data	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Double word (1 to 32767)	Double word (00000000H to 3FFFFFFFH)	Double word
		3: Odd number of bytes	(Cannot be specified)		
Reading data after converted to CSV format	0100H: Decimal (16-bit data)	0: Word	Number of elements (1 to 65535)	Number of lines	Word
		1: Even number of bytes	Number of elements (1 to 32767)	Number of lines	Byte
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 65535)	Number of lines	Word
		3: Odd number of bytes	Number of elements (1 to 32767)	Number of lines	Byte
	0110H: Decimal (signed 32-bit data)	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 32767)	Number of lines	Double word
		3: Odd number of bytes	(Cannot be specified)		
	0120H: Hexadecimal (16-bit data)	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 65535)	Number of lines	Word
		3: Odd number of bytes	(Cannot be specified)		
	0121H: Hexadecimal (32-bit data)	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 32767)	Number of lines	Double word
		3: Odd number of bytes	(Cannot be specified)		
	0130H: String (ASCII data)	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 1023)	Number of lines	Number of elements
		3: Odd number of bytes	(Cannot be specified)		
	0140H: Floating point real number (single-precision real number)	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 32767)	Number of lines	Double word
		3: Odd number of bytes	(Cannot be specified)		
	0141H: Floating point real number (double-precision real number)	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 16383)	Number of lines	4 words
		3: Odd number of bytes	(Cannot be specified)		

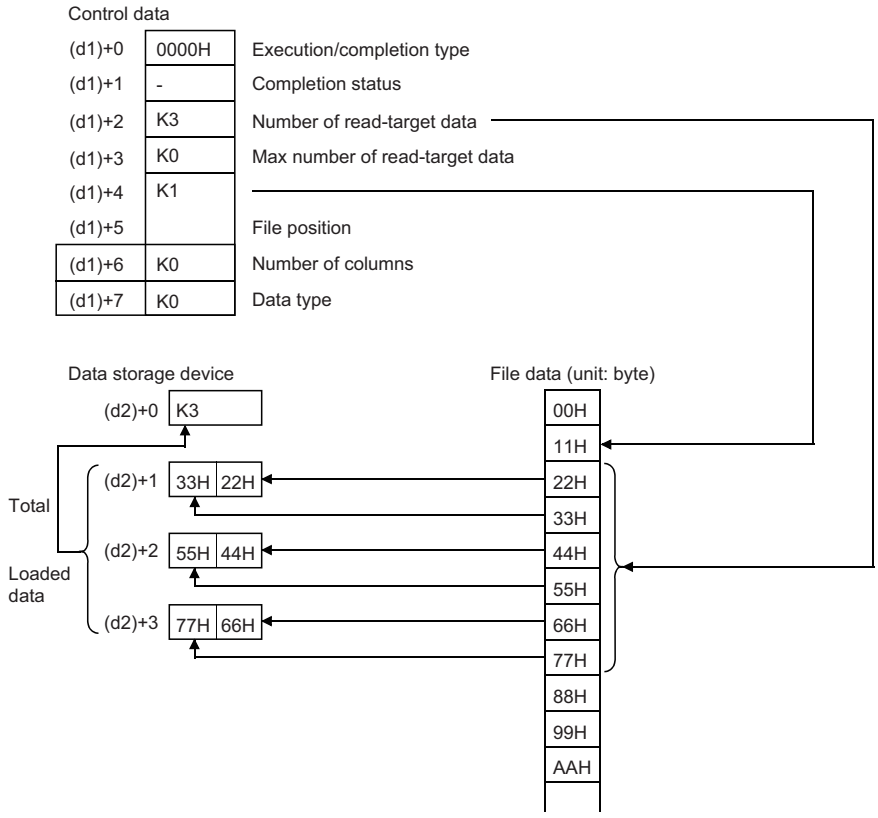
■When reading binary data

- If the extension of the target file is omitted, the extension will be ".BIN".
- If the specified file does not exist, an error occurs.
- If the position is specified exceeding the existing file size, 0 point of data is read and the processing completes successfully.

The following figure shows an example of reading binary data.

Ex.

Example of reading binary data (16-bit binary data)



■When reading data after converted to CSV format

- Elements in the CSV format file (cells in Excel®) are read by each row and stored in the device.
- If the extension of the target file is omitted, the extension will be ".CSV".
- If the specified file does not exist, an error occurs.
- If the position is specified exceeding the existing file size, 0 point of data is read and the processing completes successfully.
- The number of data specified by (d1)+2 is read from the beginning of the file. If the last data in the file is read before reaching to the number specified, only the available number of data is read.
- When the number of columns is set to 0, the data are read by ignoring the rows in the CSV format file.
- The string data in the CSV file and the value stored in the device after it is read are determined by the execution/completion type.

Execution/completion type	Data (one element) in the CSV file	Value stored in the device	Remarks
0100H: Decimal (16-bit data)	-32768 to -1	-32768 to -1 (32768 to 65535)	The value should be within the range of -32768 to -1 as signed 16-bit data and within the range of 32768 to 65535 as unsigned 16-bit data. The same value is stored in the device.
	0 to 32767	0 to 32767	—
	32768 to 65535	-32768 to -1 (32768 to 65535)	The value should be within the range of -32768 to -1 as signed 16-bit data and within the range of 32768 to 65535 as unsigned 16-bit data. The same value is stored in the device.
	• Numeric values other than above • String containing alphabets and symbols	0	Filled with 0 since it cannot be converted.
0110H: Decimal (32-bit data)	-2147483648 to -1	-2147483648 to -1 (2147483648 to 4294967295)	The value should be within the range of -2147483648 to -1 as signed 32-bit data and within the range of 2147483648 to 4294967295 as unsigned 32-bit data. The same value is stored in the device.
	0 to 2147483647	0 to 2147483647	—
	2147483648 to 4294967295	-2147483648 to -1 (2147483648 to 4294967295)	The value should be within the range of -2147483648 to -1 as signed 32-bit data and within the range of 2147483648 to 4294967295 as unsigned 32-bit data. The same value is stored in the device.
	• Numeric values other than above • String containing alphabets and symbols	0	Filled with 0 since it cannot be converted.
0120H: Hexadecimal (16-bit data)	0H to FFFFH	0H to FFFFH	—
	• Numeric values other than above • String containing alphabets other than A to F and symbols	0000H	Filled with 0 since it cannot be converted.
0121H: Hexadecimal (32-bit data)	0H to FFFFFFFFH	0H to FFFFFFFFH	—
	• Numeric values other than above • String containing alphabets other than A to F and symbols	00000000H	Filled with 0 since it cannot be converted.
0130H: String (ASCII data)	String (up to 1999 characters)	String (up to 1999 characters)	NULL (00H) is added to the end of the string. When the number of bytes of the string in the CSV file is even, 0000H is stored in the next one word. When the string in the CSV file contains 00H, it is ignored.
	String (2000 characters or more)		If the number of characters in one element exceeds 1999, characters until the 1999th character are read as one element. The 2000th character and after are not read and the next element is read.
0140H: Floating point real number (single-precision real number)	Values within the range of: $-2^{128} < \text{data} \leq -2^{-126}$, 0, $2^{-126} \leq \text{data} < 2^{128}$	As given on the left	The value is converted in either decimal point or exponential format.
	Numeric values other than above	0	Filled with 0 since it cannot be converted.
0141H: Floating point real number (double-precision real number)	Values within the range of: $-2^{1024} < \text{data} \leq -2^{-1022}$, 0, $2^{-1022} \leq \text{data} < 2^{1024}$	As given on the left	The value is converted in either decimal point or exponential format.
	Numeric values other than above	0	Filled with 0 since it cannot be converted.

Ex.

When "Reading data after converted to CSV format" (String (ASCII data)) is specified

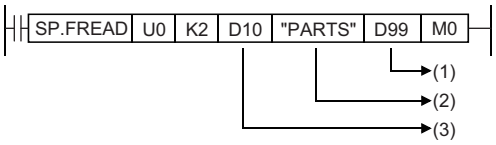
[Data stored in CSV format]

PARTS.CSV

No.	,	Name	,	Value1	,	Value2	CR	LF
AA_0001	,	Prts_A	,	100	,	200	CR	LF
BB_0002	,	Prts_B	,	300	,	400	CR	LF

Enclosed values as shown to the left are read.
(Total of six elements, three columns from two lines of PARTS.CSV, are read.)

[Data to be loaded to the device]



(1) Loaded data
(2) File name
(3) Control data

[Control data]

D10	H0130
D11	H0000
D12	K6
D13	K100
D14	K2
D15	
D16	K3
D17	K2

D10: Execution/completion type
D11: Completion status
D12: Number of read-target data
D13: Maximum number of read data
D14, D15: File positions
D16: Number of columns
D17: Data type specification

PARTS.CSV

No.	,	Name	,	Value1	,	Value2	CR	LF
AA_0001	,	Prts_A	,	100	,	200	CR	LF
BB_0002	,	Prts_B	,	300	,	400	CR	LF

D099	K6	
D100	H4141	"AA_0001"
D101	H305F	
D102	H3030	
D103	H0031	
D104	H7250	"Prts_A"
D105	H7374	
D106	H415F	
D107	H0000	
D108	H3031	"100"
D109	H0030	
D110	H4242	"BB_0002"
D111	H305F	
D112	H3030	
D113	H0032	
D114	H7250	"Prts_B"
D115	H7374	
D116	H425F	
D117	H0000	
D118	H3033	"300"
D119	H0030	

D99: Number of data actually read
D100 to D103: String in the 1st column of the 2nd line
D104 to D107: String in the 2nd column of the 2nd line
D108, D109: String in the 3rd column of the 2nd line
D110 to D113: String in the 1st column of the 3rd line
D114 to D117: String in the 2nd column of the 3rd line
D118, D119: String in the 3rd column of the 3rd line

Ex.

When "Reading data after converted to CSV format" (Decimal (16-bit data)) is specified and the number of columns is set to 0

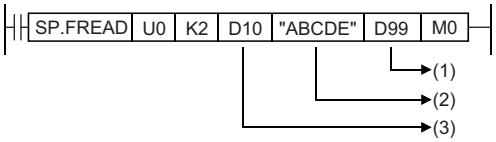
[Data created in Excel]

	A	B	C
1	Main/sub item		Measured value
2	Length	1	3
3	Temperature	-21	
4			

[Data saved in CSV format]

Main/sub item	,	,	Measured value				CR	LF	
Length	,	1	,	3	CR	LF			
Temperature	,	-21	,	CR	LF				

[Data to be loaded to the device]



(1) Loaded data
(2) File name
(3) Control data

[Control data]

D10	0100H
D11	-
D12	K9
D13	K0
D14	K0
D15	K0
D16	K0
D17	K0

D10: Execution/completion type
D11: Completion status
D12: Number of read-target data
D13: Maximum number of read data
D14, D15: File positions
D16: Number of columns
D17: Data type specification

[Loaded data]

D99	K9
D100	K0
D101	K0
D102	K0
D103	K0
D104	K1
D105	K3
D106	K0
D107	K-21
D108	K0

D99: Number of data actually read
D100: Since "Main/sub item" is not a numerical value, the conversion data, 0, is stored.
D101: Since " " is not a numerical value, the conversion data, 0, is stored.
D102: Since "Measured value" is not a numerical value, the conversion data, 0, is stored.
D103: Since "Length" is not a numerical value, the conversion data, 0, is stored.
D104: Since "1" is a numerical value, it is converted to a binary value.
D105: Since "3" is a numerical value, it is converted to a binary value.
D106: Since "Temperature" is not a numerical value, the conversion data, 0, is stored.
D107: Since "-21" is a numerical value, it is converted to a binary value.
D108: Since " " is not a numerical value, the conversion data, 0, is stored.

• When the number of columns differs in each row, the data are also read by ignoring the rows.

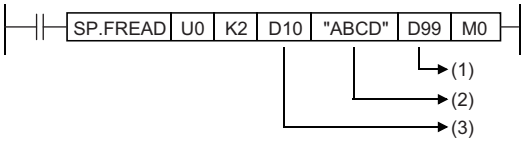
Ex.

When the number of columns differs in each row

[Data saved in CSV format]

Main/sub item	,	,	Measured value	,	Excess	CR	LF
Length	CR	LF					
Temperature	,	-21	,	CR	LF		

[Data to be loaded to the device]



(1) Loaded data
(2) File name
(3) Control data

[Control data]

D10	0100H
D11	-
D12	K7
D13	K0
D14	K0
D15	K0
D16	K0
D17	K0

D10: Execution/completion type
D11: Completion status
D12: Number of read-target data
D13: Maximum number of read data
D14, D15: File positions
D16: Number of columns
D17: Data type specification

[Loaded data]

D99	K7
D100	K0
D101	K0
D102	K0
D103	K0
D104	K0
D105	K0
D106	K-21

D99: Number of data actually read
D100: Since "Main/sub item" is not a numerical value, the conversion data, 0, is stored.
D101: Since " " is not a numerical value, the conversion data, 0, is stored.
D102: Since "Measured value" is not a numerical value, the conversion data, 0, is stored.
D103: Since "Excess" is not a numerical value, the conversion data, 0, is stored.
D104: Since "Length" is not a numerical value, the conversion data, 0, is stored.
D105: Since "Temperature" is not a numerical value, the conversion data, 0, is stored.
D106: Since "-21" is a numerical value, it is converted to a binary value.

Point

This type of file in which the number of columns vary with individual rows cannot be created by Excel. It is created when the CSV file is modified by a user.

- When the specified number of columns is set to a value other than 0, a CSV format file is read as the table with the specified number of columns. The elements outside the specified number of columns are ignored.

Ex.

When "Reading data after converted to CSV format" (Decimal (16-bit data)) is specified and the number of columns is set to a value other than 0 ((d1)+6 is set to 2)

[Data created in Excel]

	A	B	C
1	Main/sub item		Measured value
2	Length	1	3
3	Temperature	-21	
4			

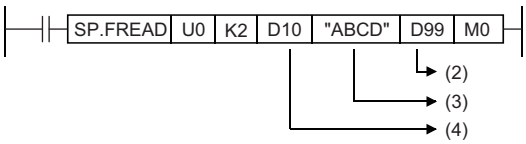
[Data saved in CSV format]

Main/sub item	,	,	Measured value			CR	LF	
Length	,	1	,	3		CR	LF	
Temperature	,	-21	,			CR	LF	

(1)

(1) Columns out of the specified range are ignored.

[Data to be loaded to the device]



(2) Loaded data
(3) File name
(4) Control data

[Control data]

D10	0100H
D11	-
D12	K6
D13	K0
D14	K0
D15	K0
D16	K2
D17	K0

D10: Execution/completion type
D11: Completion status
D12: Number of read-target data
D13: Maximum number of read data
D14, D15: File positions
D16: Number of columns
D17: Data type specification

[Loaded data]

D99	K6
D100	K0
D101	K0
D102	K0
D103	K1
D104	K0
D105	K-21

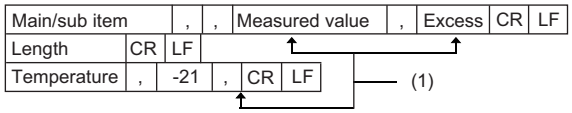
D99: Number of data actually read
D100: Since "Main/sub item" is not a numerical value, the conversion data, 0, is stored.
D101: Since " " is not a numerical value, the conversion data, 0, is stored.
D102: Since "Length" is not a numerical value, the conversion data, 0, is stored.
D103: Since "1" is a numerical value, it is converted to a binary value.
D104: Since "Temperature" is not a numerical value, the conversion data, 0, is stored.
D105: Since "-21" is a numerical value, it is converted to a binary value.

- When the number of columns differs in each row, the elements outside the specified number of columns are ignored and 0 is added to the cells where no element exists.

Ex.

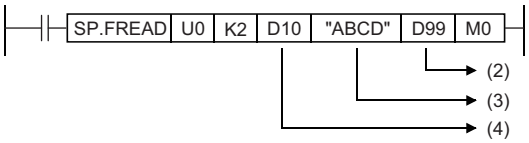
When the number of columns differs in each row

[Data saved in CSV format]



(1) Columns out of the specified range are ignored.

[Data to be read into devices]



(2) Loaded data
(3) File name
(4) Control data

[Control data]

D10	0100H
D11	-
D12	K6
D13	K0
D14	K0
D15	K0
D16	K2
D17	K0

D10: Execution/completion type
D11: Completion status
D12: Number of read-target data
D13: Maximum number of read data
D14, D15: File positions
D16: Number of columns
D17: Data type specification

[Loaded data]

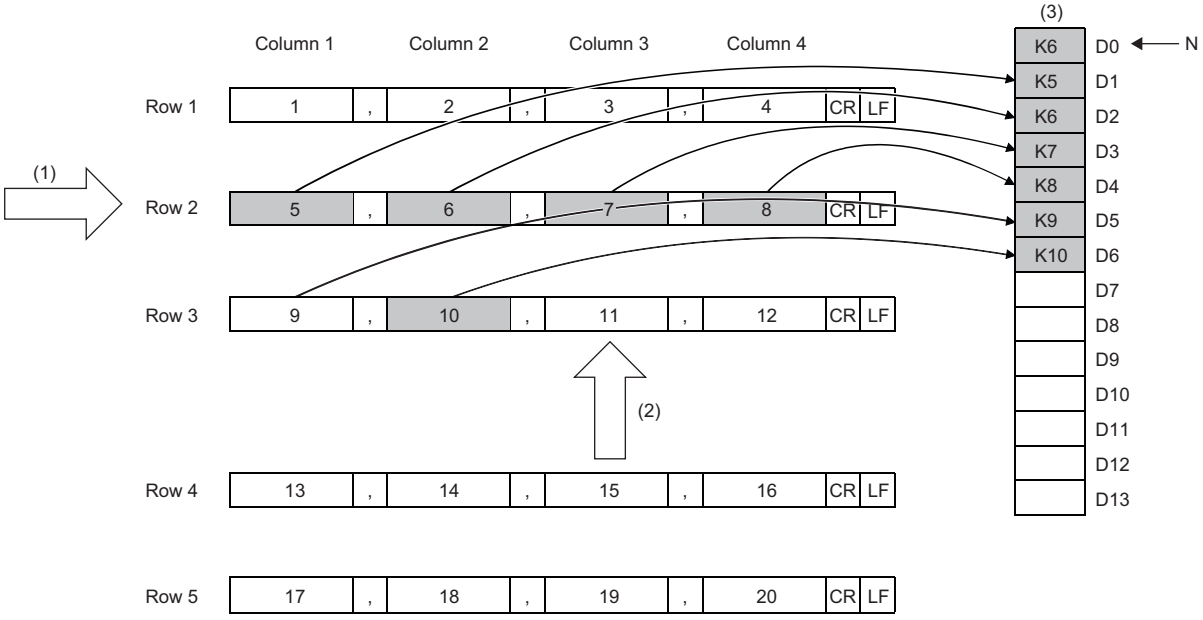
D99	K6
D100	K0
D101	K0
D102	K0
D103	K0
D104	K0
D105	K-21

D99: Number of data actually read
D100: Since "Main/sub item" is not a numerical value, the conversion data, 0, is stored.
D101: Since " " is not a numerical value, the conversion data, 0, is stored.
D102: Since "Length" is not a numerical value, the conversion data, 0, is stored.
D103: Since no element exists, the conversion data, 0, is added.
D104: Since "Temperature" is not a numerical value, the conversion data, 0, is stored.
D105: Since "-21" is a numerical value, it is converted to a binary value.

• When "Reading data after converted to CSV format" is specified, data can be divided and read.

[Specify row to start reading]

- Execution/completion type: Reading data after converted to CSV format (Decimal (16-bit binary data))
- Number of columns specification: 4H
- Data type specification: Words
- File position: 2H
- Read start device: D0
- Number of data actually read: 6H



(1) Starting row

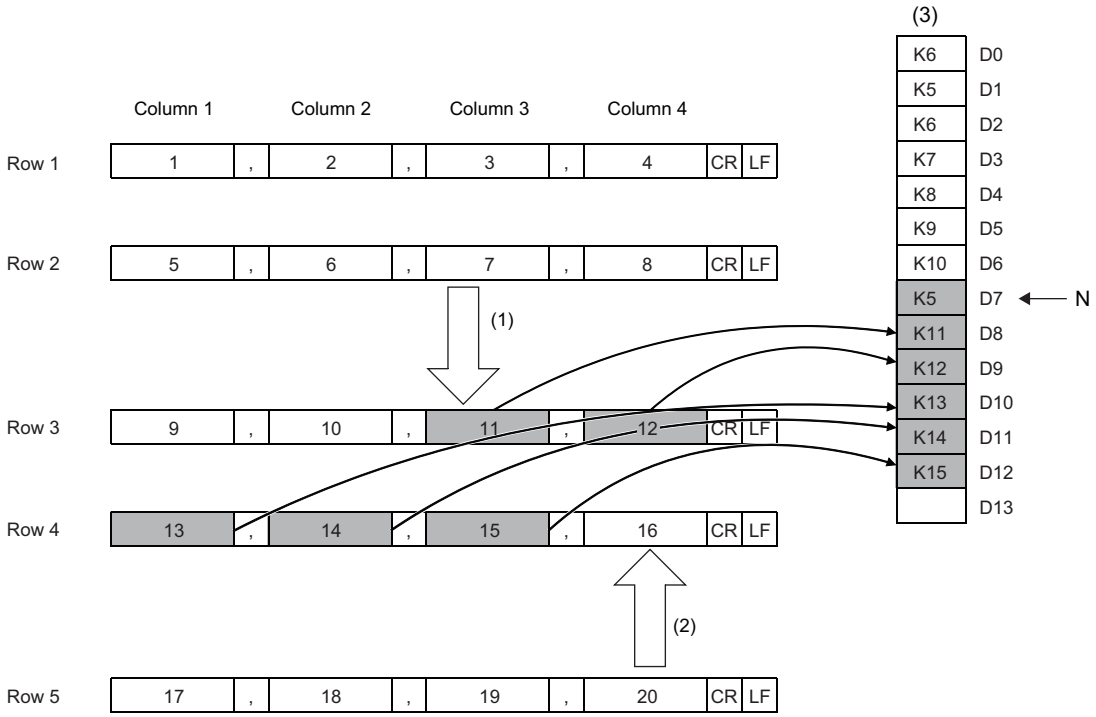
(2) Next starting position

(3) Data in the device (loaded data)

N: Number of data

[Reading data from the position where the previous reading ends]

- Execution/completion type: Reading data after converted to CSV format (Decimal (16-bit binary data))
- Number of columns specification: 4H
- Data type specification: Words
- File position: FFFFFFFFH (continuing from the position where the previous reading ends)
- Read start device: D7
- Number of data actually read: 5H



(1) Starting row

(2) Next starting position

(3) Data in the device (loaded data)

N: Number of data

Point

- When reading data from the position where the previous reading ends, specify the same values for "Execution/End type", "Number of columns", and "Data type specification". If not, data cannot be added correctly from the position where the previous reading ends.
- While reading data from the position where the previous reading ends, if the SP.FREAD instruction with different settings or the SP.FWRITE instruction is executed, data cannot be added correctly from the position where the previous reading ends.

Precautions

- Do not execute this instruction in interrupt programs. Doing so may cause malfunction of the module.
- When reading multiple elements at reading data after converted to CSV format, secure devices sufficient for the total size of the elements in the read data area before executing the instruction. Since read data is stored from (d2)+1, the number of words required to be secured as (d2) is ((total number of words for each element) + 1) words.
- When "Reading data after converted to CSV format" (String (ASCII data)) is specified, set the total size (in units of words) to (d1)+3 (Maximum number of read data).

  [Example] When reading 100 elements each of which contains 100 characters from a CSV file

  (100 (characters) + 2 (NULL)) × 100 (elements) = 10200 bytes = 5100 words

  Therefore, set 5100 to (d1)+3 to secure an area for 5101 words for (d2).
- Since the SP.FREAD instruction causes file reading, the scan time may be extended when this instruction is executed.
- The SP.FREAD instruction does not support character codes with BOM. If a file that uses a character code with BOM is executed, unintended data will be read. To read a CSV file with BOM that was created using Excel, use an application such as a text editor to change the character code to exclude BOM.

Operation error

Error code (SD0)	Description
2820H	Data is read exceeding the size of the device.
3285H	The drive specified by (s1) is not the one for the SD memory card.
	Any value that is set in the device specified by (d1) and later as control data is out of the range. ☞ Page 611 Control data (d1)
	The file name string specified by (s2) cannot be read. • The number of characters of the string in the file name specified exceeds the range. • An inhibited value is set. • The specified file name string ends with a delimiter. • The file name (not including the extension) of the specified file name string is 0 characters. ☞ Page 611 Control data (d1)
3297H	An invalid combination of (d1)+0 (Execution/completion type) and (d1)+7 (Data type specification) is specified. ☞ Page 611 Control data (d1)

If the SP.FREAD instruction completes with an error, an error code is stored in the device specified by (d1)+1. (Note that an error code is not stored if the instruction results in an operation error.)

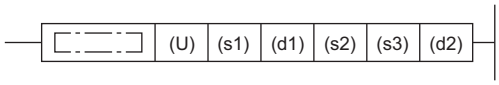
For the error code stored in (d1)+1, refer to the following.

☞ Page 660 Error codes generated for file operation instructions

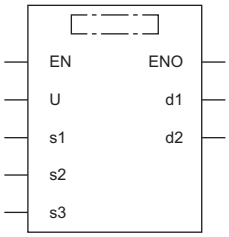
Writing data to the specified file

SP.FWRITE

This instruction writes device data to the specified file in an SD memory card.

Ladder	ST
	ENO:=SP_FWRITE(EN,U,s1,s2,s3,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.FWRITE	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	Device name	ANY16
(s1)	Drive specification	2 (fixed)*1	Word	ANY16
(d1)	Start device where the control data is stored	☞ Page 625 Control data (d1)	Word	ANY16_ARRAY (Number of elements: 8)
(s2)	Start device where the file name is stored	—	Unicode string	ANYSTRING_DOUBLE
(s3)	Start device where data is stored	—	Word	ANY16*2
(d2)	Bit device that turns on upon completion of the processing	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

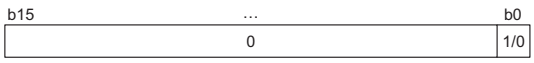
*1 Only drive 2 (for the SD memory card) can be set.

*2 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○	—	—	—	—	—	—	—	—	—

■Control data (d1)

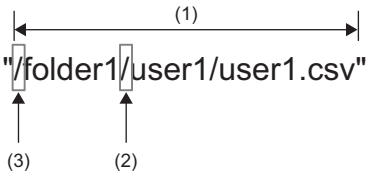
Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution/completion type	Specify the execution type. ■00**H: Writing binary data • 0000H: 16-bit binary data • 0001H: 32-bit binary data ■01**H: Writing data after converted to CSV format • 0100H: Decimal (signed 16-bit data) • 0101H: Decimal (unsigned 16-bit data) • 0110H: Decimal (signed 32-bit data) • 0111H: Decimal (unsigned 32-bit data) • 0120H: Hexadecimal (16-bit data) • 0121H: Hexadecimal (32-bit data) • 0130H: String (ASCII data) • 0140H: Floating point real number (single-precision real number) • 0141H: Floating point real number (double-precision real number)	0000H 0001H 0100H 0101H 0110H 0111H 0120H 0121H 0130H 0140H 0141H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code) (☞ Page 660 Error codes generated for file operation instructions)	—	System
+2	Number of data actually written	For the data in the device specified by (s3), the number of data actually written is stored. The data unit is determined by (d1)+7 (Data type specification).	—	System
+3	Application setting area	 b0: Write start position setting*3 When "Writing binary data" is specified by (d1)+0, set 0. When "Converting and writing data in CSV format" is specified by (d1)+0, and when "Added to the end of the file" is specified by (d1)+4 and (d1)+5, specify the write start position. • 0: Added to the end of the file • 1: Convert the return code at the end of the file to a comma and add it (write as continuation of the last line).	Refer to the "Description" column.	User
+4 +5	File position	■When "Writing binary data" is specified by (d1)+0 • 00000000H: From the beginning of the file • 00000001H to FFFFFFFEH: From the specified position (The data unit is determined by (d1)+7.) • FFFFFFFFH: Added to the end of the file. ■When "Converting and writing data in CSV format" is specified by (d1)+0 • 00000000H to FFFFFFFEH: From the beginning of the file • FFFFFFFFH: Added to the end of the file.	00000000H to FFFFFFFFH	User
+6	Number of columns	When "Writing binary data" is specified by (d1)+0, set 0. When "Converting and writing data in CSV format" is specified by (d1)+0, specify the number of columns to write. • 0: No column (Regarded as one row.) • Other than 0: Number of specified columns	0000H to FFFFH (0 to 65535)	User
+7	Data type specification	• 0: Word • 1: Even number of bytes*1 • 2: Unit of the data type specified by (d1)+0 • 3: Odd number of bytes*1*2 "0: Word", "1: Even number of bytes", and "3: Odd number of bytes" can be specified only when "0000H: 16-bit binary data" or "0100H: Decimal (signed 16-bit data)" is specified by (d1)+0.	0, 1, 2, 3	User

*1 When "1: Even number of bytes" or "3: Odd number of bytes" is specified by (d1)+7 (Data type specification), the setting range of (s3)+0 (Number of request write data) is 1 to 32767.

*2 Even when "3: Odd number of bytes" is specified by (d1)+7 (Data type specification), specify the number in units of words. Add one byte to the odd number of bytes to be written and set the number in (s3)+0 (Number of request write data).

*3 Enabled only when "Converting and writing data in CSV format" is specified by (d1)+0 (Execution/completion type), and when "Added to the end of the file" is specified by (d1)+4 and (d1)+5 (File position). Otherwise, an error will occur.

■File name (s2)

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +☐	File name string	Specify the folder path where the files are stored and the file name. • The folder path and file name (including an extension) must be within 253 characters in total. • The folder path must be within 244 characters. (Delimiters are not included.) • The number of folder path hierarchies must be within 10 levels. • When omitting an extension in the file name, omit the ". (period)" as well. • The file name (not including the extension) must be between 1 and 60 characters, and the extension must be within a period + 3 characters. If the file name (not including the extension) is 61 characters or more, the extension is ignored and replaced with ".BIN" or ".CSV".  (1): Up to 253 characters (2): Use "/" or "\" as delimiters for the folder path and file. (3): Can be omitted. When it is omitted, (1) is up to 252 characters.	Unicode string	User

■Write data (s3)

Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Number of request write data	Specify the number of data to be requested to write. The processing unit and setting range depend on the value set in (d1)+0. ■When "Writing binary data" is specified by (d1)+0 • When 16-bit binary data is specified: In units of words (1 to 65535)*1*2 • When 32-bit binary data is specified: In units of double words (1 to 32767) ■When "Converting and writing data in CSV format" is specified by (d1)+0: Number of elements • When decimal (signed 16-bit data) is specified: In units of words (1 to 65535)*1*2 • When decimal (unsigned 16-bit data) is specified: In units of words (1 to 65535) • When decimal (signed 32-bit data) is specified: In units of double words (1 to 32767) • When decimal (unsigned 32-bit data) is specified: In units of double words (1 to 32767) • When hexadecimal (16-bit data) is specified: In units of words (1 to 65535) • When hexadecimal (32-bit data) is specified: In units of double words (1 to 32767) • When a string (ASCII data) is specified: Number of elements (1 to 1023) • When a floating point real number (single-precision real number) is specified: In units of double words (1 to 32767) • When a floating point real number (double-precision real number) is specified: In units of 4 words (1 to 16383)	1 to 65535	User
+1 to +☐	Write data	The number of data to be requested to write is stored.	0000H to FFFFH	User

*1 When "1: Even number of bytes" or "3: Odd number of bytes" is specified by (d1)+7 (Data type specification), the setting range of (s3)+0 (Number of request write data) is 1 to 32767.

*2 Even when "3: Odd number of bytes" is specified by (d1)+7 (Data type specification), specify the number in units of words. Add one byte to the odd number of bytes to be written and set the number in (s3)+0 (Number of request write data).

Processing details

- This instruction writes the specified number of data to the specified file. Set the execution/completion type in the control data to specify the file write-target format.
- The write target is the SD memory card only.
- The processing completion bit device (d2) automatically turns on at the execution of the END instruction in the scan in which the completion of processing of the SP.FWRITE instruction is detected. The bit device (d2) turns off at the execution of the END instruction in the next scan.
- If the SP.FWRITE instruction completes with an error, the error completion device (d2)+1 turns on or off in synchronization with (d2).
- SM753 (File being accessed) turns on while the SP.FWRITE instruction is being executed.
- While SM753 is on, the SP.FWRITE instruction cannot be executed. (If the instruction is executed, no processing is performed.)
- Even if an error is detected during the execution of the instruction, the bit devices (d2) and (d2)+1 and SM753 do not turn on.
- Specify data in (s3)+0, (d1)+4, (d1)+5, and (d1)+2 depending on the combination of (d1)+0 and (d1)+7.

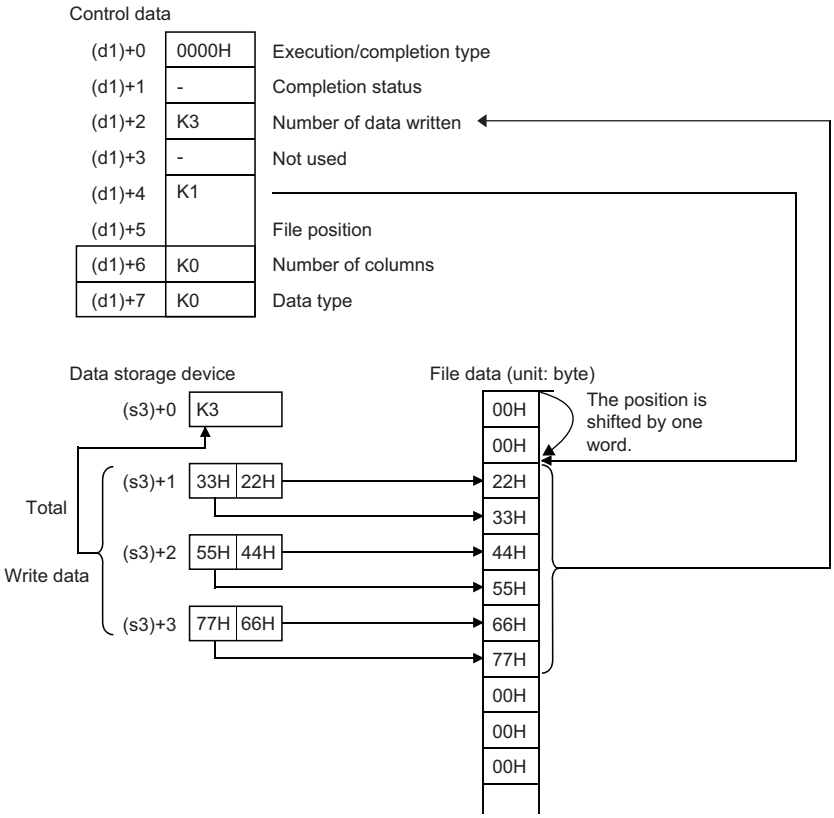
Execution/completion type (d1)+0		**Data type specification (d1)+7**	**Processing unit and setting range**		
			Number of request write data (s3)+0	**File position (d1)+4, (d1)+5**	**Number of data actually written (d1)+2**
Writing binary data	0000H: 16-bit binary data	0: Word	Word (1 to 65535)	Word (00000000H to 7FFFFFFFH, FFFFFFFFH)	Word
		1: Even number of bytes	Word (1 to 32767)	Byte (00000000H to FFFFFFFFH)	Byte
		2: Unit of the data type specified by the execution/completion type	Word (1 to 65535)	Word (00000000H to 7FFFFFFFH, FFFFFFFFH)	Word
		3: Odd number of bytes	Word (1 to 32767)	Byte (00000000H to FFFFFFFFH)	Byte
	0001H: 32-bit binary data	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Double word (1 to 32767)	Double word (00000000H to 3FFFFFFFH, FFFFFFFFH)	Double word
		3: Odd number of bytes	(Cannot be specified)		

Execution/completion type (d1)+0		Data type specification (d1)+7	Processing unit and setting range		
			Number of request write data (s3)+0	File position (d1)+4, (d1)+5	Number of data actually written (d1)+2
Writing data after converted to CSV format	0100H: Decimal (signed 16-bit data)	0: Word	Word (1 to 65535)	Head/end*3	Word
		1: Even number of bytes	Word (1 to 32767)	Head/end*3	Byte
		2: Unit of the data type specified by the execution/completion type	Word (1 to 65535)	Head/end*3	Word
		3: Odd number of bytes	Word (1 to 32767)	Head/end*3	Byte
	0101H: Decimal (unsigned 16-bit data)	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Word (1 to 65535)	Head/end*3	Word
		3: Odd number of bytes	(Cannot be specified)		
	0110H: Decimal (signed 32-bit data)	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Double word (1 to 32767)	Head/end*3	Double word
		3: Odd number of bytes	(Cannot be specified)		
	0111H: Decimal (unsigned 32-bit data)	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Double word (1 to 32767)	Head/end*3	Double word
		3: Odd number of bytes	(Cannot be specified)		
	0120H: Hexadecimal (16-bit data)	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Word (1 to 65535)	Head/end*3	Word
		3: Odd number of bytes	(Cannot be specified)		
	0121H: Hexadecimal (32-bit data)	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Double word (1 to 32767)	Head/end*3	Double word
		3: Odd number of bytes	(Cannot be specified)		
	0130H: String (ASCII data)	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Number of elements (1 to 1023)	Head/end*3	Number of elements
		3: Odd number of bytes	(Cannot be specified)		
	0140H: Floating point real number (single-precision real number)	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	Double word (1 to 32767)	Head/end*3	Double word
		3: Odd number of bytes	(Cannot be specified)		
	0141H: Floating point real number (double-precision real number)	0: Word 1: Even number of bytes	(Cannot be specified)		
		2: Unit of the data type specified by the execution/completion type	4 words (1 to 16383)	Head/end*3	4 words
		3: Odd number of bytes	(Cannot be specified)		

*3 The set value is from 00000000H to FFFFFFFEH (from the beginning of the file) or FFFFFFFFH (added to the end of the file).

■Writing binary data

- If the extension of the target file is omitted, the extension will be ".BIN".
- If a file that does not exist is specified, it will be created and data will be saved to the beginning of the file.
- When an existing file is specified, data will be saved to the beginning of the file. If the size of data exceeds the size of the existing area in the file during writing, the excess data is additionally stored.
- If the specified position exceeds the existing file size, 0 point of data is written and the processing completes successfully.
- If the media runs out of free space during additional saving of data, an error occurs. In this case, the data already added and saved successfully is held as is and as much remaining data as possible is added and saved before termination with an error.
- In 16-bit binary data write mode, when the number of request write data and file position are specified, data is written as follows.

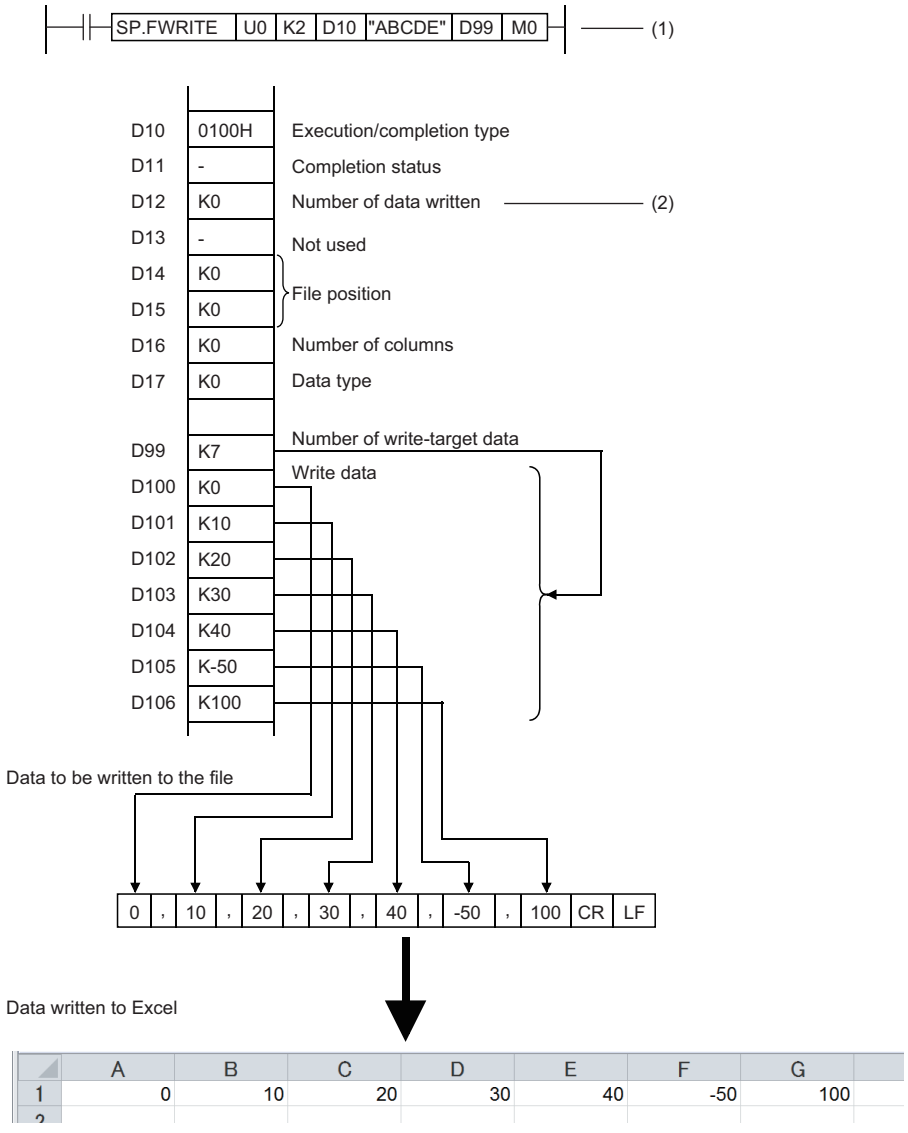


■Converting and writing data in CSV format

- If the extension is omitted, the extension will be ".CSV".
- When an existing file is specified, the following occurs.
  - When a value specified by (d1)+4 and (d1)+5 is other than FFFFFFFFH, data is saved to the file after deleting all the existing data in the file.
  - When a value specified by (d1)+4 and (d1)+5 is FFFFFFFFH, data is added and saved to the end of the file. If 1 is specified in (d1)+3, the return code at the end of the file will be converted to a comma and the data will be saved. However, if the end of the file is something other than a return code, the data will be saved from the end of the file without conversion to a comma.
- If a file that does not exist is specified, it will be created and data will be saved to the beginning of the file.
- If the media runs out of free space during additional saving of data, an error occurs. In this case, the data already added and saved successfully is held as is and as much remaining data as possible is added and saved before termination with an error.
- When the number of columns is set to 0, data is read as a single-row data in a CSV format file.

Ex.

The number of columns is set to 0 when writing data after conversion to the CSV format.



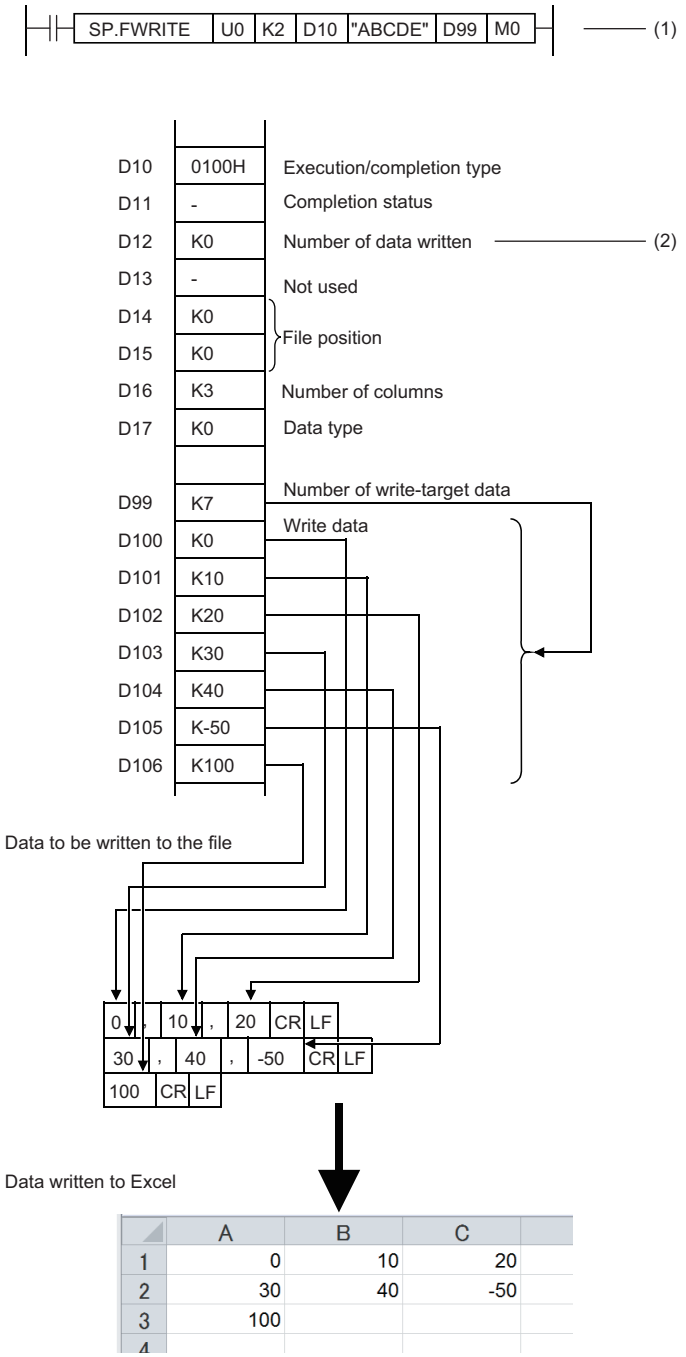
(1) Specified in units of words

(2) Same as the number of write data if case of normal completion

- When the specified number of columns is set to a value other than 0, a CSV format file is stored as the table with the specified number of columns.

Ex.

The number of columns is set to a value other than 0 when writing data after conversion to the CSV format.

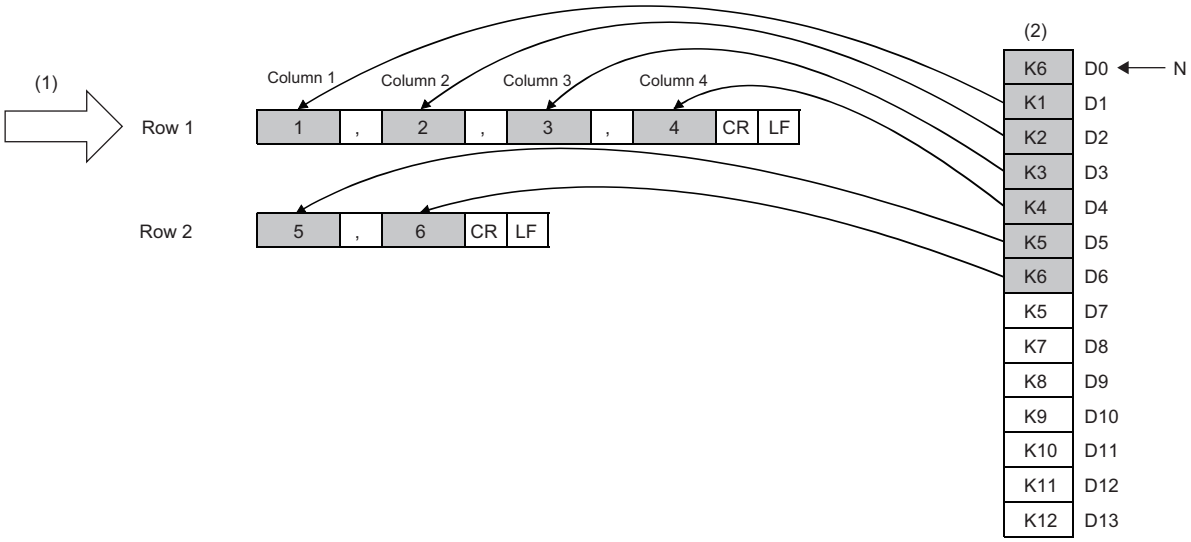


(1) Specified in units of words
(2) Same as the number of write data if case of normal completion

• The following figure shows how data is added.

[Specify the file to which data will be written.] (Even if the file exists, it is deleted and re-created.)

- Execution/completion type: Writing data after converted to CSV format (Decimal (signed 16-bit data))
- File position: 00000000H (from the beginning of the file)
- Number of columns specification: 4H
- Data type specification: Words
- Write start device: D0
- Number of request write data: 6H



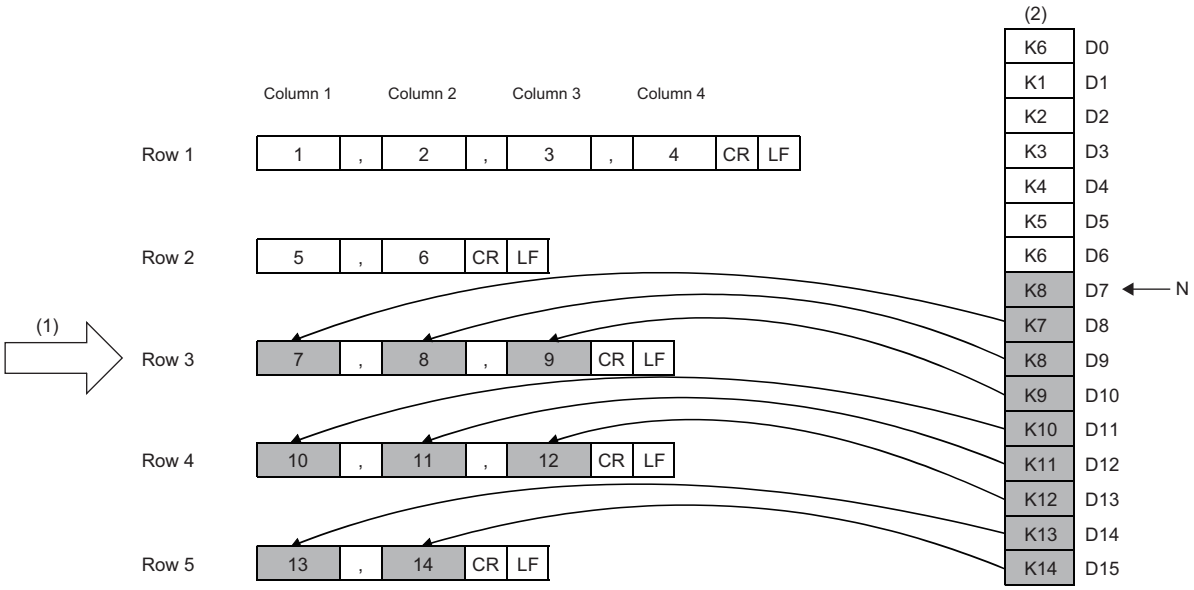
(1) Starting row

(2) Data in the device (write data)

N: Number of data

[Added to the end of the file]

- Execution/completion type: Writing data after converted to CSV format (Decimal (signed 16-bit data))
- Application setting area: 0H
- File position: FFFFFFFFH (added to the end of the file)
- Number of columns specification: 3H
- Data type specification: Words
- Write start device: D7
- Number of request write data: 8H



(1) Starting position

(2) Data in the device (write data)

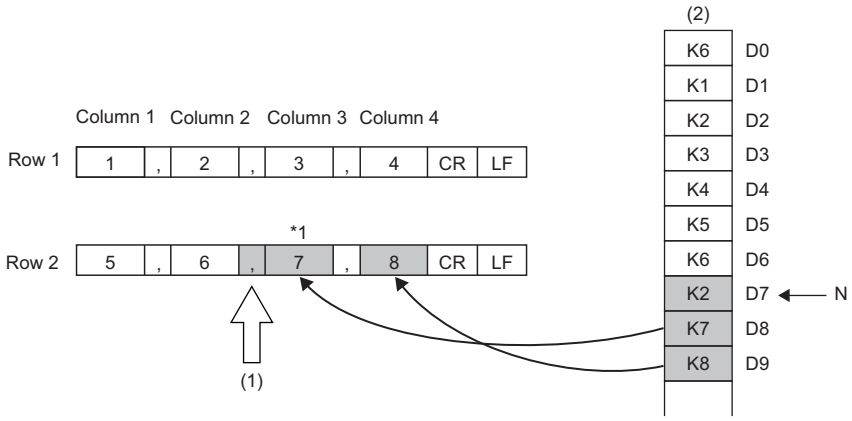
N: Number of data

Point

- An integral multiple of "Number of columns" should be specified for "Number of request write data". Otherwise, numbers of columns will be apart.
- The last data is always followed by a return code, so if the application setting area is 0, data is added starting from the beginning of a new line.
- When data is added to the end of a file, columns are shifted if "Number of columns" is changed from the previous writing.

[Added from the last line of the file]

- Execution/completion type: Writing data after converted to CSV format (Decimal (signed 16-bit data))
- Application setting area: 1H
- File position: FFFFFFFFH (added to the end of the file)
- Data type specification: Words
- Write start device: D7
- Number of request write data: 2H



(1) Starting position

(2) Data in the device (write data)

N: Number of data

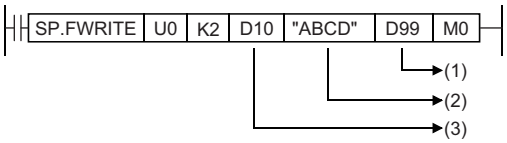
*1 If 1 is set in the application setting area, the return code at the end of the file will be converted to a comma, and data can be added from the last line.

• The following figures show an example of specifying "String (ASCII data)" for the execution/completion type.

Ex.

Writing data after converted to CSV format (string (ASCII data))

[Data to be written to a file]

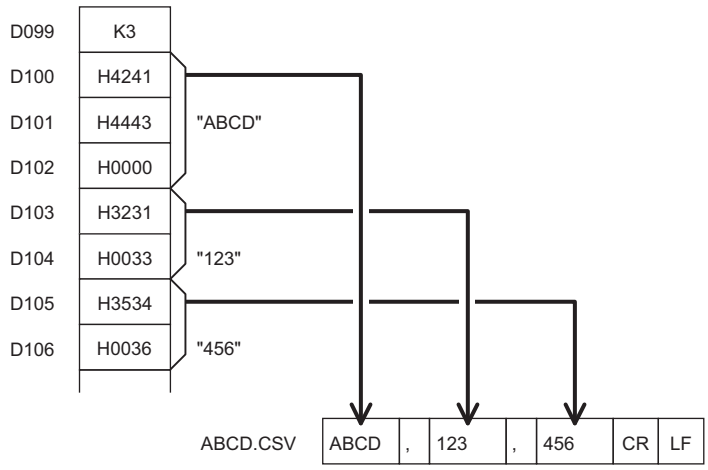


(1) Data to be written
(2) File name
(3) Control data

[Control data]

D10	H0130
D11	H0000
D12	-
D13	-
D14	K0
D15	
D16	K0
D17	K2

D10: Execution/completion type: String (ASCII data)
D11: Completion status
D12: Number of data actually written
D13: (Not used)
D14, D15: File positions
D16: Number of columns
D17: Data type specification



D99: Number of request write data
D100 to D102: String to be written to the 1st column of the 1st line
D103, D104: String to be written to the 2nd column of the 1st line
D105, D106: String to be written to the 3rd column of the 1st line

	A	B	C	D
1	ABCD	123	456	
2				

Point

- Set 00H (NULL) in the end of the string in an element. When the number of bytes of the string is even, set 0000H (two bytes of NULL) in the next one word.
- The maximum number of characters in one element is 1999. If this maximum number is exceeded and 00H (NULL) is not stored, characters of 2000th character and after are not written and the write processing shifts to the next element.
- A maximum of 1023 elements can be written in a single instruction execution.

- The following table lists values to set in (s3)+1 and later and data to be written in a CSV file when "0140H: Floating point real number (single-precision real number)" or "0141H: Floating point real number (double-precision real number)" is set to (d1)+0.

Execution/completion type ((d1)+0)	Value to set in the write data ((s3)+1 and later)	Data to be written in a CSV file
0140H: Floating point real number (single-precision real number)	Values within the range of: $-2^{128} < data \leq -2^{-126}$, 0, $2^{-126} \leq data < 2^{128}$	A value (0 to 7 digits in the decimal part) as given on the left is written in exponential format.
	Values other than above	0 is written. (Data cannot be converted.)
0141H: Floating point real number (double-precision real number)	Values within the range of: $-2^{1024} < data \leq -2^{-1022}$, 0, $2^{-1022} \leq data < 2^{1024}$	A value (0 to 15 digits in the decimal part) as given on the left is written in exponential format.
	Values other than above	0 is written. (Data cannot be converted.)

- The following shows how the file size (total number of bytes) is calculated when a CSV format file is written to the SD memory card.

[Total number of bytes] = [Total number of bytes excluding the last row] + [Number of bytes of the last row]
([Number of bytes of each row] = [Number of columns*2] +1+ [total number of bytes of all data values per line]*3)

*2 The specified number of columns applies to rows other than the last row. The number of columns of the last row is calculated as shown below because it may differ from the specified number of rows depending on the number of write data.

- The number of rows excluding the last row is calculated. (Number of rows excluding the last row = number of requested write data÷number of columns (remainders rounded down)
- The number of columns of the last row is calculated. (Number of columns of the last row = number of requested write data - (number of rows excluding the last row × number of columns))

*3 The following shows how the number of bytes of each data value is calculated.

Sign of data value	Number of bytes of each data value	Range of bytes	Example
Positive	Number of digits	1 to 5 (word specification) 1 to 3 (byte specification)	• 12345: 5 bytes • 67: 2 bytes
Negative	Number of digits + 1	2 to 6 (word specification) 2 to 4 (byte specification)	• -12345: 6 bytes • -67: 3 bytes

Precautions

- Do not execute the SP.FWRITE instruction in interrupt programs. Doing so may cause malfunction of the module.
- Since the SP.FWRITE instruction causes file write, the scan time may be extended when this instruction is executed.
- The SP.FWRITE instruction does not support character codes with BOM. To write to a CSV file with BOM that was created using Excel, use an application such as a text editor to change the character code to exclude BOM.

Operation error

Error code (SD0)	Description
2820H	The value specified by the number of request write data (s3)+0 is out of the range, or it exceeds the assignment range specified by (s3)+1 and later in the relevant device/label.
3285H	The drive specified by (s1) is not the one for the SD memory card.
	Any value that is set in the device specified by (d1) and later as control data is out of the range. ☞ Page 625 Control data (d1)
	The file name string specified by (s2) cannot be read. • The number of characters of the string in the file name specified exceeds the range. • An inhibited value is set. • The specified file name string ends with a delimiter. • The file name (not including the extension) of the specified file name string is 0 characters. ☞ Page 625 Control data (d1)
3297H	An invalid combination of (d1)+0 (Execution/completion type) and (d1)+7 (Data type specification) is specified. An invalid combination of (d1) (Execution/completion type), (d1)+3 (Write start position setting), and (d1)+4 (File position specification) is specified. ☞ Page 625 Control data (d1)

If the SP.FWRITE instruction completes with an error, an error code is stored in the device specified by (d1)+1. (Note that an error code is not stored if the instruction results in an operation error.)

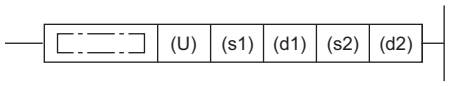
For the error code stored in (d1)+1, refer to the following.

☞ Page 660 Error codes generated for file operation instructions

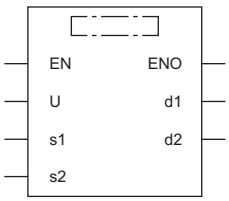
Deleting the specified file

SP.FDELETE

This instruction deletes the specified file or folder in an SD memory card.

Ladder	ST
	ENO:=SP_FDELETE(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.FDELETE	

Setting data

■Description, range, data type

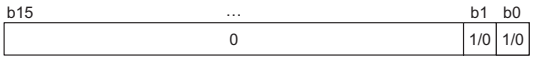
Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	Device name	ANY16
(s1)	Drive specification	2 (fixed)*1	Word	ANY16
(d1)	Start device where the control data is stored	☞ Page 639 Control data (d1)	Word	ANY16_ARRAY (Number of elements: 2)
(s2)	Start device where the file name or folder name is stored	—	Unicode string	ANYSTRING_DOUBLE
(d2)	Bit device that turns on upon completion of the processing	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only drive 2 (for the SD memory card) can be set.

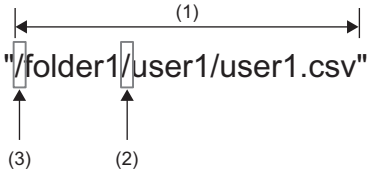
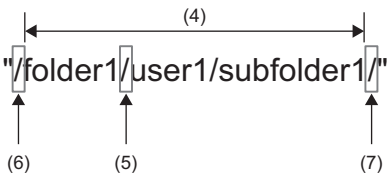
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(d2)	○	—	○	—	—	—	—	—	—	—	—	—

■Control data (d1)

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Application setting area	 b0: Target type setting Specify the type of data (file or folder) to delete. • 0: File • 1: Folder b1: Empty folder deletion setting Specify whether to delete only empty folders when deleting folders. • 0: Delete the folder even when it contains data. • 1: Delete the folder only when it is empty.	Refer to the "Description" column.	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code) (☞ Page 660 Error codes generated for file operation instructions)	—	System

■File name/folder name (s2)

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +□	File name/folder name	When specifying a file, specify the folder path where the file to be deleted is stored and the file name. When specifying a folder, specify the folder path of the folder to be deleted. • If the file name has an extension, do not omit it. • The file name, including the period and extension, must be within 64 characters. • The folder path and file name (including the period and extension) must be within 253 characters in total. • The folder path must be within 244 characters. (Delimiters are not included.) • Specify one character or more for the file name or folder path in addition to a delimiter. • When specifying a file, do not add a delimiter at the end of a string. • Do not put a period (one-byte) at the end of a string or directly before each delimiter. • The number of folder path hierarchies must be within 10 levels. ■When a file is specified  (1): Up to 253 characters (2): Use "/" or "\" as delimiters for the folder path and file. (3): Can be omitted. When it is omitted, (1) is up to 252 characters. ■When a folder is specified  (4): Up to 244 characters (5): Use "/" or "\" as delimiters for the folder path. (6): Can be omitted. When it is omitted, (4) is up to 243 characters. (7): Can be omitted.	Unicode string	User

Processing details

- This instruction deletes the file or folder specified by (s2) in the drive specified by (s1).
- SM753 (File being accessed) turns on while the SP.FDELETE instruction is being executed. While SM753 is on, the SP.FDELETE instruction cannot be executed. (If the instruction is executed, no processing is performed.)
- The processing completion bit device (d2) automatically turns on at the execution of the END instruction in the scan in which the completion of processing of the SP.FDELETE instruction is detected. The bit device (d2) turns off at the execution of the END instruction in the next scan.
- If the SP.FDELETE instruction completes with an error, the error completion device (d2)+1 turns on or off in synchronization with (d2).
- If an operation error is detected during the execution of the instruction, (d2) and (d2)+1 do not turn on.

Precautions

- Do not execute the SP.FDELETE instruction in interrupt programs. Doing so may cause malfunction of the module.
- Even though the operating status of the controller is switched from RUN to STOP during instruction execution, the controller continues the processing of the instruction.
- If the instruction is completed with an error during processing, the file or folder already deleted is not restored.
- When the size of a file to be deleted or the number of files to be deleted becomes larger, the instruction will take more time to complete.
- Do not access the file being processed by the SP.FDELETE instruction from other functions. (The file may be corrupt or an error may occur.)
- Do not operate files or folders being accessed from other functions.

Operation error

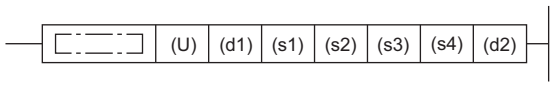
Error code (SD0)	Description
3285H	The drive specified by (s1) is not the one for the SD memory card.
	The file name/folder name string specified by (s2) cannot be read. • The specified file name string contains no character. • The specified file name string contains 254 characters or more. • The specified folder path contains 245 characters or more. • The specified folder path hierarchies contains 11 levels or more. • The specified file name string has a period (one-byte) at its end or directly before each delimiter. • When a file is specified, the file name string ends with a delimiter.
	A system folder ($MELPRJ$) which is directly under the root folder is in the folder path specified by (s2).

If the SP.FDELETE instruction completes with an error, an error code is stored in the device specified by (d1)+1. (Note that an error code is not stored if the instruction results in an operation error.)
For the error code stored in (d1)+1, refer to the following.
☞ Page 660 Error codes generated for file operation instructions

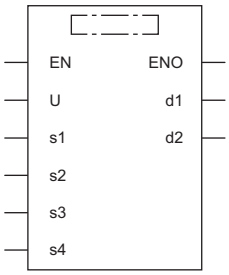
Copying the specified file

SP.FCOPY

This instruction copies the specified file or folder in an SD memory card. When a folder is specified, the specified folder is copied in its entirety, or all the files and subfolders in the specified folder are copied.

Ladder	ST
	ENO:=SP_FCOPY(EN,U,s1,s2,s3,s4,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.FCOPY	

Setting data

■Description, range, data type

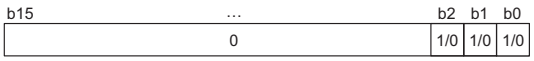
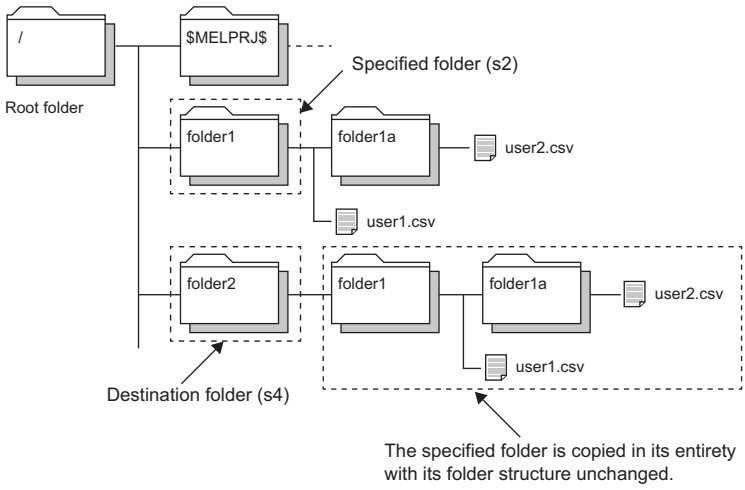
Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	Device name	ANY16
(d1)	Start device where the control data is stored	☞ Page 642 Control data (d1)	Word	ANY16_ARRAY (Number of elements: 2)
(s1)	Copy source drive	2 (fixed)*1	Word	ANY16
(s2)	Start device where the file name or folder name of the copy source is stored	—	Unicode string	ANYSTRING_DOUBLE
(s3)	Copy destination drive	2 (fixed)*1	Word	ANY16
(s4)	Start device for storing the copy destination folder path	—	Unicode string	ANYSTRING_DOUBLE
(d2)	Bit device that turns on upon completion of the processing	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

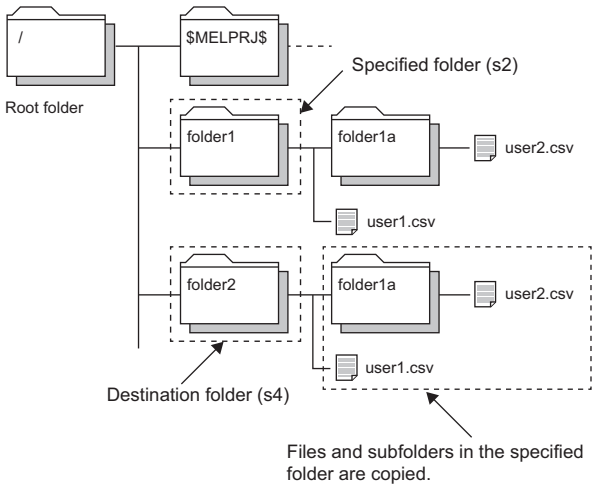
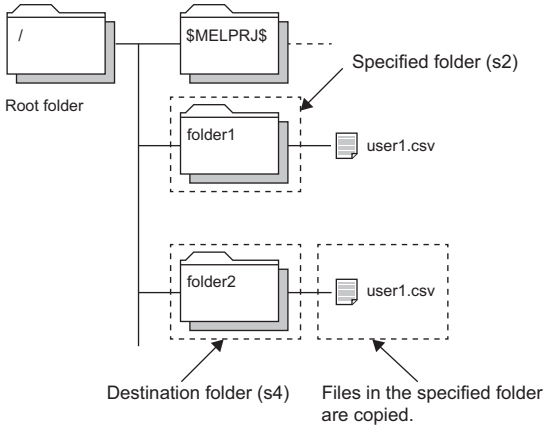
*1 Only drive 2 (for the SD memory card) can be set.

■Applicable devices

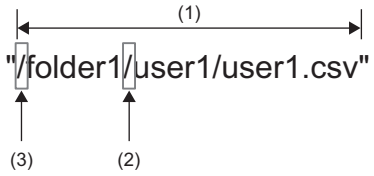
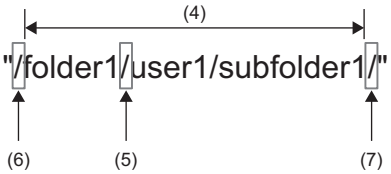
Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	○	—	○	—	—	—	—	○	○	—	—	—
(s4)	—	—	○	—	—	—	—	○	—	—	○	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—

■Control data (d1)

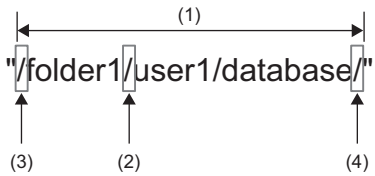
Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Application setting area	 ■b0, b2: Target type setting b0: Specify the type of data (file or folder) to copy. 0: File 1: Folder b2: Specify how the data is copied. (This bit is valid when b0 is set to 1 (Folder).) 0: Copy the folder in its entirety. 1: Copy all the files and subfolders in the folder. • Operation when b2 is set to 0 (Copy the folder in its entirety). The specified folder (including all files and subfolders in it) is copied in its entirety. 	Refer to the "Description" column.	User

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Application setting area	• Operation when b2 is set to 1 (Copy all the files and subfolders in the folder), and the specified folder includes files and a subfolder All the files and subfolders in the specified folder are copied.  • Operation when b2 is set to 1 (Copy all the files and subfolders in the folder), and the specified folder includes only files (no subfolder) All the files in the specified folder are copied.  ■b1: Overwrite setting Specify whether to overwrite a file or a folder if one with the same name as the copy source already exists in the copy destination. • 0: Do not overwrite the file or folder. • 1: Overwrite the file or folder. When b0 is set to 1 (Folder) or b1 is set to 0 (Do not overwrite the file or folder), the instruction skips copying a file or a folder having the same name with the one in the copy destination. (The instruction is completed and no error occurs.)	Refer to the "Description" column.	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code) (☞ Page 660 Error codes generated for file operation instructions)	—	System

■Copy source file name/folder name (s2)

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +☐	File name/folder name	When specifying a file, specify the folder path where the copy source file is stored and the file name. When specifying a folder, specify the copy source folder path. • If the file name has an extension, do not omit it. • The file name, including the period and extension, must be within 64 characters. • The folder path and file name (including the period and extension) must be within 253 characters in total. • The folder path must be within 244 characters. (Delimiters are not included.) • Specify one character or more for the file name or folder path in addition to a delimiter. • When specifying a file, do not add a delimiter at the end of a string. • Do not put a period (one-byte) at the end of a string or directly before each delimiter. • The number of folder path hierarchies must be within 10 levels. ■When a file is specified  (1): Up to 253 characters (2): Use "/" or "\" as delimiters for the folder path and file. (3): Can be omitted. When it is omitted, (1) is up to 252 characters. ■When a folder is specified  (4): Up to 244 characters (5): Use "/" or "\" as delimiters for the folder path. (6): Can be omitted. When it is omitted, (4) is up to 243 characters. (7): Can be omitted.	Unicode string	User

■Copy destination folder path (s4)

Operand: (s4)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +☐	Folder path	Specify the copy destination folder path. • The folder path must be within 244 characters. (Delimiters are not included.) • Specify one character or more for the folder path in addition to a delimiter. • Do not put a space (one-byte) at the end of a string or directly before each delimiter. • Do not put a period (one-byte) at the end of a string or directly before each delimiter. • The number of folder path hierarchies must be within 10 levels.  (1): Up to 244 characters (2): Use "/" or "\" as delimiters for the folder path. (3): Can be omitted. When it is omitted, (1) is up to 243 characters. (4): Can be omitted.	Unicode string	User

Processing details

- This instruction copies the file or folder specified by (s2) in the drive specified by (s1) to the folder specified by (s4) in the drive specified by (s3). When a folder is specified by (s2), a file and subfolder with the same names may exist in the copy destination. In this case, the instruction is not completed with an error even if the value of bit 1 (overwrite setting) of (d1) is 0. (Copying is skipped.)
- If the folder specified by (s4) does not exist in the copy destination, the folder is created automatically.
- SM753 (File being accessed) turns on while the SP.FCOPY instruction is being executed. While SM753 is on, the SP.FCOPY instruction cannot be executed. (If the instruction is executed, no processing is performed.)
- The processing completion bit device (d2) automatically turns on at the execution of the END instruction in the scan in which the completion of processing of the SP.FCOPY instruction is detected. The bit device (d2) turns off at the execution of the END instruction in the next scan.
- If the SP.FCOPY instruction completes with an error, the error completion device (d2)+1 turns on or off in synchronization with (d2).
- If an operation error is detected during the execution of the instruction, (d2) and (d2)+1 do not turn on.

Precautions

- Do not execute the SP.FCOPY instruction in an interrupt program. Doing so may cause malfunction of the module.
- Even though the operating status of the controller is switched from RUN to STOP during instruction execution, the controller continues the processing of the instruction.
- Ensure that the number of characters in a folder path or in the total of a folder path and a file name after copying does not exceed its limit. (The file may be inaccessible or an error may occur.)
- If the instruction is completed with an error during processing, the file or folder may remain under processing.
- When the size of a file to be copied or the number of files to be copied becomes larger, the instruction will take more time to complete.
- Do not access the file being processed by the SP.FCOPY instruction from other functions. (The file may be corrupt or an error may occur.)
- Do not operate files or folders being accessed from other functions.

Operation error

Error code (SD0)	Description
3285H	The drive specified by (s1) or (s3) is not the one for the SD memory card.
	The file name/folder name string specified by (s2) cannot be read. • The specified file name string contains no character. • The specified file name string contains 254 characters or more. • The specified folder path contains 245 characters or more. • The specified folder path hierarchies contains 11 levels or more. • When a file is specified, the file name string ends with a delimiter. • The specified file name string has a period (one-byte) at its end or directly before each delimiter.
	The folder path string specified by (s4) cannot be read. • The specified file path string contains no character. • The specified folder path string contains 245 characters or more. • The specified folder path hierarchies contains 11 levels or more. • The specified folder path string has a space (one-byte) at its end or directly before each delimiter. • The specified folder path string has a period (one-byte) at its end or directly before each delimiter.
	A system folder ($MELPRJ$) which is directly under the root folder is in the folder path specified by (s2) or (s4).

If the SP.FCOPY instruction completes with an error, an error code is stored in the device specified by (d1)+1. (Note that an error code is not stored if the instruction results in an operation error.)

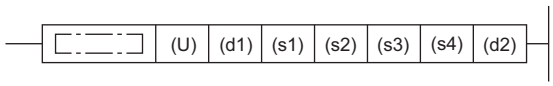
For the error code stored in (d1)+1, refer to the following.

☞ Page 660 Error codes generated for file operation instructions

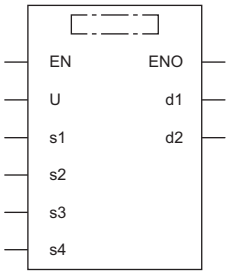
Moving the specified file

SP.FMOVE

This instruction moves the specified file or folder in an SD memory card. When a folder is specified, the specified folder is moved in its entirety, or all the files and subfolders in the specified folder are moved.

Ladder	ST
	ENO:=SP_FMOVE(EN,U,s1,s2,s3,s4,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.FMOVE	

Setting data

■Description, range, data type

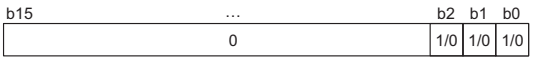
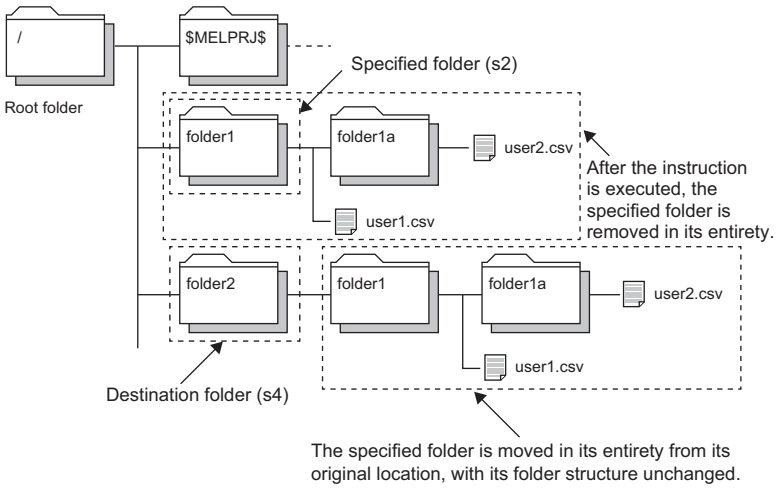
Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	Device name	ANY16
(d1)	Start device where the control data is stored	☞ Page 647 Control data (d1)	Word	ANY16_ARRAY (Number of elements: 2)
(s1)	Source drive	2 (fixed)*1	Word	ANY16
(s2)	Start device where the file name or folder name to move is stored	—	Unicode string	ANYSTRING_DOUBLE
(s3)	Destination drive	2 (fixed)*1	Word	ANY16
(s4)	Device for storing the destination folder path	—	Unicode string	ANYSTRING_DOUBLE
(d2)	Bit device that turns on upon completion of the processing	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

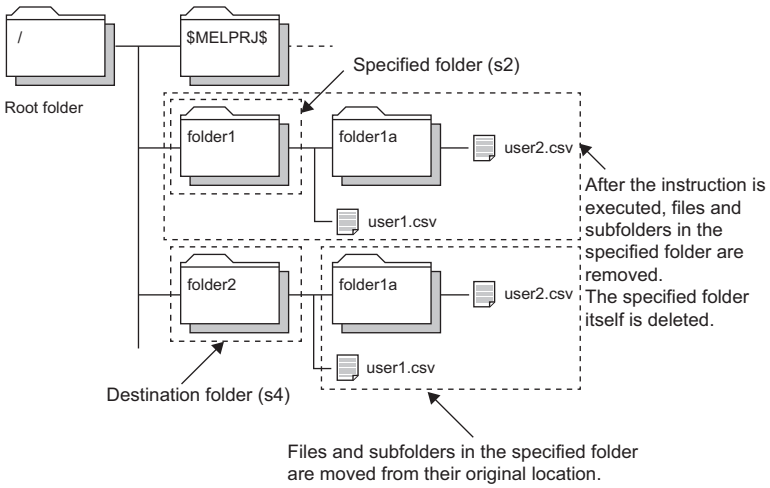
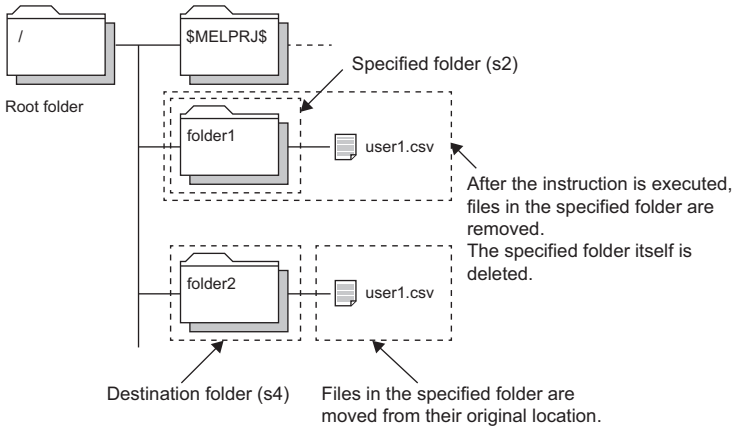
*1 Only drive 2 (for the SD memory card) can be set.

■Applicable devices

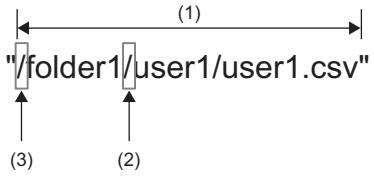
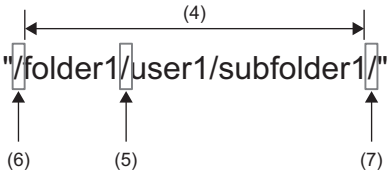
Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	○	—	○	—	—	—	—	○	○	—	—	—
(s4)	—	—	○	—	—	—	—	○	—	—	○	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—

■Control data (d1)

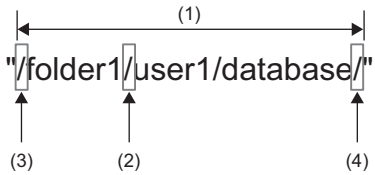
Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Application setting area	 ■b0, b2: Target type setting b0: Specify the type of data (file or folder) to move. 0: File 1: Folder b2: Specify how the data is moved. (This bit is valid when b0 is set to 1 (Folder).) 0: Move the folder in its entirety. 1: Move all the files and subfolders in the folder. • Operation when b2 is set to 0 (Move the folder in its entirety). The specified folder (including all files and subfolders in it) is moved in its entirety. 	Refer to the "Description" column.	User

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Application setting area	• Operation when b2 is set to 1 (Move all the files and subfolders in the folder), and the specified folder includes files and a subfolder All the files and subfolders in the specified folder are moved.  • Operation when b2 is set to 1 (Move all the files and subfolders in the folder), and the specified folder includes only files (no subfolder) All the files in the specified folder are moved.  ■b1: Overwrite setting Specify whether to overwrite a file or a folder if one with the same name as the source already exists in the destination. • 0: Do not overwrite the file or folder. • 1: Overwrite the file or folder.	Refer to the "Description" column.	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code) (☞ Page 660 Error codes generated for file operation instructions)	—	System

■File name/folder name (s2)

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +☐	File name/folder name	When specifying a file, specify the folder path where the file to move is stored and the file name. When specifying a folder, specify the folder path of the folder to move. • If the file name has an extension, do not omit it. • The file name, including the period and extension, must be within 64 characters. • The folder path and file name (including the period and extension) must be within 253 characters in total. • The folder path must be within 244 characters. (Delimiters are not included.) • Specify one character or more for the file name or folder path in addition to a delimiter. • When specifying a file, do not add a delimiter at the end of a string. • Do not put a period (one-byte) at the end of a string or directly before each delimiter. • The number of folder path hierarchies must be within 10 levels. ■When a file is specified  (1): Up to 253 characters (2): Use "/" or "\" as delimiters for the folder path and file. (3): Can be omitted. When it is omitted, (1) is up to 252 characters. ■When a folder is specified  (4): Up to 244 characters (5): Use "/" or "\" as delimiters for the folder path. (6): Can be omitted. When it is omitted, (4) is up to 243 characters. (7): Can be omitted.	Unicode string	User

■Destination folder path (s4)

Operand: (s4)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +☐	Folder path	Specify the destination folder path. • The folder path must be within 244 characters. (Delimiters are not included.) • Specify one character or more for the folder path in addition to a delimiter. • Do not put a space (one-byte) at the end of a string or directly before each delimiter. • Do not put a period (one-byte) at the end of a string or directly before each delimiter. • The number of folder path hierarchies must be within 10 levels.  (1): Up to 244 characters (2): Use "/" or "\" as delimiters for the folder path and file. (3): Can be omitted. When it is omitted, (1) is up to 243 characters. (4): Can be omitted.	Unicode string	User

Processing details

- This instruction moves the file or folder specified by (s2) in the drive specified by (s1) to the folder specified by (s4) in the drive specified by (s3). If the folder specified by (s4) does not exist in the destination, the folder is created automatically.
- SM753 (File being accessed) turns on while the SP.FMOVE instruction is being executed. While SM753 is on, the SP.FMOVE instruction cannot be executed. (If the instruction is executed, no processing is performed.)
- The processing completion bit device (d2) automatically turns on at the execution of the END instruction in the scan in which the completion of processing of the SP.FMOVE instruction is detected. The bit device (d2) turns off at the execution of the END instruction in the next scan.
- If the SP.FMOVE instruction completes with an error, the error completion device (d2)+1 turns on or off in synchronization with (d2).
- If an operation error is detected during the execution of the instruction, (d2) and (d2)+1 do not turn on.

Precautions

- Do not execute the SP.FMOVE instruction in an interrupt program. Doing so may cause malfunction of the module.
- Even though the operating status of the controller is switched from RUN to STOP during instruction execution, the controller continues the processing of the instruction.
- Ensure that the number of characters in a folder path or in the total of a folder path and a file name after moving does not exceed its limit. (The file may be corrupt or an error may occur.)
- If overwriting occurs, the time to complete the instruction may be extended.
- When overwriting occurs, ensure a free space of the same size as the operation target file.
- If the instruction is completed with an error during processing, the file or folder may remain under processing.
- If the size of a file to be moved or the number of files to be moved becomes larger due to overwriting, the instruction will take more time to complete.
- Do not access the file being processed by the SP.FMOVE instruction from other functions. (The file may be inaccessible or an error may occur.)
- Do not operate files or folders being accessed from other functions.

Operation error

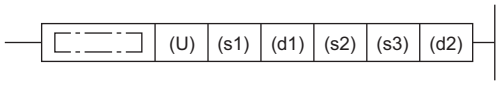
Error code (SD0)	Description
3285H	The drive specified by (s1) is not the one for the SD memory card.
	The file name/folder name string specified by (s2) cannot be read. • The specified file name string contains no character. • The specified file name string contains 254 characters or more. • The specified folder path contains 245 characters or more. • The specified folder path hierarchies contains 11 levels or more. • When a file is specified, the file name string ends with a delimiter. • The specified file name string has a period (one-byte) at its end or directly before each delimiter.
	The folder path string specified by (s4) cannot be read. • The specified file path string contains no character. • The specified folder path string contains 245 characters or more. • The specified folder path hierarchies contains 11 levels or more. • The specified folder path string has a space (one-byte) at its end or directly before each delimiter. • The specified folder path string has a period (one-byte) at its end or directly before each delimiter.
	A system folder ($MELPRJ$) which is directly under the root folder is in the folder path specified by (s2) or (s4).

If the SP.FMOVE instruction completes with an error, an error code is stored in the device specified by (d1)+1. (Note that an error code is not stored if the instruction results in an operation error.)
For the error code stored in (d1)+1, refer to the following.
☞ Page 660 Error codes generated for file operation instructions

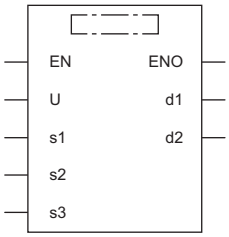
Renaming the specified file

SP.FRENAME

This instruction renames the specified file or folder in an SD memory card.

Ladder	ST
	ENO:=SP_FRENAME(EN,U,s1,s2,s3,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.FRENAME	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	Device name	ANY16
(s1)	Drive specification	2 (fixed)*1	Word	ANY16
(d1)	Start device where the control data is stored	☞ Page 652 Control data (d1)	Word	ANY16_ARRAY (Number of elements: 2)
(s2)	Start device where the file name or folder name to be changed is stored	—	Unicode string	ANYSTRING_DOUBLE
(s3)	Start device where the new file name/folder name after the change is stored	—	Unicode string	ANYSTRING_DOUBLE
(d2)	Bit device that turns on upon completion of the processing	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only drive 2 (for the SD memory card) can be set.

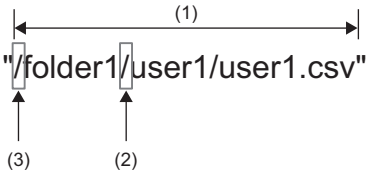
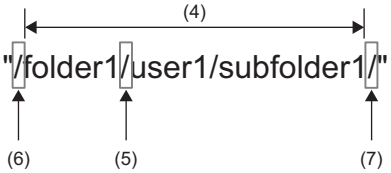
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	○	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—

■Control data (d1)

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Application setting area	b15 … b0 0 | 1/0 b0: Target type setting Specify the type of data (file or folder) to delete. • 0: File • 1: Folder	Refer to the "Description" column.	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code) (☞ Page 660 Error codes generated for file operation instructions)	—	System

■File name/folder name to be changed (s2)

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +☐	File name/folder name	When specifying a file, specify the folder path where the file to be renamed is stored and the file name. When specifying a folder, specify the folder path of the folder to be renamed. • If the file name has an extension, do not omit it. • The file name, including the period and extension, must be within 64 characters. • The folder path and file name (including the period and extension) must be within 253 characters in total. • The folder path must be within 244 characters. (Delimiters are not included.) • Specify one character or more for the file name or folder path in addition to a delimiter. • When specifying a file, do not add a delimiter at the end of a string. • Do not put a period (one-byte) at the end of a string or directly before each delimiter. • The number of folder path hierarchies must be within 10 levels. ■When a file is specified  (1): Up to 253 characters (2): Use "/" or "\" as delimiters for the folder path and file. (3): Can be omitted. When it is omitted, (1) is up to 252 characters. ■When a folder is specified  (4): Up to 244 characters (5): Use "/" or "\" as delimiters for the folder path. (6): Can be omitted. When it is omitted, (4) is up to 243 characters. (7): Can be omitted.	Unicode string	User

■File name/folder name after the change (s3)

Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +☐	File name/folder name	Specify the string of the file name/folder name after the change. • Do not specify a folder path. • The file name (including the period and extension) must be within 64 characters. • The folder name must be within 243 characters. • Specify one character or more for the file name or folder name. • Do not specify a delimiter. • Do not put a period (one-byte) at the end of a string.	Unicode string	User

Processing details

- This instruction rename the file or folder specified by (s2) in the drive specified by (s1) to the file name or folder name specified by (s3). If the file name or folder name specified by (s3) already exists, the instruction is completed with an error.
- SM753 (File being accessed) turns on while the SP.FRENAME instruction is being executed. While SM753 is on, the SP.FRENAME instruction cannot be executed. (If the instruction is executed, no processing is performed.)
- The processing completion bit device (d2) automatically turns on at the execution of the END instruction in the scan in which the completion of processing of the SP.FRENAME instruction is detected. The bit device (d2) turns off at the execution of the END instruction in the next scan.
- If the SP.FRENAME instruction completes with an error, the error completion device (d2)+1 turns on or off in synchronization with (d2).
- If an operation error is detected during the execution of the instruction, (d2) and (d2)+1 do not turn on.

Precautions

- Do not execute the SP.FRENAME instruction in an interrupt program. Doing so may cause malfunction of the module.
- Even though the operating status of the controller is switched from RUN to STOP during instruction execution, the controller continues the processing of the instruction.
- Ensure that the number of characters in a folder path or in the total of a folder path and a file name after changing does not exceed its limit. (The file may be inaccessible or an error may occur.)
- Do not access the file being processed by the SP.FRENAME instruction from other functions. (The file may be corrupt or an error may occur.)
- Do not operate files or folders being accessed from other functions.

Operation error

Error code (SD0)	Description
3285H	The drive specified by (s1) is not the one for the SD memory card.
	The file name/folder name string specified by (s2) cannot be read. • The specified file name string contains no character. • The specified file name string contains 254 characters or more. • The specified folder path contains 245 characters or more. • The specified folder path hierarchies contains 11 levels or more. • When a file is specified, the file name string ends with a delimiter. • The specified file name string has a period (one-byte) at its end or directly before each delimiter.
	The file name/folder name string specified by (s3) cannot be read. • The specified file name/folder name string contains no character. • The specified file name string contains 253 characters or more. • The specified folder name string contains 244 characters or more. • The specified file name string contains a delimiter. • The specified file name string has a period (one-byte) at its end.
	A system folder ($MELPRJ$) which is directly under the root folder is in the folder path specified by (s2).
	$MELPRJ$ is specified for the file name/folder name specified by (s3).

If the SP.FRENAME instruction completes with an error, an error code is stored in the device specified by (d1)+1. (Note that an error code is not stored if the instruction results in an operation error.)

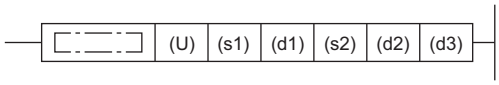
For the error code stored in (d1)+1, refer to the following.

☞ Page 660 Error codes generated for file operation instructions

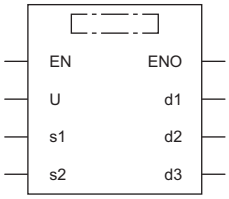
Acquiring the status of the specified file

SP.FSTATUS

This instruction acquires the status of the specified file or folder in an SD memory card.

Ladder	ST
	ENO:=SP_FSTATUS(EN,U,s1,s2,d1,d2,d3);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.FSTATUS	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	Device name	ANY16
(s1)	Drive specification	2 (fixed)*1	Word	ANY16
(d1)	Start device where the control data is stored	☞ Page 656 Control data (d1)	Word	ANY16_ARRAY (Number of elements: 2)
(s2)	Start device where the file name or folder name is stored	—	Unicode string	ANYSTRING_DOUBLE
(d2)	Start device for storing the file status	—	Word	ANY16_ARRAY (Number of elements: 10)
(d3)	Bit device that turns on upon completion of the processing	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only drive 2 (for the SD memory card) can be set.

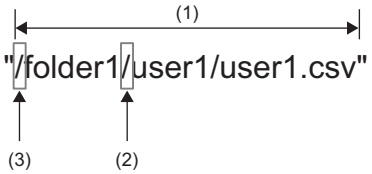
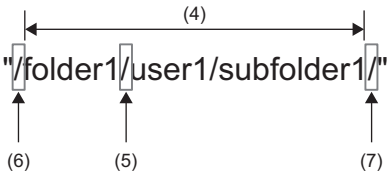
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(d3)	○	—	○	—	—	—	—	○	—	—	—	—

■Control data (d1)

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Application setting area	b15 … b0 0 | 1/0 b0: Target type setting Specify the type of data (file or folder) to acquire the status. • 0: File • 1: Folder	Refer to the "Description" column.	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code) (☞ Page 660 Error codes generated for file operation instructions)	—	System

■File name/folder name (s2)

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +☐	File name/folder name	When specifying a file, specify the folder path where the file whose status is to be acquired is stored and the file name. When specifying a folder, specify the folder path of the folder whose status is to be acquired. • If the file name has an extension, do not omit it. • The file name, including the period and extension, must be within 64 characters. • The folder path and file name (including the period and extension) must be within 253 characters in total. • The folder path must be within 244 characters. (Delimiters are not included.) • Specify one character or more for the file name or folder path in addition to a delimiter. • When specifying a file, do not add a delimiter at the end of a string. • Do not put a period (one-byte) at the end of a string or directly before each delimiter. • The number of folder path hierarchies must be within 10 levels. ■When a file is specified  (1): Up to 253 characters (2): Use "/" or "\" as delimiters for the folder path and file. (3): Can be omitted. When it is omitted, (1) is up to 252 characters. ■When a folder is specified  (4): Up to 244 characters (5): Use "/" or "\" as delimiters for the folder path. (6): Can be omitted. When it is omitted, (4) is up to 243 characters. (7): Can be omitted.	Unicode string	User

■File status (d2)

Operand: (d2)			
Device	**Item**	**Range**	**Set by**
+0	File attribute ■When a file is specified by (s2) bit0: Turns on for a read-only file. bit1 to bit4: Reserved (fixed to 0) bit5: Reserved (fixed to 1) bit6 to bit15: Reserved (fixed to 0) ■When a folder is specified by (s2) bit0 to bit3: Reserved (fixed to 0) bit4: Turns on for a folder. bit5 to bit15: Reserved (fixed to 0)	Refer to the "Item" column.	System
+1	Reserved	0	
+2 to +3	■When a file is specified by (s2) File size (in units of bytes) ■When a folder is specified by (s2) Fixed to 0	0 to 4294967294*1	
+4	Last update date/time: Year	0, 1980 to 2079*1	
+5	Last update date/time: Month	0 to 12	
+6	Last update date/time: Day	0 to 31	
+7	Last update date/time: Hour	0 to 23	
+8	Last update date/time: Minute	0 to 59	
+9	Last update date/time: Second		

*1 When the file is accessed in an environment (OS) other than the controller, the range of values to be acquired depends on the environment (OS).

Processing details

- This function acquires the status of the file or folder specified by (s2) in the drive specified by (s1) and stores it in the device (d2) and after. When a folder is specified by (s2), 0 is stored in (d2)+2 to (d2)+3.
- SM753 (File being accessed) turns on while the SP.FSTATUS instruction is being executed. While SM753 is on, the SP.FSTATUS instruction cannot be executed. (If the instruction is executed, no processing is performed.)
- The processing completion bit device (d3) automatically turns on at the execution of the END instruction in the scan in which the completion of processing of the SP.FSTATUS instruction is detected. The bit device (d3) turns off at the execution of the END instruction in the next scan.
- If the SP.FSTATUS instruction completes with an error, the error completion device (d3)+1 turns on or off in synchronization with (d3).
- If an operation error is detected during the execution of the instruction, the processing completion (d3) and the error completion ((d3)+1) do not turn on.

Precautions

- Do not execute the SP.FSTATUS instruction in an interrupt program. Doing so may cause malfunction of the module.
- Even though the operating status of the controller is switched from RUN to STOP during instruction execution, the controller continues the processing of the instruction.

Operation error

Error code (SD0)	Description
3285H	The drive specified by (s1) is not the one for the SD memory card.
	The file name/folder name string specified by (s2) cannot be read. • The specified file name string contains no character. • The specified file name string contains 254 characters or more. • The specified folder path contains 245 characters or more. • The specified folder path hierarchies contains 11 levels or more. • When a file is specified, the file name string ends with a delimiter. • The specified file name string has a period (one-byte) at its end or directly before each delimiter.

If the SP.FSTATUS instruction completes with an error, an error code is stored in the device specified by (d1)+1. (Note that an error code is not stored if the instruction results in an operation error.)

For the error code stored in (d1)+1, refer to the following.

☞ Page 660 Error codes generated for file operation instructions

Error codes generated for file operation instructions

The following table lists the error codes that could be stored in the completion status of file operation instructions.

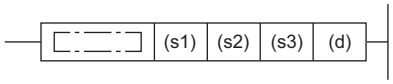
Error code	Description	Action
8000H	SM606 (SD memory card forced disable instruction) is on.	Turn off SM606 to cancel the SD memory card forced disable status.
	No SD memory card is inserted.	Insert an SD memory card.
	The SD memory card is not mounted.	Mount the SD memory card.
8001H	An access to the SD memory card has failed.	• Check that the file name string are correctly specified. • Check that no other functions are accessing the file. • Take measures to reduce noise. • Reset the controller and RUN it again. If the same error code is displayed again, the possible cause is a hardware failure of the SD memory card. Please consult your local Mitsubishi representative.
	The SD memory card is write-protected.	Unlock the write protect switch of the SD memory card.
	The file is set for read only.	Clear the read-only setting.
	The size of the file to be stored in the SD memory card exceeds the free space or the maximum capacity of the SD memory card.	• Delete unnecessary files in the SD memory card to secure free space. • Adjust the file size so that it is equal to or less than its limit.
	The file name string contains an invalid character.	Check that the file name string are correctly specified.
8002H	The specified file or folder does not exist.	• Check that the specified file or folder exists. • Check that the specified folder path exists.
	The free space in the SD memory card is insufficient.	Delete unnecessary files in the SD memory card to secure free space.
8003H	The total number of data read from the file has exceeded (d1)+3 (Maximum number of read data).	Adjust (d1)+2 (Number of read-target data) or (d1)+3 (Maximum number of read data).
8004H	The folder to be deleted is not empty.	Check the setting of the control data.
8005H	A file or folder with the same name exists.	Change the name of the file or folder.
8006H	• A folder whose folder path will exceed 244 characters after instruction execution exists. • A file whose file path will exceed 253 characters after instruction execution exists.	Ensure that the folder path or file path after instruction execution will not exceed the maximum number of characters.
	The following folder/subfolder is specified for the copy or move destination. • Same folder/subfolder as the copy source • Same folder/subfolder as the move source	Change the copy or move destination path.

14.5 Data Control Instructions

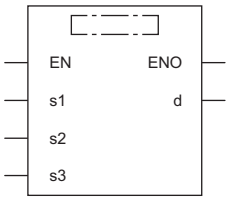
Upper and lower limit control of 16-bit binary data

LIMIT(P)(_U)

These instructions control the output value depending on whether the specified 16-bit binary bit value is within the upper and lower limits.

Ladder	ST*1	
	ENO:=LIMITP(EN,s1,s2,s3,d);	ENO:=LIMITP_U(EN,s1,s2,s3,d);

FBD/LD*1



*1 The LIMIT and LIMIT_U instructions do not support the ST and FBD/LD. Use the standard function, LIMIT.
☞ Page 1318 LIMIT(_E)

■Execution condition

Instruction	Execution condition
LIMIT LIMIT_U	
LIMITP LIMITP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	LIMIT(P)	Lower limit value (minimum output threshold value)	-32768 to 32767	16-bit signed binary	ANY16_S
	LIMIT(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	LIMIT(P)	Upper limit value (maximum output threshold value)	-32768 to 32767	16-bit signed binary	ANY16_S
	LIMIT(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s3)	LIMIT(P)	Input value to be controlled with the upper and lower limits	-32768 to 32767	16-bit signed binary	ANY16_S
	LIMIT(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	LIMIT(P)	Device for storing the output value controlled by upper/lower limit control	—	16-bit signed binary	ANY16_S
	LIMIT(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

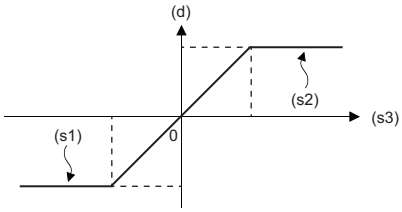
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(s3)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions control the output value to be stored in the device specified by (d) by checking the input value (16-bit binary data) in the device specified by (s3) with the upper and lower limit values specified by (s1) and (s2). The output value is controlled as follows.

Condition	Output value
Lower limit value (s1) > Input value (s3)	Lower limit value (s1)
Upper limit value (s2) < Input value (s3)	Upper limit value (s2)
Lower limit value (s1) ≤ Input value (s3) ≤ Upper limit value (s2)	Input value (s3)



- To control the input value only with the upper limit, set the minimum value within the setting range in (s1).
- To control the input value only with the lower limit, set the maximum value within the setting range in (s2).

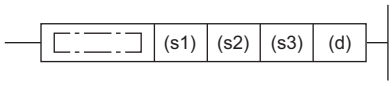
Operation error

Error code (SD0)	Description
3285H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

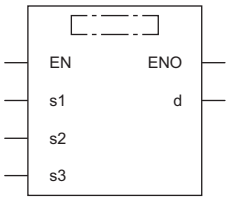
Upper and lower limit control of 32-bit binary data

DLIMIT(P)(_U)

These instructions control the output value depending on whether the specified 32-bit binary bit value is within the upper and lower limits.

Ladder	ST*1	
	ENO:=DLIMITP(EN,s1,s2,s3,d);	ENO:=DLIMITP_U(EN,s1,s2,s3,d);

FBD/LD*1



*1 The DLIMIT and DLIMIT_U instructions do not support the ST and FBD/LD. Use the standard function, LIMIT.
☞ Page 1318 LIMIT(_E)

■Execution condition

Instruction	Execution condition
DLIMIT DLIMIT_U	
DLIMITP DLIMITP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DLIMIT(P)	Device for storing the lower limit value (minimum output threshold value)	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DLIMIT(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	DLIMIT(P)	Device for storing the upper limit value (maximum output threshold value)	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DLIMIT(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s3)	DLIMIT(P)	Device for storing the input value controlled by upper/lower limit control	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DLIMIT(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	DLIMIT(P)	Start device for storing the controlled output value	—	32-bit signed binary	ANY32_S
	DLIMIT(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

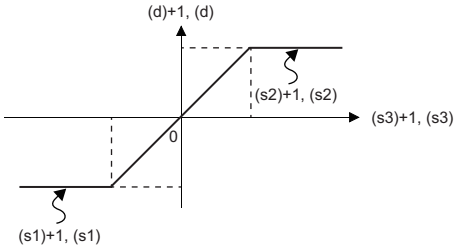
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(s3)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions control the output value to be stored in the device specified by (d) by checking the input value (32-bit binary data) in the device specified by (s3) with the upper and lower limit values specified by (s1) and (s2). The output value is controlled as follows.

Condition	Output value
Lower limit value ((s1), (s1)+1) > Input value ((s3), (s3)+1)	Lower limit value ((s1), (s1)+1)
Lower limit value ((s2), (s2)+1) < Input value ((s3), (s3)+1)	Upper limit value ((s2), (s2)+1)
Lower limit value ((s1), (s1)+1) ≤Input value ((s3), (s3)+1) ≤ Upper limit value ((s2), (s2)+1)	Input value ((s3), (s3)+1)



- To control the input value only with the upper limit, set the minimum value within the setting range in (s1).
- To control the input value only with the lower limit, set the maximum value within the setting range in (s2).

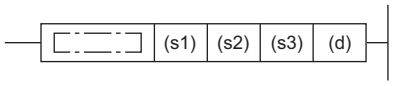
Operation error

Error code (SD0)	Description
3285H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

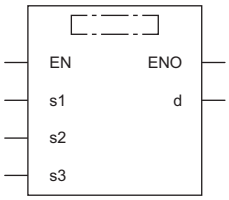
Dead band control of 16-bit binary data

BAND(P)(_U)

These instructions control the output value depending on whether the specified 16-bit binary bit value is within the upper and lower limits of the dead band.

Ladder	ST	
	ENO:=BAND(EN,s1,s2,s3,d); ENO:=BANDP(EN,s1,s2,s3,d);	ENO:=BAND_U(EN,s1,s2,s3,d); ENO:=BANDP_U(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BAND BAND_U	
BANDP BANDP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	BAND(P)	Lower limit of dead band (no-output band)	-32768 to 32767	16-bit signed binary	ANY16_S
	BAND(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	BAND(P)	Upper limit of dead band (no-output band)	-32768 to 32767	16-bit signed binary	ANY16_S
	BAND(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s3)	BAND(P)	Input value to be controlled by dead band control	-32768 to 32767	16-bit signed binary	ANY16_S
	BAND(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	BAND(P)	Device for storing the output value controlled by dead band control	—	16-bit signed binary	ANY16_S
	BAND(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

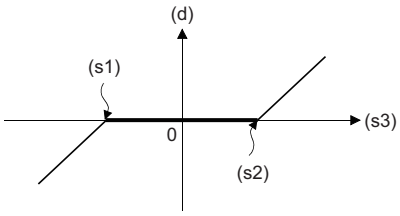
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(s3)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions control the output value to be stored in the device specified by (d) by checking whether the input value (16-bit binary data) in the device specified by (s3) is within range of the upper and lower limits of the dead band in the devices specified by (s1) and (s2). The output value is controlled as follows.

Condition	Output value
Dead band lower limit value (s1) > input value (s3)	Input value (s3) - dead band lower limit value (s1)
Dead band upper limit value (s2) < input value (s3)	Input value (s3) - dead band upper limit value (s2)
Dead band lower limit value (s1) ≤ input value (s3) ≤ dead band upper limit (s2)	0



- The following example shows the case where the operation result of the BAND(P) instruction is out of the range from -32768 to 32767.

Ex.

When (s1) is 10 and (s3) is -32768, output value is -32768-10 = 8000H-000AH = 7FF6H = 32758.

- The following example shows the case when the operation result of the BAND(P)_U instruction is out of the range from 0 to 65535.

Ex.

When (s1) is 100 and (s3) is 50, output value is 50-100 = 0032H-0064H = FFCEH = 65486.

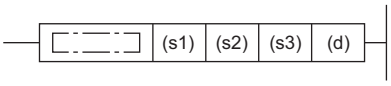
Operation error

Error code (SD0)	Description
3285H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

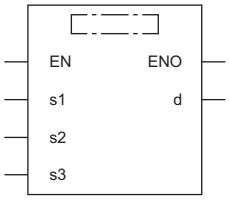
Dead band control of 32-bit binary data

DBAND(P)(_U)

These instructions control the output value depending on whether the specified 32-bit binary bit value is within the upper and lower limits of the dead band.

Ladder	ST	
	ENO:=DBAND(EN,s1,s2,s3,d); ENO:=DBANDP(EN,s1,s2,s3,d);	ENO:=DBAND_U(EN,s1,s2,s3,d); ENO:=DBANDP_U(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBAND DBAND_U	
DBANDP DBANDP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DBAND(P)	Device for storing the lower limit value of dead band (no-output band)	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DBAND(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	DBAND(P)	Device for storing the upper limit value of dead band (no-output band)	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DBAND(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s3)	DBAND(P)	Device for storing the input value controlled by dead band control	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DBAND(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	DBAND(P)	Start device for storing the output value controlled by dead band control	—	32-bit signed binary	ANY32_S
	DBAND(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

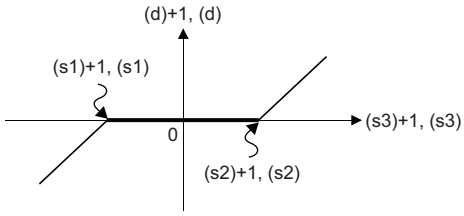
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(s3)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions control the output value to be stored in the device specified by (d) by checking whether the input value (32-bit binary data) in the device specified by (s3) is within range of the upper and lower limits of the dead band in the devices specified by (s1) and (s2). The output value is controlled as follows.

Condition	Output value
Dead band lower limit value ((s1), (s1)+1) > Input value ((s3), (s3)+1)	Input value ((s3), (s3)+1) - Dead band lower limit value ((s1), (s1)+1)
Dead band upper limit value ((s2), (s2)+1) < Input value ((s3), (s3)+1)	Input value ((s3), (s3)+1) - Dead band upper limit value ((s2), (s2)+1)
Dead band lower limit value ((s1), (s1)+1) ≤ Input value ((s3), (s3)+1) ≤ Dead band upper limit ((s2), (s2)+1)	0



- The following example shows the case when the operation result of the DBAND(P) instruction is out of the range from -2147483648 to 2147483647.

Ex.

When ((s1), (s1)+1) is 1000 and ((s3), (s3)+1) is -2147483648, output value is -2147483648-1000 = 80000000H-000003E8H = 7FFFFC18H = 2147482648.

- The following example shows the case when the operation result of the DBAND(P)_U instruction is out of the range from 0 to 4294967295.

Ex.

When ((s1), (s1)+1) is 100 and ((s3), (s3)+1) is 50, output value is 50-100 = 00000032H-00000064H = FFFFFFCEH = 4294967246.

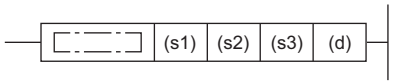
Operation error

Error code (SD0)	Description
3285H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

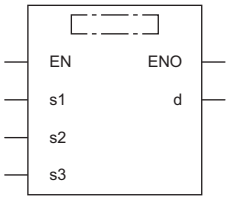
Zone control of 16-bit binary data

ZONE(P)(_U)

These instructions add a bias value to the specified input value (16-bit binary).

Ladder	ST	
	ENO:=ZONE(EN,s1,s2,s3,d); ENO:=ZONEP(EN,s1,s2,s3,d);	ENO:=ZONE_U(EN,s1,s2,s3,d); ENO:=ZONEP_U(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ZONE ZONE_U	
ZONEP ZONEP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	ZONE(P)	Negative bias value to be added to the input value	-32768 to 32767	16-bit signed binary	ANY16_S
	ZONE(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	ZONE(P)	Positive bias value to be added to the input value	-32768 to 32767	16-bit signed binary	ANY16_S
	ZONE(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s3)	ZONE(P)	Input value used for zone control	-32768 to 32767	16-bit signed binary	ANY16_S
	ZONE(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	ZONE(P)	Device for storing the output value controlled by zone control	—	16-bit signed binary	ANY16_S
	ZONE(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

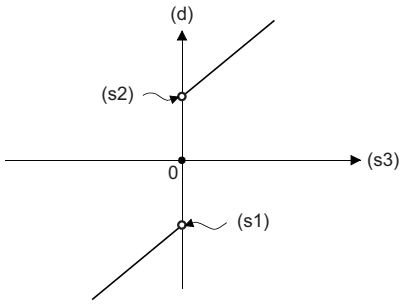
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(s3)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions add the bias value specified by (s1) or (s2) to the input value (16-bit binary) specified by (s3), and store the result in the device number specified by (d). The bias value is controlled as follows.

Condition	Output value
Input value (s3) < 0	Input value (s3) + negative bias value (s1)
Input value (s3) = 0	0
Input value (s3) > 0	Input value (s3) + positive bias value (s2)



- The following example shows the case where the operation result of the ZONE(P) instruction is out of the range from -32768 to 32767.

Ex.

When (s1) is -100 and (s3) is -32768, output value is -32768+(-100) = 8000H-FF9CH = 7F9CH = 32668.

- The following example shows the case where the operation result of the ZONE(P)_U instruction is out of the range from 0 to 65535.

Ex.

When (s2) is 100 and (s3) is 65535, output value is 65535+100 = FFFFH-0064H = 0063H = 99.

- The ZONE(P)_U instruction treats the data in the device specified by (s1) as dummy and does not use it.

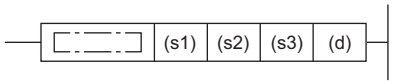
Operation error

There is no operation error.

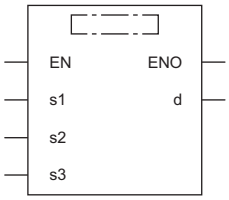
Zone control of 32-bit binary data

DZONE(P)(_U)

These instructions add a bias value to the specified input value (32-bit binary).

Ladder	ST	
	ENO:=DZONE(EN,s1,s2,s3,d); ENO:=DZONEP(EN,s1,s2,s3,d);	ENO:=DZONE_U(EN,s1,s2,s3,d); ENO:=DZONEP_U(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DZONE DZONE_U	
DZONEP DZONEP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DZONE(P)	Device for storing the negative bias value to be added to the input value	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DZONE(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	DZONE(P)	Device for storing the positive bias value to be added to the input value	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DZONE(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s3)	DZONE(P)	Device for storing the Input value used for zone control	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DZONE(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	DZONE(P)	Start device for storing the output value controlled by zone control	—	32-bit signed binary	ANY32_S
	DZONE(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

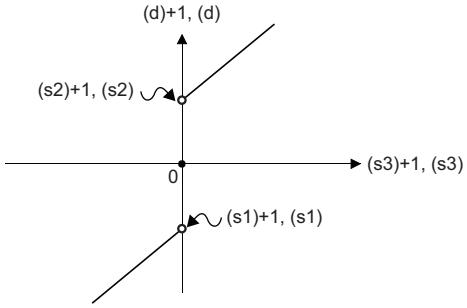
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(s3)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions add the bias value specified by (s1) or (s2) to the input value (32-bit binary) specified by (s3), and store the result in the device number specified by (d). The bias value is controlled as follows.

Condition	Output value
Input value ((s3), (s3)+1) < 0	Input value ((s3), (s3)+1) + negative bias value (s1), (s1)+1
Input value ((s3), (s3)+1) = 0	0
Input value ((s3), (s3)+1) > 0	Input value ((s3), (s3)+1) + positive bias value (s2), (s2)+1



- The following example shows the case where the operation result of the DZONE(P) instruction is out of the range from -2147483648 to 2147483647.

Ex.

When ((s1), (s1)+1) is -1000 and ((s3), (s3)+1) is -2147483648, output value is -2147483648+(-1000) = 80000000H-FFFFFC18H = 7FFFFC18H = 2147482648.

- The following example shows the case where the operation result of the DZONE(P)_U instruction is out of the range from 0 to 4294967295.

Ex.

When ((s2), (s2)+1) is 1000 and ((s3), (s3)+1) is 4294967295, output value is 4294967295+1000 = FFFFFFFFH-00003E8H = 000003E7H = 999.

- The DZONE(P)_U instruction treats the data in the device specified by (s1) and (s1)+1 as dummy and does not use them.

Operation error

There is no operation error.

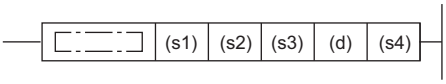
Scaling 16-bit binary data (point coordinates)

SCLN(P)(_U)

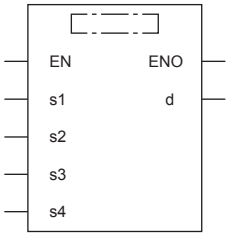
These instructions scale the scaling conversion data (16-bit data) on the basis of the specified input value (point coordinates).

Restriction

When using these instructions, check the versions of the controller and engineering tool used. (MELSEC MX Controller MX-R Model User's Manual)

Ladder	ST	
	ENO:=SCLN(EN,s1,s2,s3,s4,d); ENO:=SCLNP(EN,s1,s2,s3,s4,d);	ENO:=SCLN_U(EN,s1,s2,s3,s4,d); ENO:=SCLNP_U(EN,s1,s2,s3,s4,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SCLN SCLN_U	
SCLNP SCLNP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	SCLN(P)	Search method	0: Sequential search 2: Binary search	16-bit signed binary	ANY16_S
	SCLN(P)_U				
(s2)	SCLN(P)	Input value to be scaled or the device containing the input value	-32768 to 32767	16-bit signed binary	ANY16_S
	SCLN(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s3)	SCLN(P)	Start device containing scaling conversion data	—	16-bit signed binary*1	ANY16_S*2
	SCLN(P)_U			16-bit unsigned binary*1	ANY16_U*2
(d)	SCLN(P)	Device for storing the output value controlled by scaling	—	16-bit signed binary	ANY16_S
	SCLN(P)_U			16-bit unsigned binary	ANY16_U
(s4)	SCLN(P)	Scaling operation mode*3	0: Mode 0 1: Mode 1 2: Mode 2	16-bit signed binary	ANY16_S
	SCLN(P)_U				
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 The number of coordinate points in (s3) is represented in 16-bit unsigned binary.

*2 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

*3 For details on the scaling operation modes, refer to the function items. In SCLN(P)_U, only "1: Mode 1" can be specified.

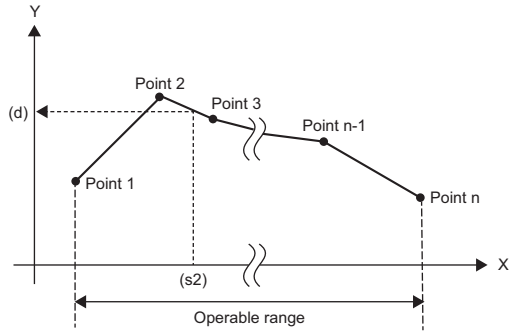
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(s4)	○	○	○	○	○	—	—	○	○	—	—	—

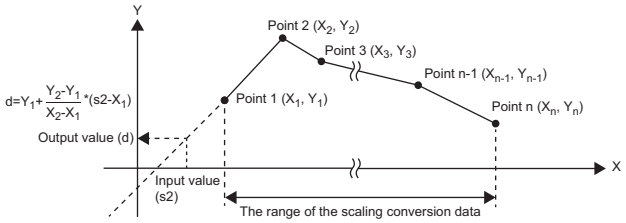
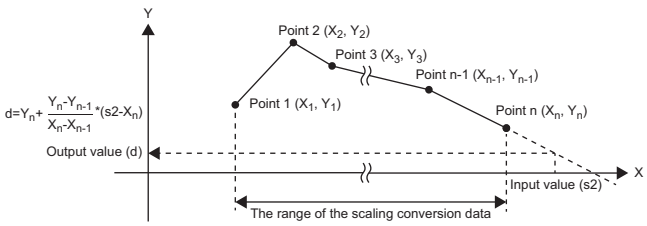
Processing details

- Scales the scaling conversion data (16-bit data) in the device specified by (s3) on the basis of the input value in the device specified by (s2), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s3) and later.

Setting item (n is the number of coordinate points specified by (s3).)		Device assignment
Number of coordinate points		(s3)
Point 1	X coordinate	(s3)+1
	Y coordinate	(s3)+2
Point 2	X coordinate	(s3)+3
	Y coordinate	(s3)+4
⋮		
Point n	X coordinate	(s3)+2n-1
	Y coordinate	(s3)+2n



- Set the X coordinate data of the scaling conversion data in ascending order.
- If two or more points indicate the same X coordinate, the Y coordinate value of the largest point number is output.
- Specify a value from 1 to 65535 for the number of coordinate points of the scaling conversion data specified by (s3).
- The operation varies depending on the scaling operation mode specified by (s4), as shown below.

Item name	(s4): Scaling operation mode		
	0: Mode 0	1: Mode 1	2: Mode 2
Operation result	The value is rounded to the nearest integer.	Same as Mode 0	The value is rounded down to the nearest integer.
Operation when the input value specified by (s2) is out of the range of the scaling conversion data	The output value is calculated using the first or last two points of the scaling conversion data. When the calculated output value (d) is outside the range of -32768 to 32767, a 3283H error occurs. However, if the number of coordinate points is one, the output value cannot be calculated, and a 3285H error occurs. • Input value (s2) < X1  • Input value (s2) > Xn 	3285H error	Same as Mode 0

Precautions

- The search method and the number of searches vary depending on the value of (s1).

(s1)	Search method	Number of searches
0	Sequential search	1≤number of searches≤65535
2	Binary search	1≤number of searches≤16

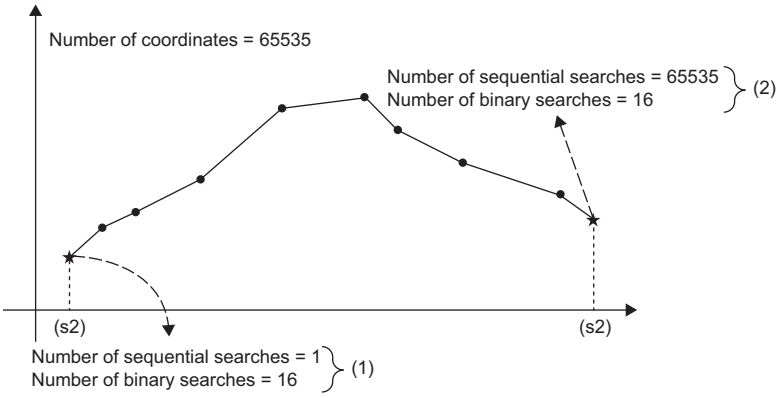
- When the scaling conversion data is sorted in ascending order, the search method varies depending on the value of (s1) and therefore the processing speed also varies. The processing speed depends on the number of searches and is faster as the number of searches is less.

• Case in which the processing speed of sequential search is faster

When the coordinate point specified by (s2) is one from 1 to 15 while the number of coordinate points is the maximum, the number of sequential searches is equal to or less than 15 and therefore the processing speed of the sequential search becomes faster.

• Case in which the processing speed of binary search is faster

The maximum number of searches is 16 and therefore when coordinate point 17 or later is specified by (s2), the number of binary searches is less than the number of sequential searches, and accordingly the processing speed of the binary search becomes faster.



(1) The processing speed of the sequential search is faster because the number of sequential searches is less than the number of binary searches.

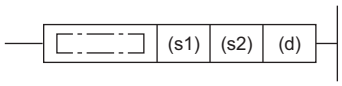
(2) The processing speed of the binary search is faster because the number of binary searches is less than the number of sequential searches.

Operation error

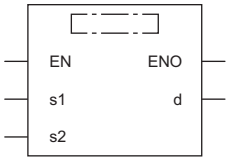
Error code (SD0)	Description
3285H	The X-coordinate data of the scaling conversion data before the point specified by (s2) is not sorted in ascending order. (Note that this error is not detected if the value of (s1) is 2.)
	The value specified by (s4) is 1 and the input value specified by (s2) is out of the range of the specified scaling conversion data.
	The value specified by (s4) is 0 or 2 while the number of coordinate points is 1 and the input value specified by (s2) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s3) is out of the range, 1 to 65535.
	The value specified by (s1) is other than 0 or 2.
	The value specified by (s4) is other than 0 to 2 (for SCLN(P)). Or, the value specified by (s4) is other than 1 (for SCLN(P)_U).
3283H	The value specified by (s4) is 0 or 2 while the input value specified by (s2) is out of the range of the specified scaling conversion data and (d) is out of the range from -32768 to 32767.
2820H	The device specified by (s3) cannot be secured for the size of the scaling conversion data.

SCL(P)(_U)

These instructions scale the scaling conversion data (16-bit data) on the basis of the specified input value (point coordinates).

Ladder	ST	
	ENO:=SCL(EN,s1,s2,d); ENO:=SCLP(EN,s1,s2,d);	ENO:=SCL_U(EN,s1,s2,d); ENO:=SCLP_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SCL SCL_U	
SCLP SCLP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	SCL(P)	Input value to be scaled or the device containing the input value	-32768 to 32767	16-bit signed binary	ANY16_S
	SCL(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	SCL(P)	Start device containing scaling conversion data	—	16-bit signed binary*1	ANY16_S*2
	SCL(P)_U			16-bit unsigned binary*1	ANY16_U*2
(d)	SCL(P)	Device for storing the output value controlled by scaling	—	16-bit signed binary	ANY16_S
	SCL(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 The number of coordinate points in (s2) is represented in 16-bit unsigned binary.
*2 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

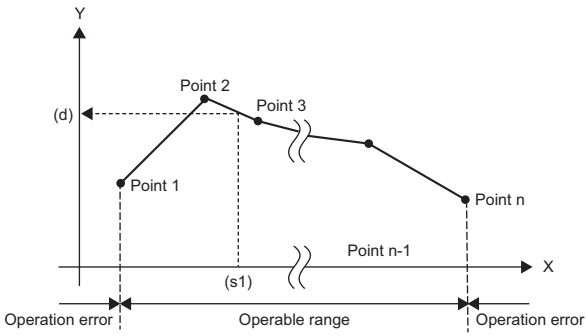
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions scale the scaling conversion data (16-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.

Setting item (n is the number of coordinate points specified by (s2).)		Device assignment
Number of coordinate points		(s2)
Point 1	X coordinate	(s2)+1
	Y coordinate	(s2)+2
Point 2	X coordinate	(s2)+3
	Y coordinate	(s2)+4
⋮		
Point n	X coordinate	(s2)+2n-1
	Y coordinate	(s2)+2n



- If the operation result is not an integer, the first decimal place is rounded off.
- Set the X coordinate data of the scaling conversion data in ascending order.
- Set the value in (s1) within the range of the scaling conversion data (device value in (s2)).
- If two or more points indicate the same X coordinate, the Y coordinate value of the largest point number is output.
- Specify a value from 1 to 65535 for the number of coordinate points of the scaling conversion data specified by (s2).

Precautions

- The search method and the number of searches vary depending on whether SM755 is on or off.

SM755	Search method	Number of searches
Off	Sequential search	1≤number of searches≤65535
On	Binary search	1≤number of searches≤16

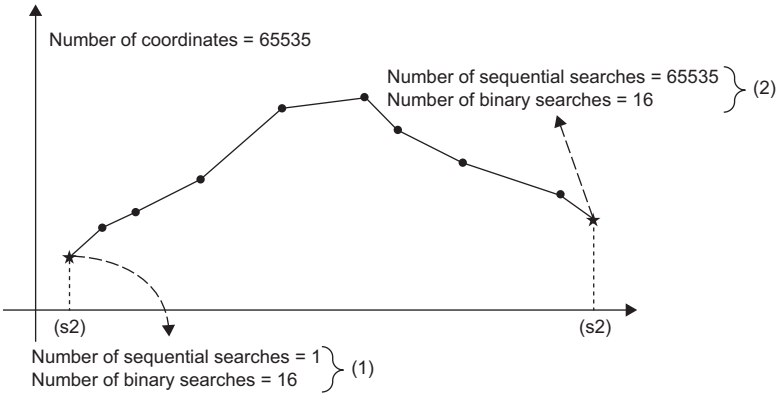
- When the scaling conversion data is sorted in ascending order, the search method varies depending on the status of SM755 and therefore the processing speed also varies. The processing speed depends on the number of searches and is faster as the number of searches is less.

• Case in which the processing speed of sequential search is faster

When the coordinate point specified by (s1) is one from 1 to 15 while the number of coordinate points is the maximum, the number of sequential searches is equal to or less than 15 and therefore the processing speed of the sequential search becomes faster.

• Case in which the processing speed of binary search is faster

The maximum number of searches is 16 and therefore when coordinate point 17 or later is specified by (s1), the number of binary searches is less than the number of sequential searches, and accordingly the processing speed of the binary search becomes faster.



(1) The processing speed of the sequential search is faster because the number of sequential searches is less than the number of binary searches.

(2) The processing speed of the binary search is faster because the number of binary searches is less than the number of sequential searches.

Operation error

Error code (SD0)	Description
3285H	The X-coordinate data of the scaling conversion data before the point specified by (s1) is not sorted in ascending order. (Note that this error is not detected when SM755 is on.)
	The input value specified by (s1) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s2) is out of the range, 1 to 65535.
2820H	The device specified by (s2) cannot be secured for the size of the scaling conversion data.

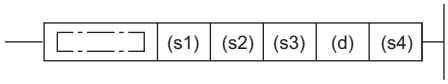
Scaling 32-bit binary data (point coordinates)

DSCLN(P)(_U)

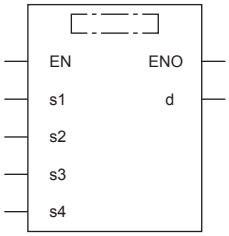
These instructions scale the scaling conversion data (32-bit data) on the basis of the specified input value (point coordinates).

Restriction

When using these instructions, check the versions of the controller and engineering tool used. (MELSEC MX Controller MX-R Model User's Manual)

Ladder	ST	
	ENO:=DSCLN(EN,s1,s2,s3,s4,d); ENO:=DSCLNP(EN,s1,s2,s3,s4,d);	ENO:=DSCLN_U(EN,s1,s2,s3,s4,d); ENO:=DSCLNP_U(EN,s1,s2,s3,s4,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSCLN DSCLN_U	
DSCLNP DSCLNP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DSCLN(P)	Search method	0: Sequential search 2: Binary search	16-bit signed binary	ANY16_S
	DSCLN(P)_U				
(s2)	DSCLN(P)	Input value to be scaled or the device containing the input value	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DSCLN(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s3)	DSCLN(P)	Start device containing scaling conversion data	—	32-bit signed binary*1	ANY32_S*2
	DSCLN(P)_U			32-bit unsigned binary*1	ANY32_U*2
(d)	DSCLN(P)	Device for storing the output value controlled by scaling	—	32-bit signed binary	ANY32_S
	DSCLN(P)_U			32-bit unsigned binary	ANY32_U
(s4)	DSCLN(P)	Scaling operation mode*3	0: Mode 0 1: Mode 1 2: Mode 2	16-bit signed binary	ANY16_S
	DSCLN(P)_U				
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 The number of coordinate points in (s3) is represented in 32-bit unsigned binary.

*2 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

*3 For details on the scaling operation modes, refer to the function items. In DSCLN(P)_U, only "1: Mode 1" can be specified.

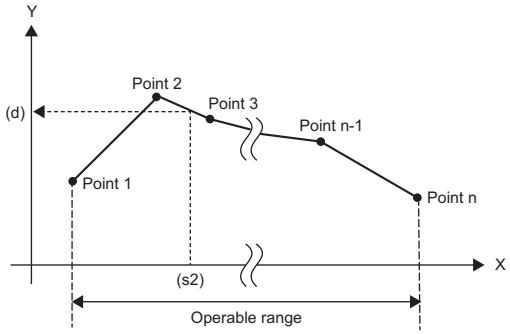
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—
(s4)	○	○	○	○	○	—	—	○	○	—	—	—

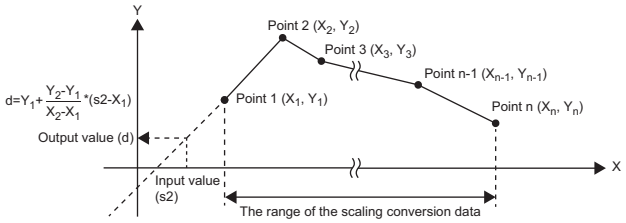
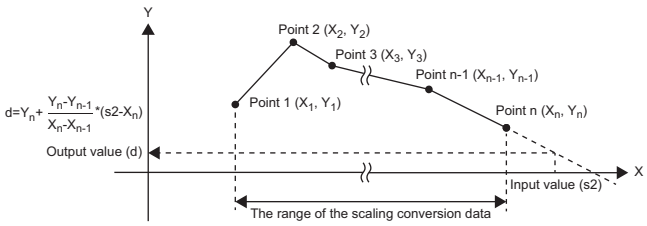
Processing details

- Scales the scaling conversion data (32-bit data) in the device specified by (s3) on the basis of the input value in the device specified by (s2), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s3) and later.

Setting item (n is the number of coordinate points specified by (s3).)		Device assignment
Number of coordinate points		(s3)+1, (s3)
Point 1	X coordinate	(s3)+3, (s3)+2
	Y coordinate	(s3)+5, (s3)+4
Point 2	X coordinate	(s3)+7, (s3)+6
	Y coordinate	(s3)+9, (s3)+8
⋮		
Point n	X coordinate	(s3)+4n-1, (s3)+4n-2
	Y coordinate	(s3)+4n+1, (s3)+4n



- Set the X coordinate data of the scaling conversion data in ascending order.
- If two or more points indicate the same X coordinate, the Y coordinate value of the largest point number is output.
- Specify a value from 1 to 4294967295 for the number of coordinate points of the scaling conversion data specified by (s3).
- The operation varies depending on the scaling operation mode specified by (s4), as shown below.

Item name	(s4): Scaling operation mode		
	0: Mode 0	1: Mode 1	2: Mode 2
Operation result	The value is rounded to the nearest integer.	Same as Mode 0	The value is rounded down to the nearest integer.
Operation when the input value specified by (s2) is out of the range of the scaling conversion data	The output value is calculated using the first or last two points of the scaling conversion data. When the calculated output value (d) is outside the range of -2147483648 to 2147483647, a 3283H error occurs. However, if the number of coordinate points is one, the output value cannot be calculated, and a 3285H error occurs. • Input value (s2) < X1  • Input value (s2) > Xn 	3285H error	Same as Mode 0

Precautions

- The search method and the number of searches vary depending on the value of (s1).

(s1)	Search method	Number of searches
0	Sequential search	1≤number of searches≤4294967295
2	Binary search	1≤number of searches≤32

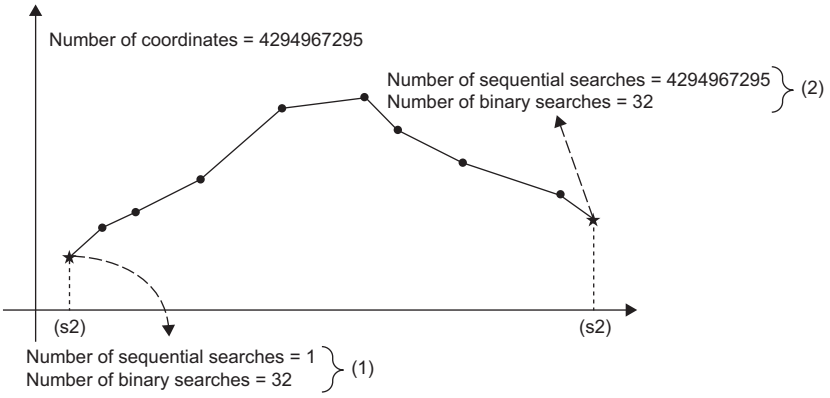
- When the scaling conversion data is sorted in ascending order, the search method varies depending on the value of (s1) and therefore the processing speed also varies. The processing speed depends on the number of searches and is faster as the number of searches is less.

• Case in which the processing speed of sequential search is faster

When the coordinate point specified by (s2) is one from 1 to 31 while the number of coordinate points is the maximum, the number of sequential searches is equal to or less than 31 and therefore the processing speed of the sequential search becomes faster.

• Case in which the processing speed of binary search is faster

The maximum number of searches is 32 and therefore when coordinate point 33 or later is specified by (s2), the number of binary searches is less than the number of sequential searches, and accordingly the processing speed of the binary search becomes faster.



(1) The processing speed of the sequential search is faster because the number of sequential searches is less than the number of binary searches.

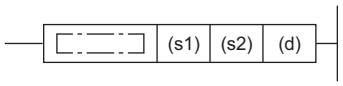
(2) The processing speed of the binary search is faster because the number of binary searches is less than the number of sequential searches.

Operation error

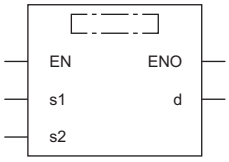
Error code (SD0)	Description
3285H	The X-coordinate data of the scaling conversion data before the point specified by (s2) is not sorted in ascending order. (Note that this error is not detected if the value of (s1) is 2.)
	The value specified by (s4) is 1 and the input value specified by (s2) is out of the range of the specified scaling conversion data.
	The value specified by (s4) is 0 or 2 while the number of coordinate points is 1 and the input value specified by (s2) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s3) is out of the range, 1 to 4294967295.
	The value specified by (s1) is other than 0 or 2.
	The value specified by (s4) is other than 0 to 2 (for DSCLN(P)). Or, the value specified by (s4) is other than 1 (for DSCLN(P)_U).
3283H	The value specified by (s4) is 0 or 2 while the input value specified by (s2) is out of the range of the specified scaling conversion data and (d) is out of the range of -2147483648 to 2147483647.
2820H	The device specified by (s3) cannot be secured for the size of the scaling conversion data.

DSCL(P)(_U)

These instructions scale the scaling conversion data (32-bit data) on the basis of the specified input value (point coordinates).

Ladder	ST	
	ENO:=DSCL(EN,s1,s2,d); ENO:=DSCLP(EN,s1,s2,d);	ENO:=DSCL_U(EN,s1,s2,d); ENO:=DSCLP_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSCL DSCL_U	
DSCLP DSCLP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DSCL(P)	Input value to be scaled or the start device containing the input value	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DSCL(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	DSCL(P)	Start device containing scaling conversion data	—	32-bit signed binary*1	ANY32_S*2
	DSCL(P)_U			32-bit unsigned binary*1	ANY32_U*2
(d)	DSCL(P)	Start device for storing the output value controlled by scaling	—	32-bit signed binary	ANY32_S
	DSCL(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 The number of coordinate points in (s2)+0 and (s2)+1 is represented in 32-bit unsigned binary.

*2 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

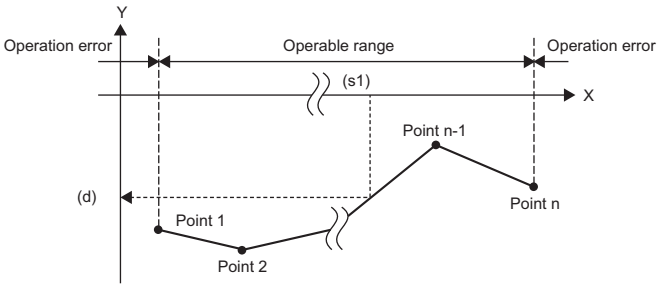
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions scale the scaling conversion data (32-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.

Setting item (n is the number of coordinate points specified by (s2).)		Device assignment
Number of coordinate points		(s2)+1, (s2)
Point 1	X coordinate	(s2)+3, (s2)+2
	Y coordinate	(s2)+5, (s2)+4
Point 2	X coordinate	(s2)+7, (s2)+6
	Y coordinate	(s2)+9, (s2)+8
⋮		
Point n	X coordinate	(s2)+4n-1, (s2)+4n-2
	Y coordinate	(s2)+4n+1, (s2)+4n



- If the operation result is not an integer, the first decimal place is rounded off.
- Set the X coordinate data of the scaling conversion data in ascending order.
- Set the value in (s1) within the range of the scaling conversion data (device value in (s2), (s2)+1).
- If two or more points indicate the same X coordinate, the Y coordinate value of the largest point number is output.
- Specify a value from 1 to 4294967295 for the number of coordinate points of the scaling conversion data specified by (s2).

Precautions

- The search method and the number of searches vary depending on whether SM755 is on or off.

SM755	Search method	Number of searches
Off	Sequential search	1≤number of searches≤4294967295
On	Binary search	1≤number of searches≤32

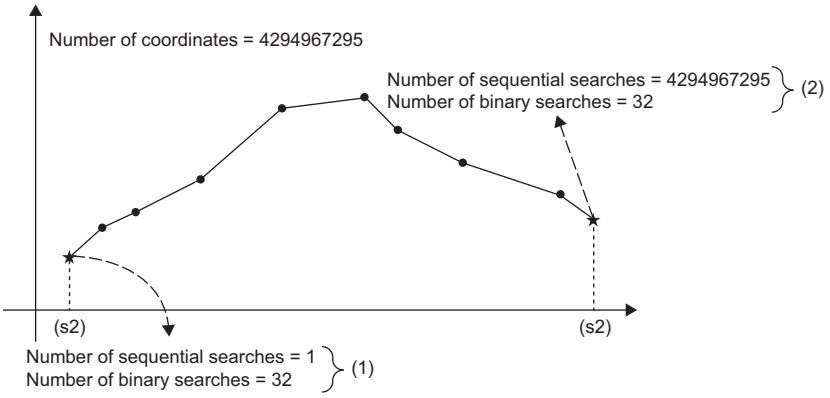
- When the scaling conversion data is sorted in ascending order, the search method varies depending on the status of SM755 and therefore the processing speed also varies. The processing speed depends on the number of searches and is faster as the number of searches is less.

• Case in which the processing speed of sequential search is faster

When the coordinate point specified by (s1) is one from 1 to 15 while the number of coordinate points is the maximum, the number of sequential searches is equal to or less than 15 and therefore the processing speed of the sequential search becomes faster.

• Case in which the processing speed of binary search is faster

The maximum number of searches is 32 and therefore when coordinate point 33 or later is specified by (s1), the number of binary searches is less than the number of sequential searches, and accordingly the processing speed of the binary search becomes faster.



(1) The processing speed of the sequential search is faster because the number of sequential searches is less than the number of binary searches.

(2) The processing speed of the binary search is faster because the number of binary searches is less than the number of sequential searches.

Operation error

Error code (SD0)	Description
3285H	The X-coordinate data of the scaling conversion data before the point specified by (s1) is not sorted in ascending order. (Note that this error is not detected when SM755 is on.)
	The input value specified by (s1) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s2) is out of the range, 1 to 4294967295.
2820H	The device specified by (s2) cannot be secured for the size of the scaling conversion data.

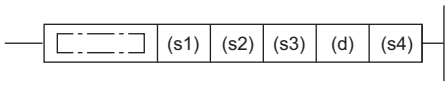
Scaling double-precision real number data (point coordinates)

EDSCLN(P)

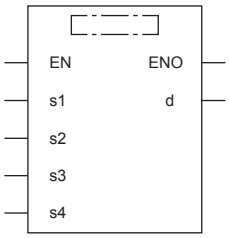
These instructions scale the scaling conversion data (double-precision real number) on the basis of the specified input value (point coordinates).

Restriction

When using these instructions, check the versions of the controller and engineering tool used. (MELSEC MX Controller MX-R Model User's Manual)

Ladder	ST
	ENO:=EDSCLN(EN,s1,s2,s3,s4,d); ENO:=EDSCLNP(EN,s1,s2,s3,s4,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EDSCLN	
EDSCLNP	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	EDSCLN(P)	Search method	0: Sequential search 2: Binary search	16-bit signed binary	ANY16_S
(s2)	EDSCLN(P)	Input value to be scaled or the device containing the input value	0, $2^{-1022} \leq \|(s)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(s3)	EDSCLN(P)	Start device containing scaling conversion data	—	Double-precision real number	ANYREAL_64
(d)	EDSCLN(P)	Device for storing the output value controlled by scaling	—	Double-precision real number	ANYREAL_64
(s4)	EDSCLN(P)	Scaling operation mode*1	0: Mode 0 1: Mode 1	16-bit signed binary	ANY16_S
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 For details on the scaling operation modes, refer to the function items.

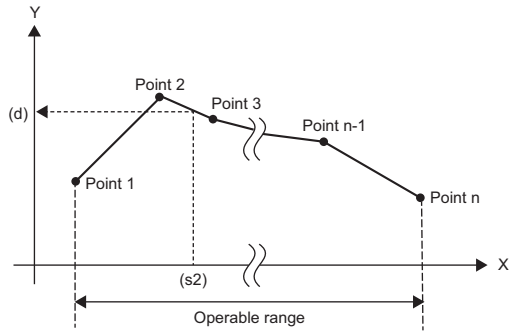
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—
(s4)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

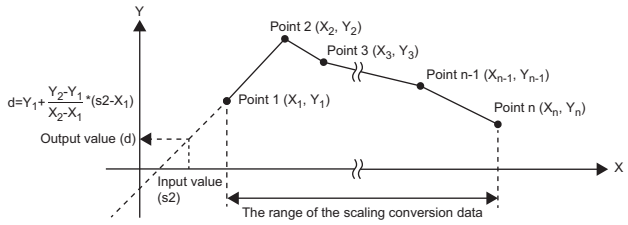
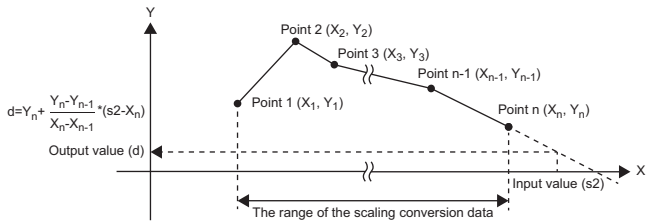
- These instructions scale the scaling conversion data (double-precision real number) in the device specified by (s3) on the basis of the input value in the device specified by (s2), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s3) and later.

Setting item (n is the number of coordinate points specified by (s3).)		Device assignment
Number of coordinate points		(s3)+3, (s3)+2, (s3)+1, (s3)
Point 1	X coordinate	(s3)+7, (s3)+6, (s3)+5, (s3)+4
	Y coordinate	(s3)+11, (s3)+10, (s3)+9, (s3)+8
Point 2	X coordinate	(s3)+15, (s3)+14, (s3)+13, (s3)+12
	Y coordinate	(s3)+19, (s3)+18, (s3)+17, (s3)+16
⋮		
Point n	X coordinate	(s3)+8n-1, (s3)+8n-2, (s3)+8n-3, (s3)+8n-4
	Y coordinate	(s3)+8n+3, (s3)+8n+2, (s3)+8n+1, (s3)+8n



- Set the X coordinate data of the scaling conversion data in ascending order.
- If two or more points indicate the same X coordinate, the Y coordinate value of the largest point number is output.
- For (s3) to (s3)+3, because the value rounded to the nearest integer becomes the number of coordinate points, specify (s3) to (s3)+3 so that the value rounded to the nearest integer is in the range of 1 to 65535.
- For (s2) and (s3), when an input value is set using the engineering tool, a rounding error may occur.

• The operation varies depending on the scaling operation mode specified by (s4), as shown below.

Item name	(s4): Scaling operation mode	
	0: Mode 0	1: Mode 1
Operation when the input value specified by (s2) is out of the range of the scaling conversion data	The output value is calculated using the first or last two points of the scaling conversion data. When the calculated output value (d) is outside the range of $\|(d)\| < 2^{1024}$ (an overflow has occurred), a 3283H error occurs. When -0 or an underflow has occurred, the output value (d) becomes 0. However, if the number of coordinate points is one, the output value cannot be calculated, and a 3285H error occurs. • Input value (s2) < X1  • Input value (s2) > Xn 	3285H error

Precautions

• The search method and the number of searches vary depending on the value of (s1).

(s1)	Search method	Number of searches
0	Sequential search	1≤number of searches≤65535
2	Binary search	1≤number of searches≤16

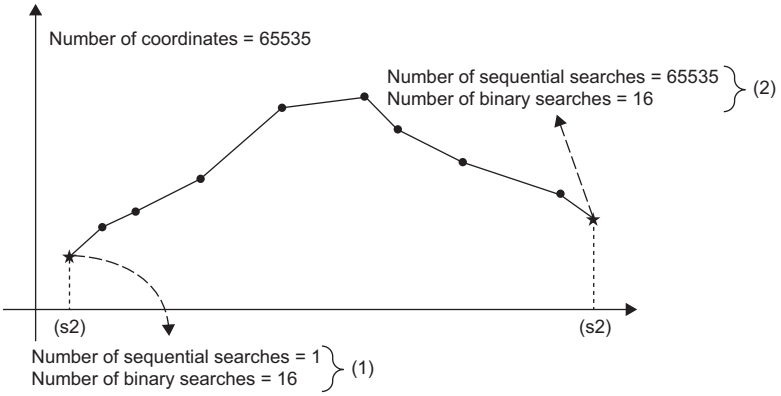
• When the scaling conversion data is sorted in ascending order, the search method varies depending on the value of (s1) and therefore the processing speed also varies. The processing speed depends on the number of searches and is faster as the number of searches is less.

• Case in which the processing speed of sequential search is faster

When the coordinate point specified by (s2) is one from 1 to 15 while the number of coordinate points is the maximum, the number of sequential searches is equal to or less than 15 and therefore the processing speed of the sequential search becomes faster.

• Case in which the processing speed of binary search is faster

The maximum number of searches is 16 and therefore when coordinate point 17 or later is specified by (s2), the number of binary searches is less than the number of sequential searches, and accordingly the processing speed of the binary search becomes faster.



(1) The processing speed of the sequential search is faster because the number of sequential searches is less than the number of binary searches.

(2) The processing speed of the binary search is faster because the number of binary searches is less than the number of sequential searches.

Operation error

Error code (SD0)	Description
3285H	The X-coordinate data of the scaling conversion data before the point specified by (s2) is not sorted in ascending order. (Note that this error is not detected if the value of (s1) is 2.)
	The value specified by (s4) is 1 and the input value specified by (s2) is out of the range of the specified scaling conversion data.
	The value specified by (s4) is 0 while the number of coordinate points is 1 and the input value specified by (s2) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s3) is out of the range, 1 to 65535.
	The value specified by (s1) is other than 0 or 2.
	The value specified by (s4) is other than 0 or 1.
3282H	The input value specified by (s2) is -0, a subnormal number, NaN (not a number), or $\pm\infty$.
	The scaling conversion data specified by (s3) is -0, a subnormal number, NaN (not a number), or $\pm\infty$.
3283H	The value specified by (s4) is 0 while the input value specified by (s2) is out of the range of the specified scaling conversion data and (d) is out of the range of $\|(d)\| < 2^{1024}$ (an overflow has occurred).
2820H	The device specified by (s3) cannot be secured for the size of the scaling conversion data.

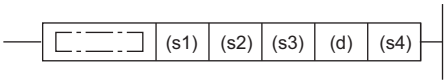
Scaling 16-bit binary data (XY coordinates)

SCL2N(P)(_U)

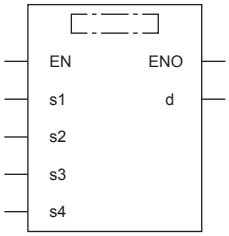
These instructions scale the scaling conversion data (16-bit data) on the basis of the specified input value (XY coordinates).

Restriction

When using these instructions, check the versions of the controller and engineering tool used. (MELSEC MX Controller MX-R Model User's Manual)

Ladder	ST	
	ENO:=SCL2N(EN,s1,s2,s3,s4,d); ENO:=SCL2NP(EN,s1,s2,s3,s4,d);	ENO:=SCL2N_U(EN,s1,s2,s3,s4,d); ENO:=SCL2NP_U(EN,s1,s2,s3,s4,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SCL2N SCL2N_U	
SCL2NP SCL2NP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	SCL2N(P)	Search method	0: Sequential search 2: Binary search	16-bit signed binary	ANY16_S
	SCL2N(P)_U				
(s2)	SCL2N(P)	Input value to be scaled or the device containing the input value	-32768 to 32767	16-bit signed binary	ANY16_S
	SCL2N(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s3)	SCL2N(P)	Start device containing scaling conversion data	—	16-bit signed binary*1	ANY16_S
	SCL2N(P)_U			16-bit unsigned binary*1	ANY16_U
(d)	SCL2N(P)	Device for storing the output value controlled by scaling	—	16-bit signed binary	ANY16_S
	SCL2N(P)_U			16-bit unsigned binary	ANY16_U
(s4)	SCL2N(P)	Scaling operation mode	1: Mode 1 (fixed)	16-bit signed binary	ANY16_S
	SCL2N(P)_U				
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 The number of coordinate points in (s3) is represented in 16-bit unsigned binary.

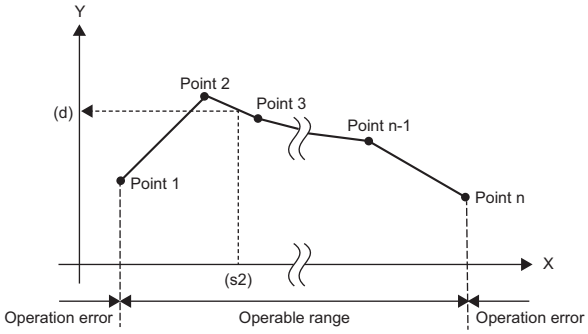
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(s4)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions scale the scaling conversion data (16-bit data) in the device specified by (s3) on the basis of the input value in the device specified by (s2), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s3) and later.

Setting item (n is the number of coordinate points specified by (s3).)		Device assignment
Number of coordinate points		(s3)
X coordinate	Point 1	(s3)+1
	Point 2	(s3)+2
	⋮	⋮
	Point n	(s3)+n
Y coordinate	Point 1	(s3)+n+1
	Point 2	(s3)+n+2
	⋮	⋮
	Point n	(s3)+2n



- If the operation result is not an integer, the first decimal place is rounded off.
- Set the X coordinate data of the scaling conversion data in ascending order.
- Set the value in (s2) within the range of the scaling conversion data (device value in (s3)).
- If two or more points indicate the same X coordinate, the Y coordinate value of the largest point number is output.
- Specify a value from 1 to 65535 for the number of coordinate points of the scaling conversion data.

Precautions

When the scaling conversion data is sorted in ascending order, the search method varies depending on the value of (s1) and therefore the processing speed also varies. For details, refer to the SCLN(P)(_U) instruction.

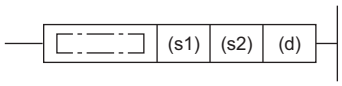
☞ Page 673 SCLN(P)(_U)

Operation error

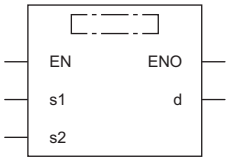
Error code (SD0)	Description
3285H	The X-coordinate data of the scaling conversion data before the point specified by (s2) is not sorted in ascending order. (Note that this error is not detected if the value of (s1) is 2.)
	The input value specified by (s2) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s3) is out of the range, 1 to 65535.
	The value specified by (s1) is other than 0 or 2.
	The value specified by (s4) is other than 1.
2820H	The device specified by (s3) cannot be secured for the size of the scaling conversion data.

SCL2(P)(_U)

These instructions scale the scaling conversion data (16-bit data) on the basis of the specified input value (XY coordinates).

Ladder	ST	
	ENO:=SCL2(EN,s1,s2,d); ENO:=SCL2P(EN,s1,s2,d);	ENO:=SCL2_U(EN,s1,s2,d); ENO:=SCL2P_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SCL2 SCL2_U	
SCL2P SCL2P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	SCL2(P)	Input value to be scaled or the device containing the input value	-32768 to 32767	16-bit signed binary	ANY16_S
	SCL2(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	SCL2(P)	Start device containing scaling conversion data	—	16-bit signed binary*1	ANY16_S*2
	SCL2(P)_U			16-bit unsigned binary*1	ANY16_U*2
(d)	SCL2(P)	Device for storing the output value controlled by scaling	—	16-bit signed binary	ANY16_S
	SCL2(P)_U			16-bit unsigned binary	ANY16_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 The number of coordinate points in (s2) is represented in 16-bit unsigned binary.

*2 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

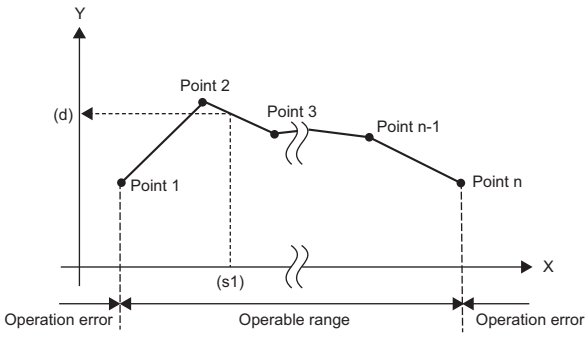
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions scale the scaling conversion data (16-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.

Setting item (n is the number of coordinate points specified by (s2).)		Device assignment
Number of coordinate points		(s2)
X coordinate	Point 1	(s2)+1
	Point 2	(s2)+2
	⋮	⋮
	Point n	(s2)+n
Y coordinate	Point 1	(s2)+n+1
	Point 2	(s2)+n+2
	⋮	⋮
	Point n	(s2)+2n



- If the operation result is not an integer, the first decimal place is rounded off.
- Set the X coordinate data of the scaling conversion data in ascending order.
- Set the value in (s1) within the range of the scaling conversion data (device value in (s2)).
- If two or more points indicate the same X coordinate, the Y coordinate value of the largest point number is output.
- Specify a value from 1 to 65535 for the number of coordinate points of the scaling conversion data.

Precautions

When the scaling conversion data is sorted in ascending order, the search method varies depending on the status of SM755 and therefore the processing speed also varies. For details, refer to the SCL(P)(_U) instruction.

☞ Page 677 SCL(P)(_U)

Operation error

Error code (SD0)	Description
3285H	The X-coordinate data is not sorted in ascending order.
	The input value specified by (s1) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s2) is out of the range, 1 to 65535.
2820H	The device specified by (s2) cannot be secured for the size of the scaling conversion data.

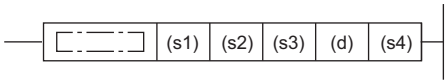
Scaling 32-bit binary data (XY coordinates)

DSCL2N(P)(_U)

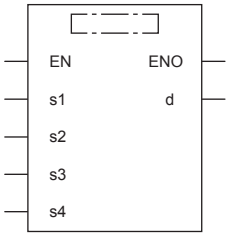
These instructions scale the scaling conversion data (32-bit data) on the basis of the specified input value (XY coordinates).

Restriction

When using these instructions, check the versions of the controller and engineering tool used. (MELSEC MX Controller MX-R Model User's Manual)

Ladder	ST	
	ENO:=DSCL2N(EN,s1,s2,s3,s4,d); ENO:=DSCL2NP(EN,s1,s2,s3,s4,d);	ENO:=DSCL2N_U(EN,s1,s2,s3,s4,d); ENO:=DSCL2NP_U(EN,s1,s2,s3,s4,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSCL2N DSCL2N_U	
DSCL2NP DSCL2NP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DSCL2N(P)	Search method	0: Sequential search 2: Binary search	16-bit signed binary	ANY16_S
	DSCL2N(P)_U				
(s2)	DSCL2N(P)	Input value to be scaled or the device containing the input value	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DSCL2N(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s3)	DSCL2N(P)	Start device containing scaling conversion data	—	32-bit signed binary*1	ANY32_S
	DSCL2N(P)_U			32-bit unsigned binary*1	ANY32_U
(d)	DSCL2N(P)	Device for storing the output value controlled by scaling	—	32-bit signed binary	ANY32_S
	DSCL2N(P)_U			32-bit unsigned binary	ANY32_U
(s4)	DSCL2N(P)	Scaling operation mode	1: Mode 1 (fixed)	16-bit signed binary	ANY16_S
	DSCL2N(P)_U				
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 The number of coordinate points in (s3)+0 and (s3)+1 is represented in 32-bit unsigned binary.

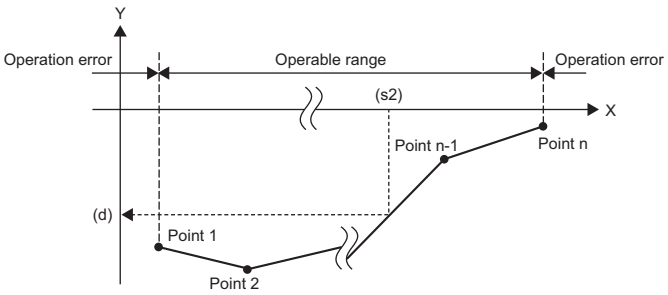
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—
(s4)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions scale the scaling conversion data (32-bit data) in the device specified by (s3) on the basis of the input value in the device specified by (s2), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s3) and later.

Setting item (n is the number of coordinate points specified by (s3).)		Device assignment
Number of coordinate points		(s3)+1, (s3)
X coordinate	Point 1	(s3)+3, (s3)+2
	Point 2	(s3)+5, (s3)+4
	⋮	⋮
	Point n	(s3)+2n+1, (s3)+2n
Y coordinate	Point 1	(s3)+2n+3, (s3)+2n+2
	Point 2	(s3)+2n+5, (s3)+2n+4
	⋮	⋮
	Point n	(s3)+4n+1, (s3)+4n



- If the operation result is not an integer, the first decimal place is rounded off.
- Set the X coordinate data of the scaling conversion data in ascending order.
- Set the value in (s2) within the range of the scaling conversion data (device value in (s3)+0 to (s3)+1).
- If two or more points indicate the same X coordinate, the Y coordinate value of the largest point number is output.
- Specify a value from 1 to 4294967295 for the number of coordinate points of the scaling conversion data.

Precautions

When the scaling conversion data is sorted in ascending order, the search method varies depending on the value of (s1) and therefore the processing speed also varies. For details, refer to the DSCLN(P)(_U) instruction.

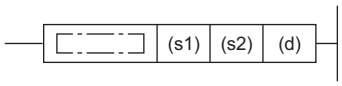
☞ Page 680 DSCLN(P)(_U)

Operation error

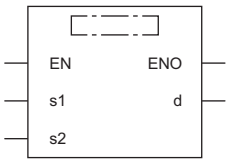
Error code (SD0)	Description
3285H	The X-coordinate data of the scaling conversion data before the point specified by (s2) is not sorted in ascending order. (Note that this error is not detected if the value of (s1) is 2.)
	The input value specified by (s2) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s3) is out of the range, 1 to 4294967295.
	The value specified by (s1) is other than 0 or 2.
	The value specified by (s4) is other than 1.
2820H	The device specified by (s3) cannot be secured for the size of the scaling conversion data.

DSCL2(P)(_U)

These instructions scale the scaling conversion data (32-bit data) on the basis of the specified input value (XY coordinates).

Ladder	ST	
	ENO:=DSCL2(EN,s1,s2,d); ENO:=DSCL2P(EN,s1,s2,d);	ENO:=DSCL2_U(EN,s1,s2,d); ENO:=DSCL2P_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSCL2 DSCL2_U	
DSCL2P DSCL2P_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DSCL2(P)	Input value to be scaled or the start device containing the input value	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DSCL2(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	DSCL2(P)	Start device containing scaling conversion data	—	32-bit signed binary*1	ANY32_S*2
	DSCL2(P)_U			32-bit unsigned binary*1	ANY32_U*2
(d)	DSCL2(P)	Start device for storing the output value controlled by scaling	—	32-bit signed binary	ANY32_S
	DSCL2(P)_U			32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 The number of coordinate points in (s2) to (s2)+1 is represented in 32-bit unsigned binary.

*2 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

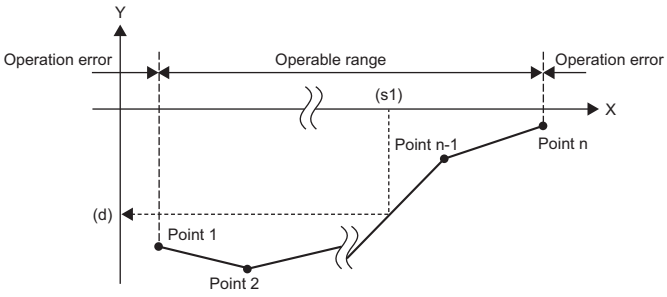
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions scale the scaling conversion data (32-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.

Setting item (n is the number of coordinate points specified by (s2).)		Device assignment
Number of coordinate points		(s2)+1, (s2)
X coordinate	Point 1	(s2)+3, (s2)+2
	Point 2	(s2)+5, (s2)+4
	⋮	⋮
	Point n	(s2)+2n+1, (s2)+2n
Y coordinate	Point 1	(s2)+2n+3, (s2)+2n+2
	Point 2	(s2)+2n+5, (s2)+2n+4
	⋮	⋮
	Point n	(s2)+4n+1, (s2)+4n



- If the operation result is not an integer, the first decimal place is rounded off.
- Set the X coordinate data of the scaling conversion data in ascending order.
- Set the value in (s1) within the range of the scaling conversion data (device value in (s2) to (s2)+1).
- If two or more points indicate the same X coordinate, the Y coordinate value of the largest point number is output.
- Specify a value from 1 to 4294967295 for the number of coordinate points of the scaling conversion data.

Precautions

When the scaling conversion data is sorted in ascending order, the search method varies depending on the status of SM755 and therefore the processing speed also varies. For details, refer to the DSCL(P)(_U) instruction.

☞ Page 684 DSCL(P)(_U)

Operation error

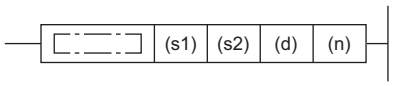
Error code (SD0)	Description
3285H	The X-coordinate data is not sorted in ascending order.
	The input value specified by (s1) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s2) is out of the range, 1 to 4294967295.
2820H	The device specified by (s2) cannot be secured for the size of the scaling conversion data.

14.6 Data Processing Instructions

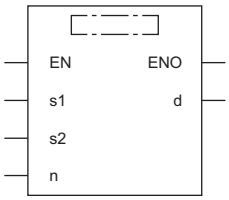
Searching 16-bit binary data

SERDATA(P)

These instructions search the (n) points in the 16-bit binary data specified by (s2) for the 16-bit binary data specified by (s1).

Ladder	ST
	ENO:=SERDATA(EN,s1,s2,n,d); ENO:=SERDATAP(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SERDATA	
SERDATAP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Search data or the device containing the search data	-32768 to 32767	16-bit signed binary	ANY16
(s2)	Start device where the search target data is stored	—	16-bit signed binary	ANY16*1
(d)	Start device for storing the search result	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
(n)	Number of search target data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

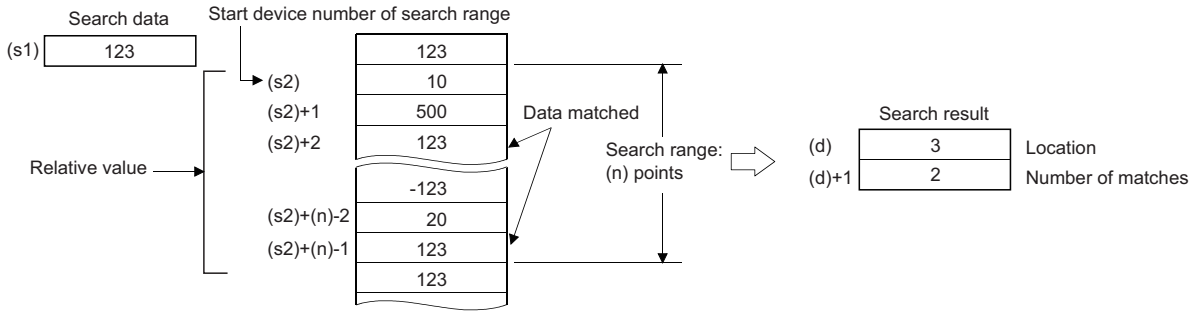
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

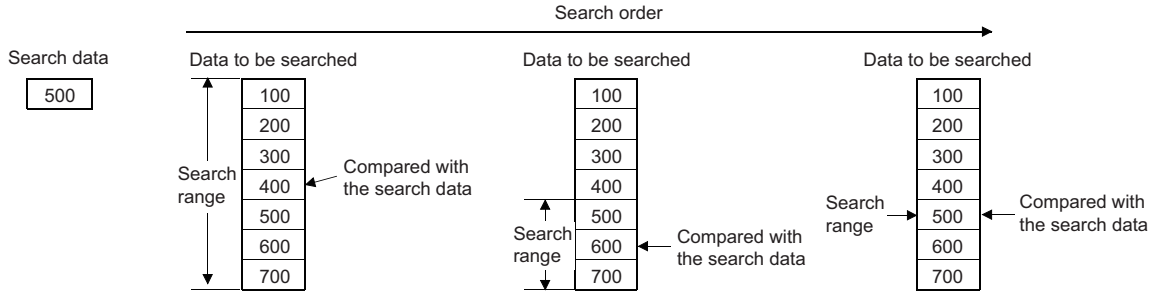
- These instructions search the (n) points in units of 16-bit binary data from the device specified by (s2) using the 16-bit binary data in the device specified by (s1) as a keyword. Each instruction stores the number of data which matches the keyword in the device specified by (d)+1 and also stores the relative value of the first-matched device number from (s2) in the device specified by (d).



- If the value specified by (n) is 0, no processing is performed.
- If no matching data is found as the result of search, 0 is stored in the devices specified by (d) and (d)+1.

Point

- If the data to be searched by the SERDATA(P) instruction has been sorted in ascending order, turning on SM702[*1] enables a binary search which can process the search faster. If SM702 is turned on even though the data to be search has not been sorted in ascending order, normal search results cannot be obtained. The following figure shows an example of binary search.



*1 SM702 is a special relay for setting the search method.
SM702 is off: Sequential search (linear search)
This method compares the search data with the data to be searched for starting from the start of data
SM702 is on: Binary search
For the data that has been sorted in ascending order, this method checks the center value of the search range, determining whether the center value is larger or smaller than the search value, and thereby narrows the search range to either side. Thus, target data is searched for by repeating this processing.

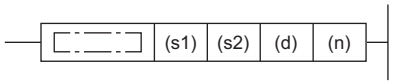
Operation error

There is no operation error.

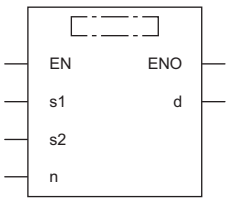
Searching 32-bit binary data

DSERDATA(P)

These instructions search the (n) points in the 32-bit binary data specified by (s2) for the 32-bit binary data specified by (s1).

Ladder	ST
	ENO:=DSERDATA(EN,s1,s2,n,d); ENO:=DSERDATAP(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSERDATA	
DSERDATAP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Search data or the start device containing the search data	-2147483648 to 2147483647	32-bit signed binary	ANY32
(s2)	Start device where the search target data is stored	—	32-bit signed binary	ANY32*1
(d)	Start device for storing the search result	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
(n)	Number of search target data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

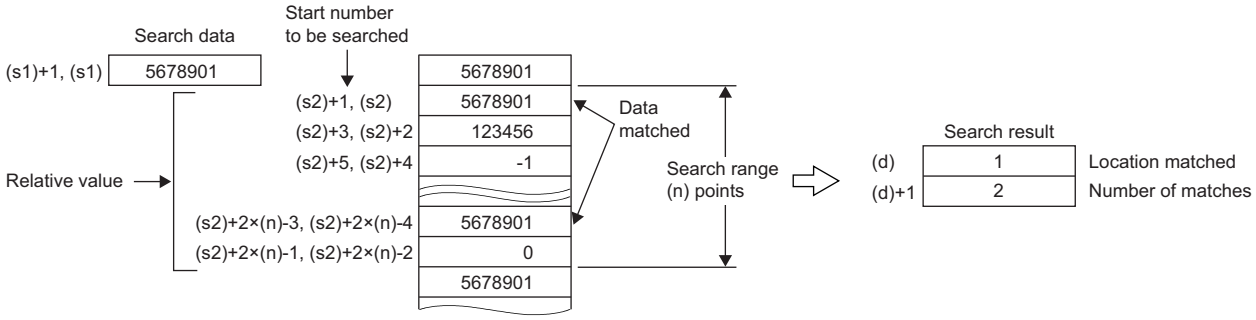
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

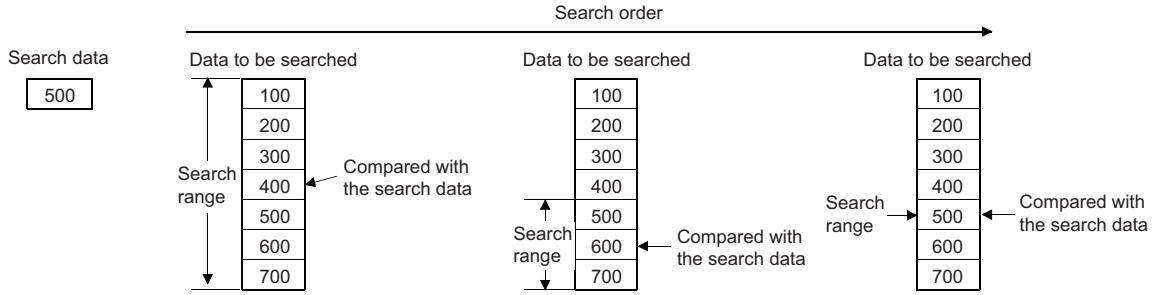
- These instructions search the (n) points of data in units of 32-bit binary data (2 × (n) points of data in units of 16 bits) from the device specified by (s2), using the 32-bit binary data in the device specified by (s1) as a keyword. Each instruction stores the number of data which matches the keyword in the device specified by (d)+1 and also stores the relative value of the first-matched device number from (s2) in the device specified by (d).



- If the value specified by (n) is 0, no processing is performed.
- If no matching data is found as the result of search, 0 is stored in the devices specified by (d) and (d)+1.

Point

- If the data to be searched by the DSERDATA(P) instruction has been sorted in ascending order, turning on SM702[*1] enables a binary search which can process the search faster. If SM702 is turned on even though the data to be search has not been sorted in ascending order, normal search results cannot be obtained. The following figure shows an example of binary search.



*1 SM702 is a special relay for setting the search method.
SM702 is off: Sequential search (linear search)
This method compares the search data with the data to be searched for starting from the start of data
SM702 is on: Binary search
For the data that has been sorted in ascending order, this method checks the center value of the search range, determining whether the center value is larger or smaller than the search value, and thereby narrows the search range to either side. Thus, target data is searched for by repeating this processing.

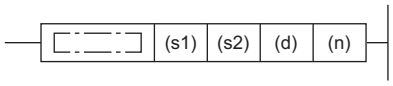
Operation error

There is no operation error.

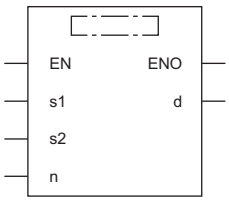
Searching 16-bit binary data (minimum, match, maximum)

SERMM(P)

These instructions search the (n) points in the 16-bit binary data specified by (s1) for the same data as the 16-bit binary data specified by (s2), the minimum value, and the maximum value.

Ladder	ST
	ENO:=SERMM(EN,s1,s2,n,d); ENO:=SERMMP(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SERMM	
SERMMP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device in which the same data, maximum value, and minimum value are searched	—	16-bit signed binary	ANY16
(s2)	Data to be searched for or device storing data	—	16-bit signed binary	ANY16
(d)	Start device storing number of the same data, maximum value, and minimum value detected by search	—	16-bit unsigned binary	ANY16_ARRAY (Number of elements: 5)
(n)	Number of data in which the same data, maximum value, and minimum value are searched	0 to 65535	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○*1	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	○	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

*1 FX and FY cannot be used.

Processing details

- These instructions search the same data as the 16-bit binary data of (s2) in (n) data starting from (s1), and store the search result in (d) to (d)+4.
- When the same data exists, five devices starting from (d) store the number of the same data, first and last positions of the same data, maximum value position, and minimum value position.
- When the same data does not exist, five devices starting from (d) store the number of the same data, first and last positions of the same data, maximum value position, and minimum value position. In this case, however, 0s are stored in three devices starting from (d) (which store the number of the same data, first and last positions of the same data).
- When there are two or more maximum or minimum values in the searched data, the last position of the maximum/minimum values is stored.
- If the value specified by (n) is 0, no processing is performed.
- The following table shows example of configuration of search result table and data. (n=10)

Searched device (s1)	Searched data (s1) value (example)	Comparison data (s2) value (example)	Data position	Search result		
				Maximum value (d)+4	Match (d)	Minimum value (d)+3
(s1)	100	100	0	—	○ (First position)	—
(s1)+1	111		1	—	—	—
(s1)+2	100		2	—	○	—
(s1)+3	98		3	—	—	—
(s1)+4	123		4	—	—	—
(s1)+5	66		5	—	—	○
(s1)+6	100		6	—	○ (Last position)	—
(s1)+7	95		7	—	—	—
(s1)+8	210		8	○	—	—
(s1)+9	88		9	—	—	—

- The following table shows the search result table obtained by the above example.

Device number	Description	Search result item
(d)	3	Number of the same data
(d)+1	0	Position of the same data (first position)
(d)+2	6	Position of the same data (last position)
(d)+3	5	Minimum value position (last position)
(d)+4	8	Maximum value position (last position)

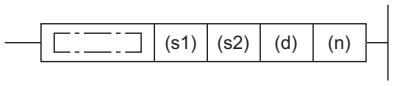
Operation error

There is no operation error.

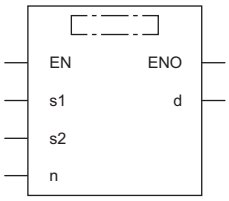
Searching 32-bit binary data (minimum, match, maximum)

DSERMM(P)

These instructions search the (n) points in the 32-bit binary data specified by (s1) for the same data as the 32-bit binary data specified by (s2), the minimum value, and the maximum value.

Ladder	ST
	ENO:=DSERMM(EN,s1,s2,n,d); ENO:=DSERMMP(EN,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSERMM	
DSERMMP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device in which the same data, maximum value, and minimum value are searched	—	32-bit signed binary	ANY32
(s2)	Data to be searched for or device storing data	—	32-bit signed binary	ANY32
(d)	Start device storing number of the same data, maximum value, and minimum value detected by search	—	32-bit unsigned binary	ANY32_ARRAY (Number of elements: 5)
(n)	Number of data in which the same data, maximum value, and minimum value are searched	0 to 65535	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○*1	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	○	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

*1 FX and FY cannot be used.

Processing details

- These instructions search the same data as the 32-bit binary data of (s2)+1 and (s2) in (n) data starting from (s1)+1 and (s1), and store the search result in [(d)+1, (d)] to [(d)+9, (d)+8].
- When the same data exists, five devices starting from (d)+1 and (d) store the number of the same data, first and last positions of the same data, maximum value position, and minimum value position.
- When the same data does not exist, five devices starting from (d)+1 and (d) store the number of the same data, first and last positions of the same data, maximum value position and minimum value position. In this case, however, 0s are stored in three devices starting from (d)+1 and (d) (which store the number of the same data, first and last positions of the same data).
- When there are two or more maximum or minimum values in the searched data, the last position of the maximum/minimum values is stored.
- If the value specified by (n) is 0, no processing is performed.
- The following table shows example of configuration of search result table and data. (n=10)

Searched device (s1)	Searched data (s1) value (example)	Comparison data (s2) value (example)	Data position	Search result		
				Maximum value (d)+9, (d)+8	Match (d)	Minimum value (d)+7, (d)+6
(s1)+1, (s1)	100000	100000	0	—	○ (First position)	—
(s1)+3, (s1)+2	110100		1	—	—	—
(s1)+5, (s1)+4	100000		2	—	○	—
(s1)+7, (s1)+6	98000		3	—	—	—
(s1)+9, (s1)+8	123000		4	—	—	—
(s1)+11, (s1)+10	66000		5	—	—	○
(s1)+13, (s1)+12	100000		6	—	○ (Last position)	—
(s1)+15, (s1)+14	95000		7	—	—	—
(s1)+17, (s1)+16	910000		8	○	—	—
(s1)+19, (s1)+18	910000		9	○	—	—

- The following table shows the search result table obtained by the above example.

Device number	Description	Search result item
(d)+1, (d)	3	Number of the same data
(d)+3, (d)+2	0	Position of the same data (first position)
(d)+5, (d)+4	6	Position of the same data (last position)
(d)+7, (d)+6	5	Minimum value position (last position)
(d)+9, (d)+8	9	Maximum value position (last position)

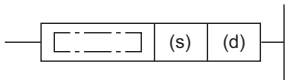
Operation error

There is no operation error.

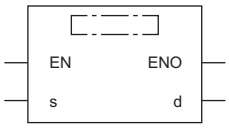
Checking 16-bit binary data

SUM(P)

These instructions store the total number of "1" bits in the 16-bit binary data stored in the specified device.

Ladder	ST
	ENO:=SUM(EN,s,d); ENO:=SUMP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SUM	
SUMP	

Setting data

■Description, range, data type

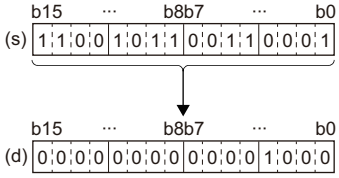
Operand	Description	Range	Data type	Data type (label)
(s)	Device containing data in which the total number of "1" bits is to be counted	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the total number of bits	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions store the total number of "1" bits in the 16-bit binary data, which is stored in the device specified by (s), in the device specified by (d).



The total number of 1s is stored in binary. (Eight 1s in the left example)

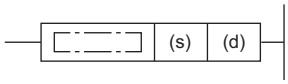
Operation error

There is no operation error.

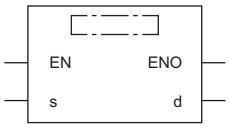
Checking 32-bit binary data

DSUM(P)

These instructions store the total number of "1" bits in the 32-bit binary data stored in the specified device.

Ladder	ST
	ENO:=DSUM(EN,s,d); ENO:=DSUMP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSUM	
DSUMP	

Setting data

■Description, range, data type

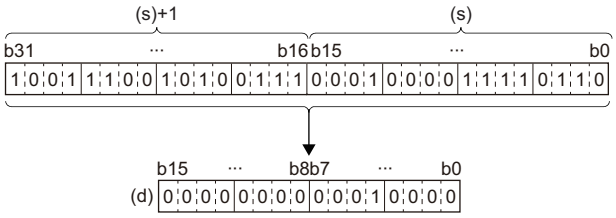
Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing data in which the total number of "1" bits is to be counted	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Device for storing the total number of bits	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions store the total number of "1" bits in the 32-bit binary data, which is stored in the device specified by (s), in the device specified by (d).



The total number of 1s is stored in binary. (Sixteen 1s in the left example)

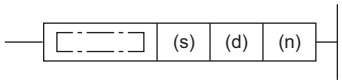
Operation error

There is no operation error.

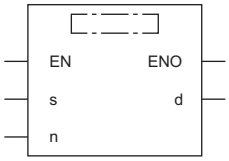
Searching the maximum value of 16-bit binary data

MAX(P)(_U)

These instructions search the (n) points of 16-bit binary data in the specified device for the maximum value.

Ladder	ST*1	
	ENO:=MAXP(EN,s,n,d);	ENO:=MAXP_U(EN,s,n,d);

FBD/LD*1



*1 The MAX and MAX_U instructions do not support the ST and FBD/LD. Use the standard function, MAX.
☞ Page 1316 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
MAX MAX_U	
MAXP MAXP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	MAX(P)	Start device where the search target data is stored	—	16-bit signed binary	ANY16_S*1
	MAX(P)_U			16-bit unsigned binary	ANY16_U*1
(d)	MAX(P)	Start device for storing the search result of the maximum value	—	16-bit signed binary	ANY16_S_ARRAY (Number of elements: 3)
	MAX(P)_U			16-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 3)
(n)		Number of search data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

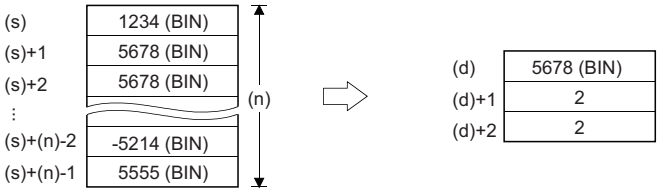
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search the (n) points of 16-bit binary data in the device specified by (s) for the maximum value, and store the maximum value in the device specified by (d). Each instruction searches data starting from the device specified by (s) and detects first the maximum value in the xth point from (s), and stores x in (d)+1 and the number of maximum values in (d)+2.



(d): Maximum value
(d)+1: Location
(d)+2: Number of maximum values

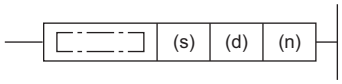
Operation error

There is no operation error.

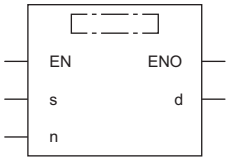
Searching the maximum value of 32-bit binary data

DMAX(P)(_U)

These instructions search the (n) points of 32-bit binary data in the specified device for the maximum value.

Ladder	ST*1	
	ENO:=DMAXP(EN,s,n,d);	ENO:=DMAXP_U(EN,s,n,d);

FBD/LD*1



*1 The DMAX and DMAX_U instructions do not support the ST and FBD/LD. Use the standard function, MAX.
☞ Page 1316 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
DMAX DMAX_U	
DMAXP DMAXP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	DMAX(P)	Start device where the search target data is stored	—	32-bit signed binary	ANY32_S*1
	DMAX(P)_U			32-bit unsigned binary	ANY32_U*1
(d)	DMAX(P)	Start device for storing the search result of the maximum value	—	32-bit signed binary	—*2 (ANY32_S_ARRAY)
	DMAX(P)_U			32-bit unsigned binary	—*2 (ANY32_U_ARRAY)
(n)		Number of search data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

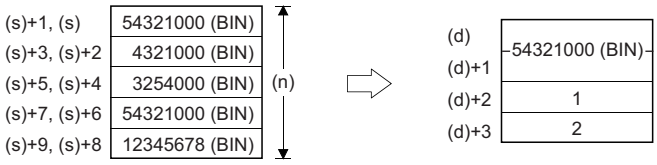
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.
*2 Specify a device regardless of the programming language used. Do not specify labels.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search the (n) points of 32-bit binary data in the device specified by (s) for the maximum value, and store the maximum value in the devices specified by (d) and (d)+1. Each instruction searches data starting from the device specified by (s) and detects first the maximum value in the xth point from (s), and stores x in (d)+2 and the number of maximum values in (d)+3.



(d), (d)+1: Maximum value
(d)+2: Position
(d)+3: Number of maximum values

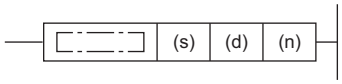
Operation error

There is no operation error.

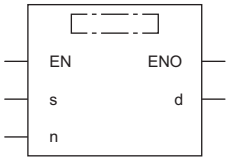
Searching the minimum value of 16-bit binary data

MIN(P)(_U)

These instructions search the (n) points of 16-bit binary data in the specified device for the minimum value.

Ladder	ST*1	
	ENO:=MINP(EN,s,n,d);	ENO:=MINP_U(EN,s,n,d);

FBD/LD*1



*1 The MIN and MIN_U instructions do not support the ST and FBD/LD. Use the standard function, MIN.
☞ Page 1316 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
MIN MIN_U	
MINP MINP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	MIN(P)	Start device where the search target data is stored	—	16-bit signed binary	ANY16_S*1
	MIN(P)_U			16-bit unsigned binary	ANY16_U*1
(d)	MIN(P)	Start device for storing the search result of the minimum value	—	16-bit signed binary	ANY16_S_ARRAY (Number of elements: 3)
	MIN(P)_U			16-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 3)
(n)		Number of search data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

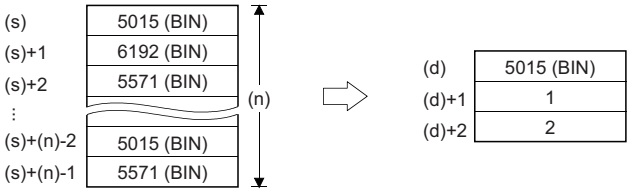
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search the (n) points of 16-bit binary data in the device specified by (s) for the minimum value, and store the minimum value in the device specified by (d). Each instruction searches data starting from the device specified by (s) and detects first the minimum value in the xth point from (s), and stores x in (d)+1 and the number of minimum values in (d)+2.



(d) Minimum value
(d)+1: Location
(d)+2: Number of minimum values

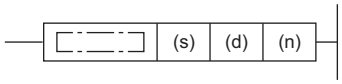
Operation error

There is no operation error.

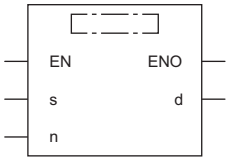
Searching the minimum value of 32-bit binary data

DMIN(P)(_U)

These instructions search the (n) points of 32-bit binary data in the specified device for the minimum value.

Ladder	ST*1	
	ENO:=DMINP(EN,s,n,d);	ENO:=DMINP_U(EN,s,n,d);

FBD/LD*1



*1 The DMIN and DMIN_U instructions do not support the ST and FBD/LD. Use the standard function, MIN.
☞ Page 1316 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
DMIN DMIN_U	
DMINP DMINP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	DMIN(P)	Start device where the search target data is stored	—	32-bit signed binary	ANY32_S*1
	DMIN(P)_U			32-bit unsigned binary	ANY32_U*1
(d)	DMIN(P)	Start device for storing the search result of the minimum value	—	32-bit signed binary	—*2 (ANY32_S_ARRAY)
	DMIN(P)_U			32-bit unsigned binary	—*2 (ANY32_U_ARRAY)
(n)		Number of search data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

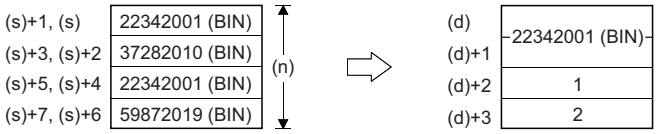
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.
*2 Specify a device regardless of the programming language used. Do not specify labels.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search the (n) points of 32-bit binary data in the device specified by (s) for the minimum value, and store the minimum value in the devices specified by (d) and (d)+1. Each instruction searches data starting from the device specified by (s) and detects first the minimum value in the xth point from (s), and stores x in (d)+2 and the number of minimum values in (d)+3.



(d), (d)+1: Minimum value
(d)+2: Position
(d)+3: Number of maximum values

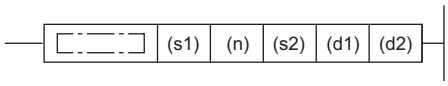
Operation error

There is no operation error.

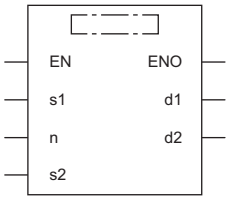
Sorting 16-bit binary data

SORTD(_U)

These instructions sort (n) points of 16-bit binary data in ascending or descending order.

Ladder	ST	
	ENO:=SORTD(EN,s1,n,s2,d1,d2);	ENO:=SORTD_U(EN,s1,n,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
SORTD SORTD_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	SORTD	Start device of the table data to be sorted	—	16-bit signed binary	ANY16_S*1
	SORTD_U			16-bit unsigned binary	ANY16_U*1
(n)		Number of sort data	0 to 65535	16-bit unsigned binary	ANY16
(s2)		Number of data to be compared once	1 to 65535	16-bit unsigned binary	ANY16
(d1)		Number of the bit device to be turned on upon completion of sort	—	Bit	ANY_BOOL
(d2)		Device used by the system	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

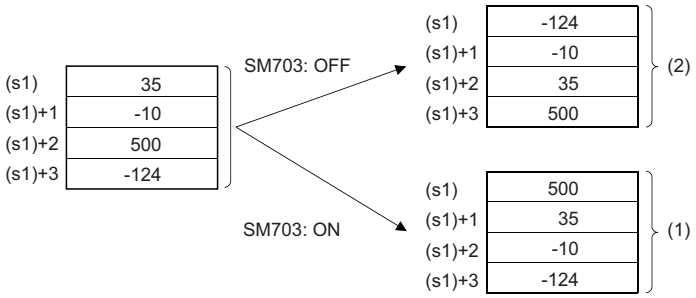
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d1)	○	—	○*1	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

*1 T, C, and ST cannot be used.

Processing details

- These instructions sort (n) points of 16-bit binary data from (s1) in ascending or descending order. Data is sorted in ascending order when SM703 is off and in descending order when SM703 is on.



(1) Data are sorted in descending order.
(2) Data are sorted in ascending order.

- Sorting by the SORTD(_U) instruction requires several scans. The number of scans required till completion of sorting is determined by dividing the maximum number of executions performed before completion of sorting by the number of data compared once specified by (s2). (The decimal fractions are rounded up.) When the value in (s2) is increased, the number of scans before completion of sorting is decreased but the scan time is increased.
- The maximum number of executions before completion of sorting is calculated by (n)×(n-1)÷2 (times). When (n)=10, for example, 10×(10-1)÷2 = 45 times. At this time, setting (s2)=2, for example, makes 45÷2 = 22.5 meaning that 23 scans are required before completion of sorting.
- The completion device specified by (d1) turns off at start of execution of the SORTD(_U) instruction and turns on upon completion of sorting. After completion of sorting, the device specified by (d1) is kept on. Turn it off as needed.
- The two points from the device specified by (d2) are used by the system at execution of the SORTD(_U) instruction. Do not change the two points from the device specified by (d2). If they are changed, an error may occur. (Error code: 3285H)
- If the value in (n) is changed during sorting, the new number of sort data is used for sorting.
- If the execution command is turned off during sorting, sorting is interrupted. If the execution command is turned on again, sorting is performed from the beginning.
- If the next sorting is performed continuously after completion of the previous sorting, the execution command needs to be turned off and turned on again.

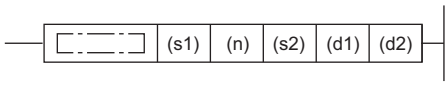
Operation error

Error code (SD0)	Description
2821H	The device range of (n) points from the device specified by (s1) and the device range of two points from the device specified by (d2) are overlapping.
3285H	The value in (s2) is 0.
	In the second scan or after, the value in (d2) used by the system is equal to or greater than the value in (n).
	In the second scan or after, the value in (d2) used by the system is (d2)<(d2)+1.

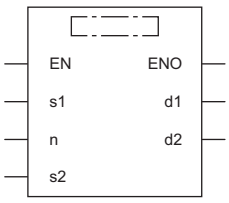
Sorting 32-bit binary data

DSORTD(_U)

These instructions sort (n) points of 32-bit binary data in ascending or descending order.

Ladder	ST	
	ENO:=DSORTD(EN,s1,n,s2,d1,d2);	ENO:=DSORTD_U(EN,s1,n,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DSORTD DSORTD_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DSORTD	Start device of the table data to be sorted	—	32-bit signed binary	ANY32_S*1
	DSORTD_U			32-bit unsigned binary	ANY32_U*1
(n)		Number of sort data	0 to 65535	16-bit unsigned binary	ANY16
(s2)		Number of data to be compared once	1 to 65535	16-bit unsigned binary	ANY16
(d1)		Number of the bit device to be turned on upon completion of sort	—	Bit	ANY_BOOL
(d2)		Device used by the system	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

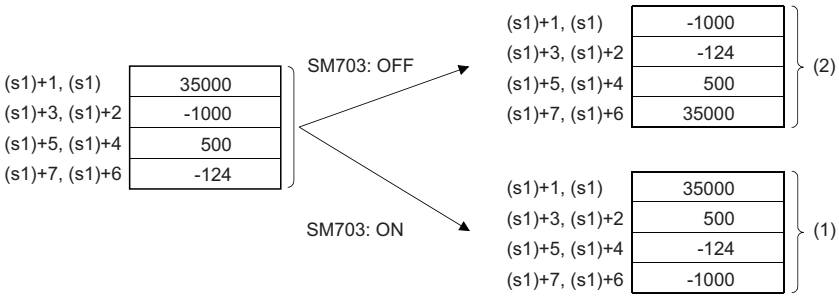
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d1)	○	—	○*1	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

*1 T, C, and ST cannot be used.

Processing details

- These instructions sort (n) points of 32-bit binary data from (s1) in ascending or descending order. Data is sorted in ascending order when SM703 is off and in descending order when SM703 is on.



(1) Data are sorted in descending order.
(2) Data are sorted in ascending order.

- Sorting by the DSORTD(_U) instruction requires several scans. The number of scans required till completion of sorting is determined by dividing the maximum number of executions performed before completion of sorting by the number of data compared once specified by (s2). (The decimal fractions are rounded up.) When the value in (s2) is increased, the number of scans before completion of sorting is decreased but the scan time is increased.
- The maximum number of executions before completion of sorting is calculated by (n)×(n-1)÷2 (times). When (n)=10, for example, 10×(10-1)÷2 = 45 times. At this time, setting (s2)=2, for example, makes 45÷2 = 22.5 meaning that 23 scans are required before completion of sorting.
- The completion device specified by (d1) turns off at start of execution of the DSORTD(_U) instruction and turns on upon completion of sorting. After completion of sorting, the device specified by (d1) is kept on. Turn it off as needed.
- The two points from the device specified by (d2) are used by the system at execution of the DSORTD(_U) instruction. Do not change the two points from the device specified by (d2). If they are changed, an error may occur. (Error code: 3285H)
- If the value in (n) is changed during sorting, the new number of sort data is used for sorting.
- If the execution command is turned off during sorting, sorting is interrupted. If the execution command is turned on again, sorting is performed from the beginning.
- If the next sorting is performed continuously after completion of the previous sorting, the execution command needs to be turned off and turned on again.

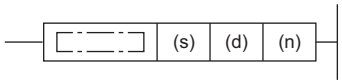
Operation error

Error code (SD0)	Description
2821H	The device range of 2×(n) points from the device specified by (s1) and the device range of two points from the device specified by (d2) are overlapping.
3285H	The value in (s2) is 0.
	In the second scan or after, the value in (d2) used by the system is equal to or greater than the value in (n).
	In the second scan or after, the value in (d2) used by the system is (d2)<(d2)+1.

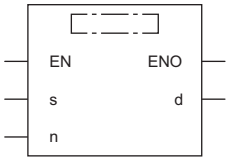
Adding 16-bit binary data

WSUM(P)(_U)

These instructions add the (n) points of 16-bit binary data from the specified device.

Ladder	ST	
	ENO:=WSUM(EN,s,n,d); ENO:=WSUMP(EN,s,n,d);	ENO:=WSUM_U(EN,s,n,d); ENO:=WSUMP_U(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
WSUM WSUM_U	
WSUMP WSUMP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	WSUM(P)	Start device where the data for calculating the total value are stored	—	16-bit signed binary	ANY16_S*1
	WSUM(P)_U			16-bit unsigned binary	ANY16_U*1
(d)	WSUM(P)	Start device for storing the total value	—	32-bit signed binary	ANY32_S
	WSUM(P)_U			32-bit unsigned binary	ANY32_U
(n)		Number of data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

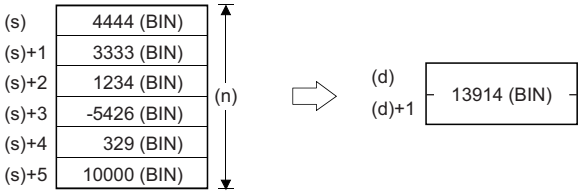
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions add the (n) points of 16-bit binary data in the device starting from the one specified by (s), and store the result in the device specified by (d).



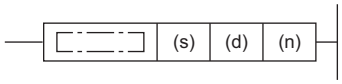
Operation error

There is no operation error.

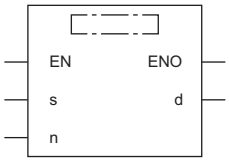
Adding 32-bit binary data

DWSUM(P)(_U)

These instructions add the (n) points of 32-bit binary data in the devices starting from the specified one.

Ladder	ST	
	ENO:=DWSUM(EN,s,n,d); ENO:=DWSUMP(EN,s,n,d);	ENO:=DWSUM_U(EN,s,n,d); ENO:=DWSUMP_U(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DWSUM DWSUM_U	
DWSUMP DWSUMP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	DWSUM(P)	Start device where the data for calculating the total value are stored	—	32-bit signed binary	ANY32_S*1
	DWSUM(P)_U			32-bit unsigned binary	ANY32_U*1
(d)	DWSUM(P)	Start device for storing the total value	—	64-bit signed binary	ANY32_ARRAY (Number of elements: 2)
	DWSUM(P)_U			64-bit unsigned binary	
(n)		Number of data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

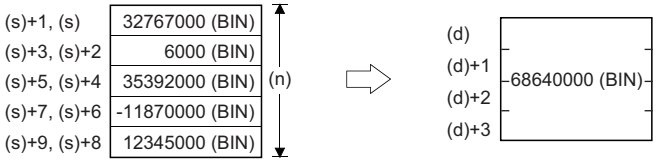
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions add the (n) points of 32-bit binary data in the device starting from the one specified by (s), and store the result in the device specified by (d).



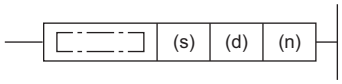
Operation error

There is no operation error.

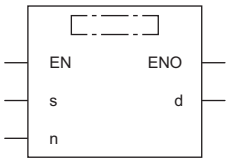
Calculating the mean value of 16-bit binary data

MEAN(P)(_U)

These instructions calculate the average value of the (n) points of 16-bit data in the devices starting from the specified one.

Ladder	ST	
	ENO:=MEAN(EN,s,n,d); ENO:=MEANP(EN,s,n,d);	ENO:=MEAN_U(EN,s,n,d); ENO:=MEANP_U(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
MEAN MEAN_U	
MEANP MEANP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	MEAN(P)	Start device where the data for calculating the average value are stored	—	16-bit signed binary	ANY16_S*1
	MEAN(P)_U			16-bit unsigned binary	ANY16_U*1
(d)	MEAN(P)	Device for storing the mean value	—	16-bit signed binary	ANY16_S
	MEAN(P)_U			16-bit unsigned binary	ANY16_U
(n)		Number of data, or the device number where the number of data is stored	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

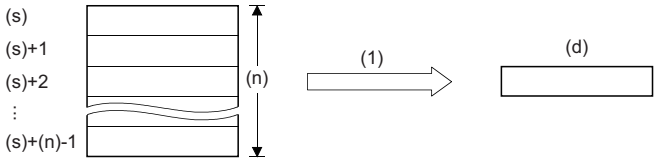
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions calculate the average value of the (n) points of 16-bit binary data in the devices starting from the one specified by (s), and stores the average value in the device specified by (d).



(1) Mean value

- If the calculation result is not an integer, the first decimal place is rounded down.
- When (n) is 0, the processing is not performed.

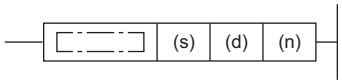
Operation error

There is no operation error.

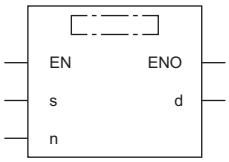
Calculating the mean value of 32-bit binary data

DMEAN(P)(_U)

These instructions calculate the average value of the (n) points of 32-bit data in the devices starting from the specified one.

Ladder	ST	
	ENO:=DMEAN(EN,s,n,d); ENO:=DMEANP(EN,s,n,d);	ENO:=DMEAN_U(EN,s,n,d); ENO:=DMEANP_U(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DMEAN DMEAN_U	
DMEANP DMEANP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	DMEAN(P)	Start device where the data for calculating the average value are stored	—	32-bit signed binary	ANY32_S*1
	DMEAN(P)_U			32-bit unsigned binary	ANY32_U*1
(d)	DMEAN(P)	Start device for storing the average value	—	32-bit signed binary	ANY32_S
	DMEAN(P)_U			32-bit unsigned binary	ANY32_U
(n)		Number of data, or the device number where the number of data is stored	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

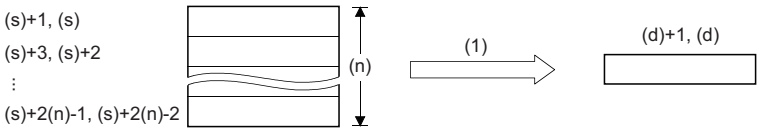
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	—	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions calculate the average value of the (n) points of 32-bit binary data in the devices starting from the one specified by (s), and stores the average value in the device specified by (d).



(1) Mean value

- If the calculation result is not an integer, the first decimal place is rounded down.
- When (n) is 0, the processing is not performed.

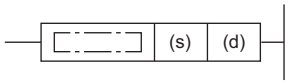
Operation error

There is no operation error.

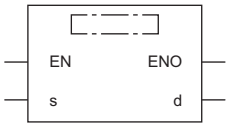
Calculating the square root of 16-bit binary data

SQRT(P)

These instructions perform a square root operation of the specified 16-bit binary data.

Ladder	ST*1
	ENO:=SQRTP(EN,s,d);

FBD/LD*1



*1 The SQRT instruction is not supported by the structured text language and the FBD/LD language. Use the standard function, SQRT. (☞ Page 1276 SQRT(_E))

■Execution condition

Instruction	Execution condition
SQRT	
SQRTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device where the data whose square root is to be calculated is stored	0 to 65535	16-bit unsigned binary	ANY16
(d)	Device where the obtained square root is stored	—	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	○	○	○	○	—	—	○	○	—	—	—
(d)	○*1	○	○	○	○	—	—	○	—	—	—	—

*1 FX and FY cannot be used.

Processing details

- These instructions perform a square root operation of the 16-bit binary data specified by (s). and stores the result in (d). The obtained square root is an integer because the decimal places are rounded down.

$\sqrt{(s)} \rightarrow (d)$

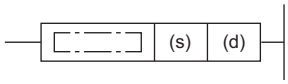
Operation error

There is no operation error.

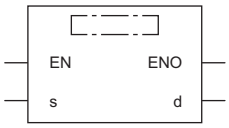
Calculating the square root of 32-bit binary data

DSQRT(P)

These instructions perform a square root operation of the specified 32-bit binary data.

Ladder	ST
	ENO:=DSQRT(EN,s,d); ENO:=DSQRTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSQRT	
DSQRTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device where the data whose square root is to be calculated is stored	0 to 4294967295	32-bit unsigned binary	ANY32
(d)	Device where the obtained square root is stored	—	32-bit unsigned binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	○	○	○	○	○	○	○	○	—	—	—
(d)	○*1	○	○	○	○	○	○	○	—	—	—	—

*1 FX and FY cannot be used.

Processing details

- These instructions perform a square root operation of the 32-bit binary data specified by (s). and stores the result in (d).
  The obtained square root is an integer because the decimal places are rounded down.

$\sqrt{(s)+1, (s)} \rightarrow (d)$

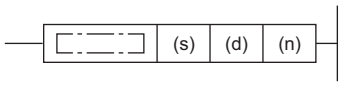
Operation error

There is no operation error.

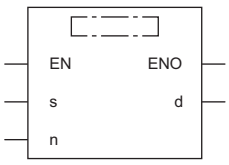
CRC operation

CRC(P)

These instructions generate the CRC value for (n) 8-bit data (unit: byte) starting from the device specified by (s), and store the CRC value to the device specified by (d).

Ladder	ST
	ENO:=CRC(EN,s,n,d); ENO:=CRCP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
CRC	
CRCP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the target data of CRC value generation is stored	—	16-bit signed binary	ANY16
(d)	Device where the generated CRC value is stored	—	16-bit signed binary	ANY16
(n)	Number of 8-bit data (unit: byte) for which the CRC value is generated or the device storing the number of 8-bit data (unit: byte)	0 to 65535	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	—	○	—	—	—	—	○	—	—	—	—
(d)	○*1	—	○	—	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

*1 FX and FY cannot be used.

Processing details

- These instructions generate the CRC value for (n) 8-bit data (unit: byte) starting from the device specified by (s), and store the CRC value to the device specified by (d). "$X^{16} + X^{15} + X^2 + 1$" is used in the generator polynomial of the CRC value (CRC-16). The 16-bit conversion mode and 8-bit conversion mode are available for these instructions. The conversion mode can be selected by turning on or off SM772.
- If the value specified by (n) is 0, no processing is performed.
- The operation in each conversion mode is described below.

■16-bit conversion mode (while SM772 is off)

In this mode, the CRC operation is executed for upper 8 bits (in units of byte) and lower 8 bits (in units of byte) of the device specified by (s). The operation result is stored to 16 bits in one device specified by (d).

Ex.

When (n)=6

In 16-bit conversion mode, the six bytes in the following shaded portion become an operation target. The CRC value is determined as "A57BH", and therefore "A57BH" is stored in the device specified by (d).

	Decimal	Hexadecimal	
		Upper	Lower
D0	24932	61H	64H
D1	4219	10H	7BH
D2	-1333	FAH	CBH
D3	-1	FFH	FFH
D4	32761	7FH	F9H
D5	10000	27H	10H

■8-bit conversion mode (while SM772 is on)

CRC operation is executed only for lower 8 bits (lower byte) of the device specified by (s). With regard to the operation result, lower 8 bits (in units of byte) are stored to the device specified by (d), and upper 8 bits (in units of byte) are stored to a device specified by (d)+1.

Ex.

When (n)=6

In 8-bit conversion mode, the six bytes in the following shaded portion become an operation target. The CRC value is "BDA1H" and therefore "A1H" is stored in the device specified by (d) and "BDH" is stored in the device specified by (d)+1.

	Decimal	Hexadecimal	
		Upper	Lower
D0	24932	61H	64H
D1	4219	10H	7BH
D2	-1333	FAH	CBH
D3	-1	FFH	FFH
D4	32761	7FH	F9H
D5	10000	27H	10H

Operation error

Error code (SD0)	Description
3285H	A digit other than 4 is specified in the digit-specified bit device in (s) and (d).

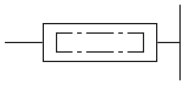
15 DEBUGGING AND FAILURE DIAGNOSTIC

15.1 Debugging and Failure Diagnostic Instructions

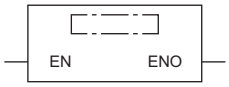
Resetting the error display and the annunciator display

LEDR

This instruction resets the self-diagnostic error (continuation error) display and the annunciator display of the controller.

Ladder	ST
	ENO:=LEDR(EN);

FBD/LD



■Execution condition

Instruction	Execution condition
LEDR	

Processing details

- This instruction resets the self-diagnostic error (continuation error) display and the annunciator display of the controller.
  Executing the instruction once resets both the error display and the annunciator display.
- When a self-diagnostic error has occurred, the controller operates as follows:
  - When a self-diagnostic error (continuation error) has occurred

The ERROR LED on the front of the controller remains off.
The values in SM0, SM1, and SD0 are not reset automatically at this time. Reset the values by the program.

  - When a battery error has occurred

When the LEDR instruction is executed after battery replacement, the BATTERY LED on the front of the controller turns off. At this time, SM51 also turns off.

- When the annunciator (F) is on, the controller operates as follows:
  - The USER LED turns off.
  - The values in SD62, SD63, and SD64 to SD79 are all cleared to 0.



(1) All data are cleared.

Operation error

There is no operation error.

MEMO

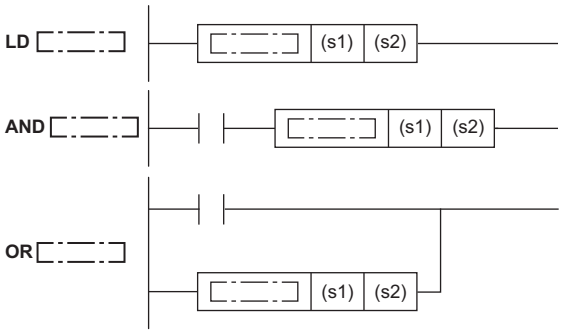
16 STRING PROCESSING

16.1 String Processing Instructions

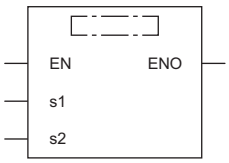
Comparing string data

LD$☐, AND$☐, OR$☐

These instructions compare string data as normally open contacts.

Ladder	ST
 (☐ is replaced by $=, $<>, $>, $<=, $<, or $>=.)	ENO:=LDSTRING_☐(EN,s1,s2); ENO:=ANDSTRING_☐(EN,s1,s2); ENO:=ORSTRING_☐(EN,s1,s2); (☐ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)*1

FBD/LD



(☐ is replaced by a combination of LDSTRING_, ANDSTRING_, or ORSTRING_ and EQ, NE, GT, LE, LT, or GE.)*1

*1 EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
LD$☐, AND$☐, OR$☐	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Comparison data or the start device where the comparison data is stored	—	String	ANYSTRING_SINGLE
(s2)	Comparison data or the start device where the comparison data is stored	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

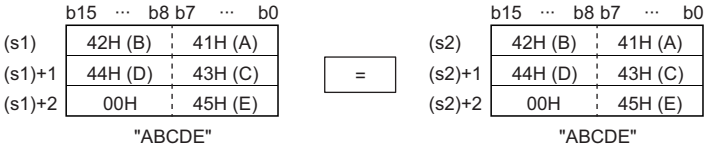
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—

Processing details

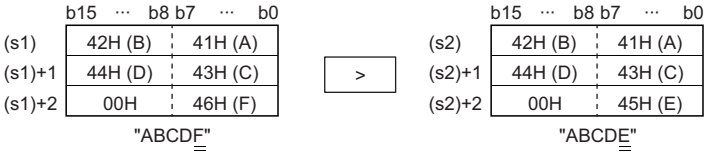
- These instructions perform a comparison operation between the string data in the device specified by (s1) and the string data in the device specified by (s2). (Devices are used as normally open contacts).
- In comparison operation, the ASCII code of string is compared character by character from the beginning of the string.
- The strings in the devices specified by (s1) and (s2) mean those in the device numbers from the specified one to the one containing 00H.

• The comparison result turns out matching if all strings match.



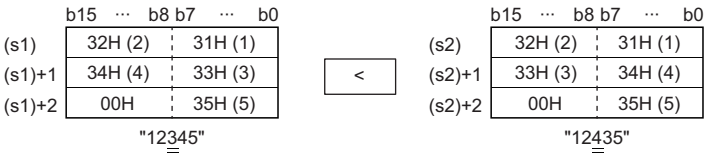
☐ Instruction symbol (ladder, FBD/LD)	Result
$=, EQ	Continuity state (ENO is on.)
$<>, NE	Non-continuity state (ENO is off.)
$>, GT	Non-continuity state (ENO is off.)
$<=, LE	Continuity state (ENO is on.)
$<, LT	Non-continuity state (ENO is off.)
$>=, GE	Continuity state (ENO is on.)

• When different strings are compared, the string with a larger character code is greater.



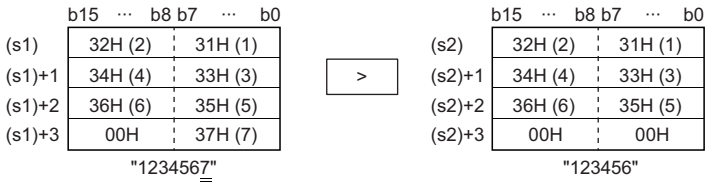
☐ Instruction symbol (ladder, FBD/LD)	Result
$=, EQ	Non-continuity state (ENO is off.)
$<>, NE	Continuity state (ENO is on.)
$>, GT	Continuity state (ENO is on.)
$<=, LE	Non-continuity state (ENO is off.)
$<, LT	Non-continuity state (ENO is off.)
$>=, GE	Continuity state (ENO is on.)

• When different strings are compared, the relative size of a string is determined by the relative size of the first different character codes.



☐ Instruction symbol (ladder, FBD/LD)	Result
$=, EQ	Non-continuity state (ENO is off.)
$<>, NE	Continuity state (ENO is on.)
$>, GT	Non-continuity state (ENO is off.)
$<=, LE	Continuity state (ENO is on.)
$<, LT	Continuity state (ENO is on.)
$>=, GE	Non-continuity state (ENO is off.)

• When the lengths of the string data in the devices specified by (s1) and (s2) are different, the longer string data is greater.



☐ Instruction symbol (ladder, FBD/LD)	Result
$=, EQ	Non-continuity state (ENO is off.)
$<>, NE	Continuity state (ENO is on.)
$>, GT	Continuity state (ENO is on.)
$<=, LE	Non-continuity state (ENO is off.)
$<, LT	Non-continuity state (ENO is off.)
$>=, GE	Continuity state (ENO is on.)

- If the string in the device specified by (s1) or (s2) exceeds 16383 characters, the operation result will be non-continuity (ENO is off).
- If the LDSTRING_☐ instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.
- If the ORSTRING☐ instruction is used in the program written in FBD/LD and EN is set to TRUE, ENO turns on. EN will not be an execution condition.

Operation error

There is no operation error.

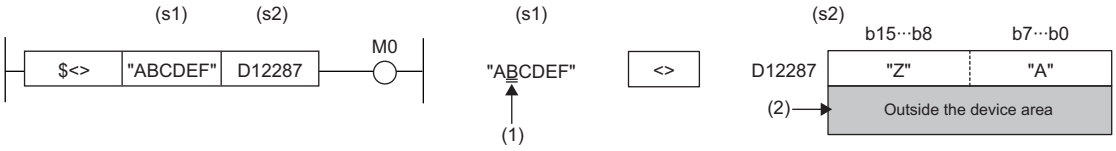
Point

The string comparison instructions perform the following checks while comparing strings.

- Checking whether the device area range is exceeded
- Checking whether the string is within 16383 characters

If 00H does not exist in the device area or the string exceeds 16383 characters and a character mismatch is detected, the instruction outputs comparison operation results without causing non-continuity (ENO is off).

The following example shows the operation result when the last device number of the device area is D12287.



(1) The second character of (s1) differs from that of (s2) ((s1)≠(s2)), and accordingly the operation result will be continuity (ENO is on).

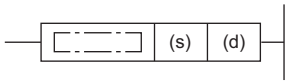
(2) D12287 and later are outside the device area, and accordingly string data comparison is performed using data up to D12287.

Since a string mismatch has been detected, the condition is satisfied and processing ends.

Concatenating string data

$+(P) [when two operands are set]

These instructions concatenate string data.

Ladder	ST
	Not supported (☞ Page 742 $+(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 742 $+(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
$+	
$+P	

Setting data

■Description, range, data type

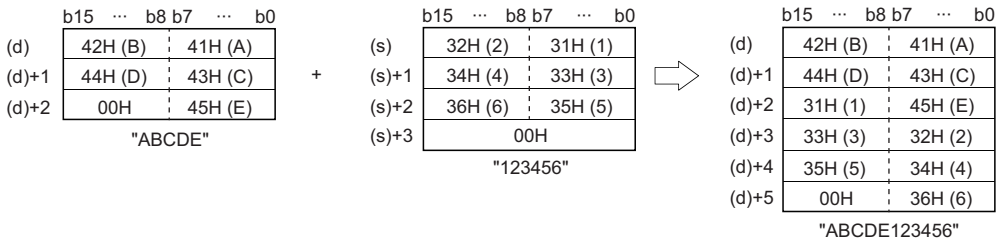
Operand	Description	Range	Data type	Data type (label)
(s)	Data to be concatenated or the start device containing the data	—	String	ANYSTRING_SINGLE
(d)	Start device where the data to be concatenated is stored	—	String	ANYSTRING_SINGLE

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- This instruction connects the string stored in the device number specified by (s) and later to the end of the string data stored in the device number specified by (d) and later, and stores the connected data in the device number specified by (d) and later.



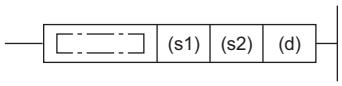
- For concatenating strings, the instruction ignores 00H that indicates the end of the string in the device specified by (d) and appends the string in the device specified by (s) following the last character in the device specified by (d).

Operation error

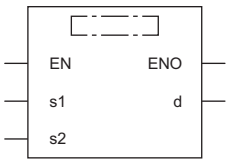
Error code (SD0)	Description
2820H	There is no NULL code (00H) in the assignment range of the relevant device/label for the device number specified by (s) and later.
	There is no NULL code (00H) in the assignment range of the relevant device/label for the device number specified by (d) and later.
2821H	The device numbers for storing the strings in the devices specified by (s) and (d) are overlapping.
3285H	The number of characters in the string specified by (s) exceeds 16383.
	The number of characters in the string specified by (d) exceeds 16383.
3286H	The number of characters in the concatenated string ((s)+(d)) exceeds 16383.
	The entire string after concatenate processing in the assignment range of the relevant device/label for the device number specified by (d) and later cannot be stored.

$+(P) [when three operands are set]

These instructions concatenate string data.

Ladder	ST
	ENO:=STRINGPLUS(EN,s1,s2,d); ENO:=STRINGPLUSP(EN,s1,s2,d);

FBD/LD



(□ is replaced by STRINGPLUS or STRINGPLUSP.)

■Execution condition

Instruction	Execution condition
$+	
$+P	

Setting data

■Descriptions, ranges, and data types

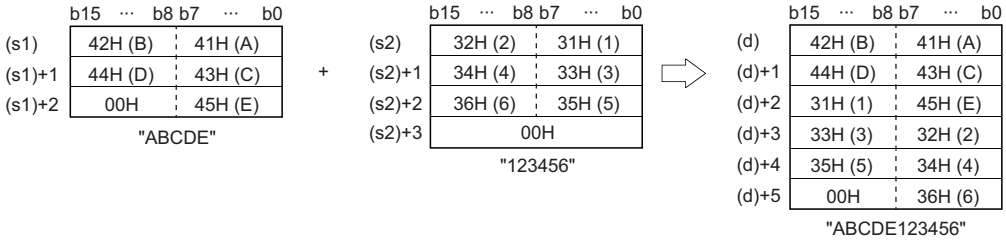
Operand	Description	Range	Data type	Data type (label)
(s1)	Data to be concatenated or the start device containing the data	—	String	ANYSTRING_SINGLE
(s2)	Data to be concatenated or the start device containing the data to be concatenated	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the concatenated data	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- This instruction connects the string stored in the device number specified by (s2) and later to the end of the string data stored in the device number specified by (s1) and later, and stores the connected data in the device number specified by (d) and later.



- For concatenating strings, the instruction ignores 00H that indicates the end of the string in the device specified by (s1) and appends the string in the device specified by (s2) following the last character in the device specified by (s1).

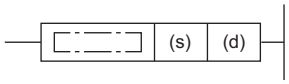
Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the assignment range of the relevant device/label for the device number specified by (s1) and later.
	There is no NULL code (00H) in the assignment range of the relevant device/label for the device number specified by (s2) and later.
2821H	The device numbers for storing the strings in the devices specified by (s2) and (d) are overlapping.
3285H	The number of characters in the string specified by (s1) exceeds 16383.
	The number of characters in the string specified by (s2) exceeds 16383.
3286H	The string stored in the device specified by (d) is out of the output enable range. • The number of characters in the concatenated string exceeds 16383. • The entire string after concatenate processing in the assignment range of the relevant device/label for the specified device number and later cannot be stored.

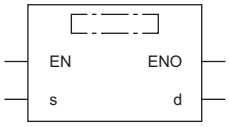
Transferring string data

$MOV(P)

These instructions transfer string data to the specified device number and later.

Ladder	ST
	ENO:=STRINGMOV(EN,s,d); ENO:=STRINGMOVP(EN,s,d);

FBD/LD



(□ is replaced by STRINGMOV or STRINGMOVP.)

■Execution condition

Instruction	Execution condition
$MOV	
$MOVP	

Setting data

■Descriptions, ranges, and data types

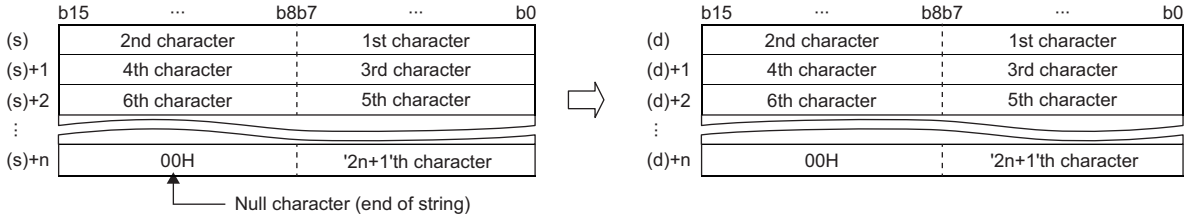
Operand	Description	Range	Data type	Data type (label)
(s)	String to be transferred (maximum of 255 characters) or the start device containing such string	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the transferred string	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

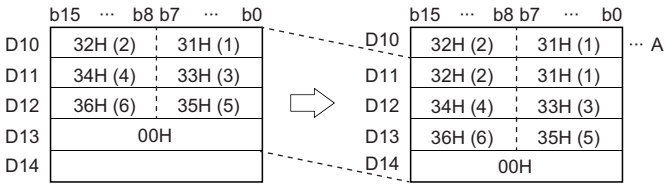
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	—	○	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions transfer the string data in the device specified by (s) to the device number specified by (d) and later. The strings specified by (s) or the strings from the device number to the device number containing 00H are transferred all at once.

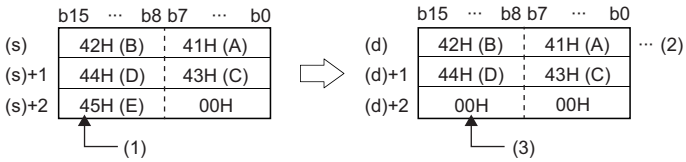


- Even when the device range ((s) to (s)+n) in which the string data to be transferred and the device range ((d) to (d)+n) for storing the transferred data are overlapping, the processing is performed normally. For example, the strings stored in the devices specified by D10 to D13 are transferred to the devices specified by D11 to D14 as shown below.



A: Data remain the same.

- When 00H is stored in the lower byte of (s)+n, 00H will be stored in both upper and lower bytes of (d)+n.



(1) Data (upper byte) is not transferred.

(2) Data remain the same.

(3) 00H is automatically stored in the upper byte.

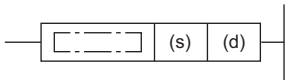
Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the assignment range of the relevant device/label for the device number specified by (s) and later.
3285H	The number of characters in the string specified by (s) exceeds 16383.
3286H	The entire string that was specified in the assignment range of the relevant device/label for the device number specified by (d) and later cannot be stored.

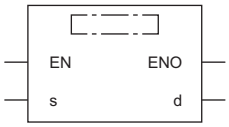
Transferring Unicode string data

$MOV(P)_WS

These instructions transfer Unicode string data to the specified device number and later.

Ladder	ST
	ENO:=STRINGMOV_WS(EN,s,d); ENO:=STRINGMOVP_WS(EN,s,d);

FBD/LD



(☐ is replaced by STRINGMOV_WS or STRINGMOVP_WS.)

■Execution condition

Instruction	Execution condition
$MOV_WS	
$MOVP_WS	

Setting data

■Descriptions, ranges, and data types

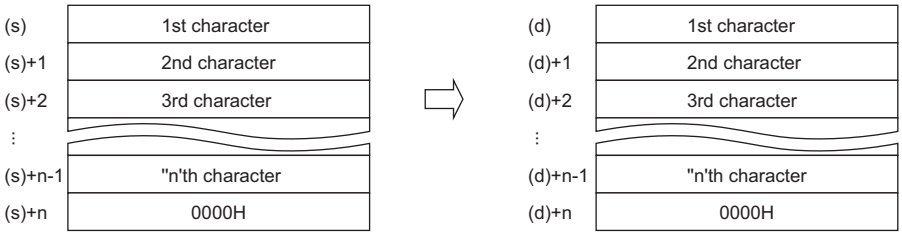
Operand	Description	Range	Data type	Data type (label)
(s)	Unicode string to be transferred (maximum of 255 characters) or the start device containing the Unicode string	—	Unicode string	ANYSTRING_DOUBLE
(d)	Start device for storing the transferred Unicode string	—	Unicode string	ANYSTRING_DOUBLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

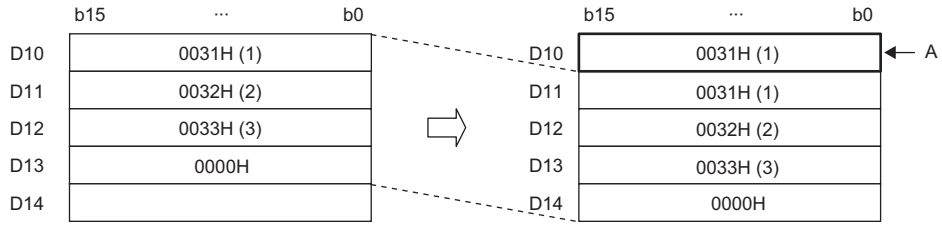
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	—	○	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions transfer the Unicode string data in the device specified by (s) to the device number specified by (d) and later. The Unicode strings specified by (s) or the Unicode strings from the device number to the device number containing 0000H are transferred all at once.



- Even when the device range ((s) to (s)+n) in which the Unicode string data to be transferred and the device range ((d) to (d)+n) for storing the transferred data are overlapping, the processing is performed normally. For example, the strings stored in the devices specified by D10 to D13 are transferred to the devices specified by D11 to D14 as shown below.



A: Same as before transfer

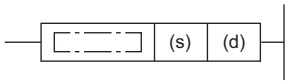
Operation error

Error code (SD0)	Description
2820H	There is no NULL code (0000H) in the assignment range of the relevant device/label for the device number specified by (s) and later.
3285H	The number of characters in the Unicode string specified by (s) exceeds 16383.
3286H	The entire Unicode string that was specified in the assignment range of the relevant device/label for the device number specified by (d) and later cannot be stored.

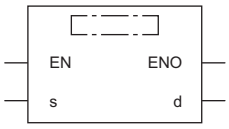
Converting 16-bit binary data to decimal ASCII

BINDA(P)(_U)

These instructions convert 16-bit binary data to the decimal ASCII code.

Ladder	ST	
	ENO:=BINDA(EN,s,d); ENO:=BINDAP(EN,s,d)	ENO:=BINDA_U(EN,s,d); ENO:=BINDAP_U(EN,s,d)

FBD/LD



■Execution condition

Instruction	Execution condition
BINDA BINDA_U	
BINDAP BINDAP_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s)	BINDA(P)	Binary data used for ASCII conversion	-32768 to 32767	16-bit signed binary	ANY16_S
	BINDA(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)		Start device for storing the conversion result	—	String	ANYSTRING_SINGLE
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the digit in decimal notation of the 16-bit binary data in the device specified by (s) to the ASCII code, and store the converted data in the device number specified by (d) and later.
- The format of the decimal ASCII data to be stored in (d) depends on the status of SM705 (Number of conversion digits selection).

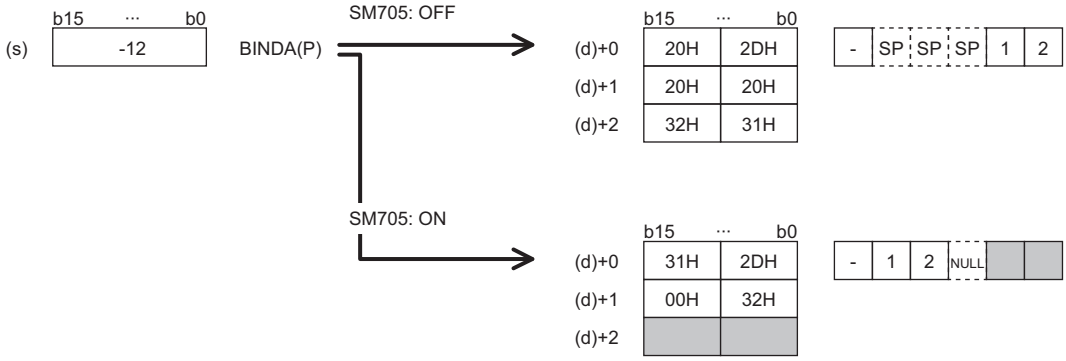
Status of SM705	Storage format of (d)	Reference
Off	Data is stored in a fixed number of digits (a sign + 5 digits)	Page 750 Operation of when SM705 (Number of conversion digits selection) is off
On	Each digit is stored left-justified depending on the value of (s).	Page 751 Operation of when SM705 (Number of conversion digits selection) is on

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

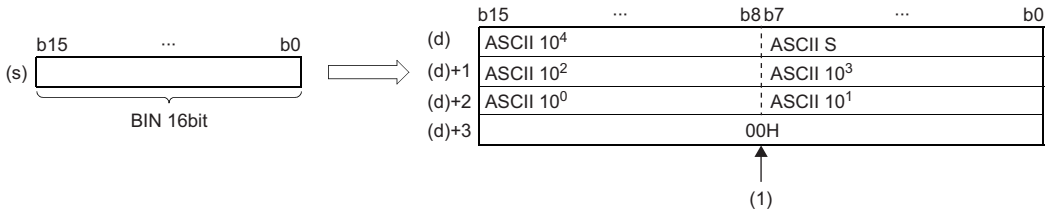
When the BINDA(P) instruction is executed with the numeric value "-12" stored in (s)



- When SM705 is off, the number of digits is fixed. The first character is a sign and it is 2DH(-) in the above example. (If (s) is 0 or positive, the first character is 20H (space).) The numeric part is right-justified. When the length of the numeric part is less than 5 digits, the ASCII code 20H (space) is stored for the ASCII code of the upper digit(s).
- When SM705 is on, data is left-justified. When the length of the numeric part is less than 5 digits, 00H is stored in the end.

■Operation of when SM705 (Number of conversion digits selection) is off

Decimal ASCII data is stored in a fixed number of digits in (d) to (d)+2.



ASCII S: Sign data of ASCII code*1

ASCII 10^4: Ten-thousands place of ASCII code*2

ASCII 10^3: Thousands place of ASCII code*3

ASCII 10^2: Hundreds place of ASCII code*4

ASCII 10^1: Tens place of ASCII code*5

ASCII 10^0: Ones place of ASCII code

(1): 00H is stored in (d)+3 when SM701 (Number of output characters selection) is off. When it is on, the value remains unchanged.

*1 When the value is 0 or positive, 20H (space) is stored. When the value is negative, 2DH (-) is stored.

*2 When the length of the numeric part is 4 digits or less, 20H (space) is stored in ASCII 10^4.

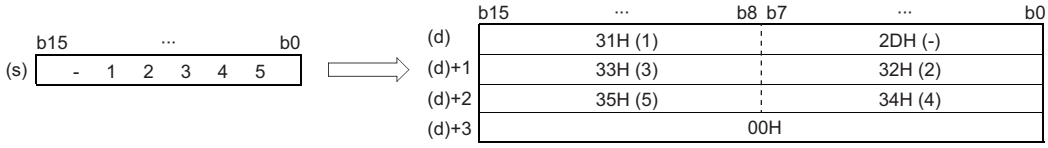
*3 When the length of the numeric part is 3 digits or less, 20H (space) is stored in ASCII 10^3.

*4 When the length of the numeric part is 2 digits or less, 20H (space) is stored in ASCII 10^2.

*5 When the length of the numeric part is 1 digit, 20H (space) is stored in ASCII 10^1.

Ex.

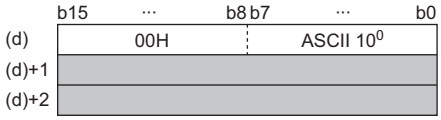
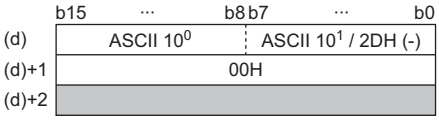
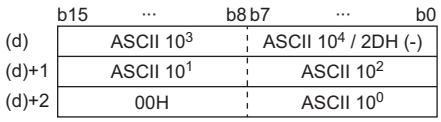
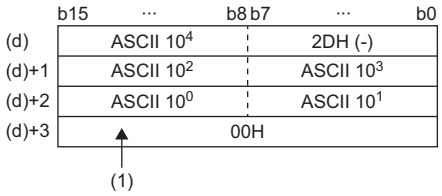
-12345 is set in (s) when the BINDA(P) instruction is used.



■Operation of when SM705 (Number of conversion digits selection) is on

Decimal ASCII data is stored right-justified in (d).

The following figures show an example of a value of (s) and a value stored in (d).

<table>
<tr><th>Value of (s)</th><th>Data of (d) to (d)+2</th><th>Value of (s)</th><th>Data of (d) to (d)+2</th></tr>
<tr>
<td>• 0
• Positive value (1 digit in numeric part)</td>
<td>• The upper byte of (d) is filled with 00H.
• Data in (d)+1 and (d)+2 remains unchanged.

</td>
<td>• Positive value (2 digits in numeric part)
• Negative value (1 digit in numeric part)</td>
<td>• (d)+1 is filled with 00H.
• Data in (d)+2 remains unchanged.

</td>
</tr>
<tr><td colspan="4">⋮</td></tr>
<tr>
<td>• Positive value (5 digits in numeric part)
• Negative value (4 digits in numeric part)</td>
<td>• The upper byte of (d)+2 is filled with 00H.

</td>
<td>• Negative value (5 digits in numeric part)</td>
<td>• (1): (d)+3 is filled with 00H only when SM701 (Number of output characters selection) is off.

</td>
</tr>
</table>

ASCII 10^4: Ten-thousands place of ASCII code
ASCII 10^3: Thousands place of ASCII code
ASCII 10^2: Hundreds place of ASCII code
ASCII 10^1: Tens place of ASCII code
ASCII 10^0: Ones place of ASCII code

- When the number of operation digits is less than the maximum number of digits (sign + 5 digits), 00H is stored in the end of the string regardless of the status (on/off) of SM701 (Number of output characters selection). If the end of the string is the lower byte, 00H is also stored in the upper byte.
- When the number of operation digits is equal to the maximum number of digits (a sign + 5 digits), 00H is stored in (d)+3 when SM701 is off. (d)+3 remains unchanged if SM701 is on.

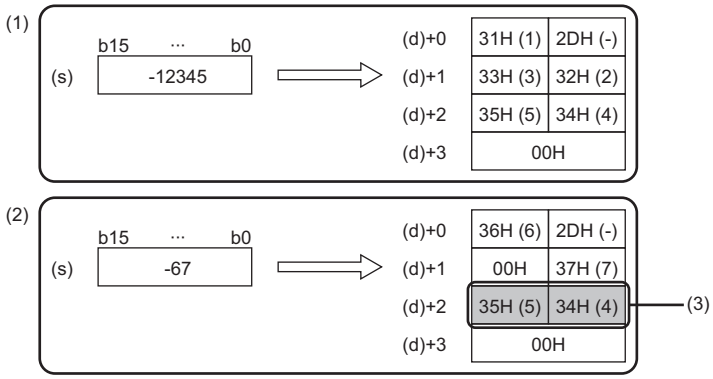
Operation error

There is no operation error.

Precautions

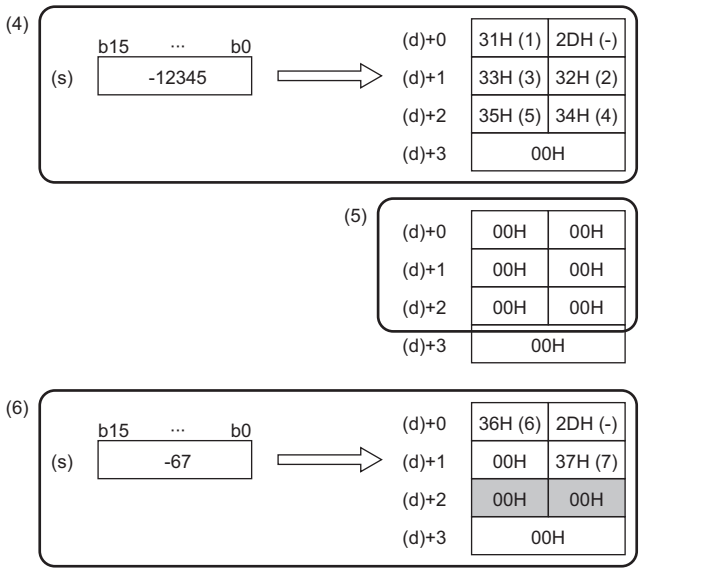
When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the BINDA(P)(_U) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation result may not be overwritten by the succeeding result and can remain in (d).

[Example] Executing the BINDA(P) instruction when (s) is "-12345" and then executing another BINDA(P) instruction when (s) is "-67"



(1) "-12345" is converted into a string.
(2) "-67" is converted into a string.
(3) A part of the previous conversion result remains in (d)+2.

To avoid this, create a program to clear the entire data storage areas (d)+0 to (d)+2 before executing the BINDA(P)(_U) instruction.

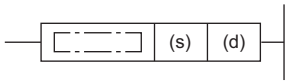


(4) "-12345" is converted into a string.
(5) (d)+0 to (d)+2 are cleared.
(6) "-67" is converted into a string.

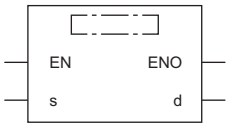
Converting 32-bit binary data to decimal ASCII

DBINDA(P)(_U)

These instructions convert 32-bit binary data to the decimal ASCII code.

Ladder	ST	
	ENO:=DBINDA(EN,s,d); ENO:=DBINDAP(EN,s,d);	ENO:=DBINDA_U(EN,s,d); ENO:=DBINDAP_U(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBINDA DBINDA_U	
DBINDAP DBINDAP_U	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s)	DBINDA(P)	Binary data used for ASCII conversion	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DBINDA(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)		Start device for storing the conversion result	—	String	ANYSTRING_SINGLE
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the digit in decimal notation of the 32-bit binary data in the device specified by (s) to the ASCII code, and store the converted data in the device number specified by (d) and later.
- The format of the decimal ASCII data to be stored in (d) depends on the status of SM705 (Number of conversion digits selection).

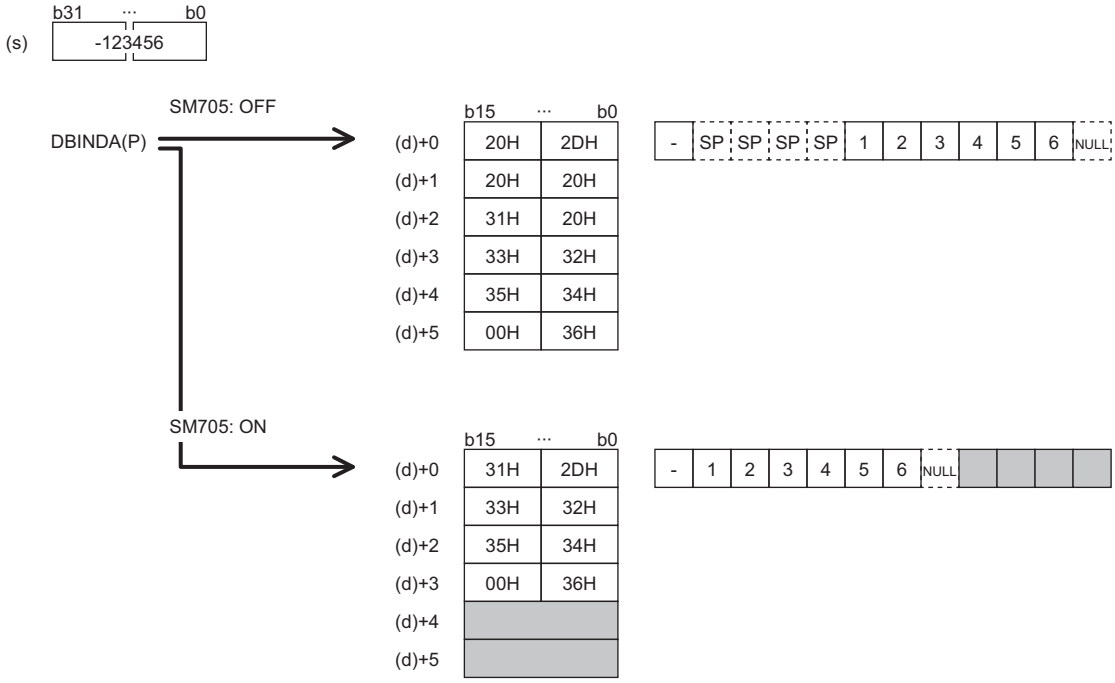
Status of SM705	Storage format of (d)	Reference
Off	Data is stored in a fixed number of digits (a sign + 10 digits)	Page 755 Operation of when SM705 (Number of conversion digits selection) is off
On	Each digit is stored left-justified depending on the value of (s).	Page 756 Operation of when SM705 (Number of conversion digits selection) is on

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

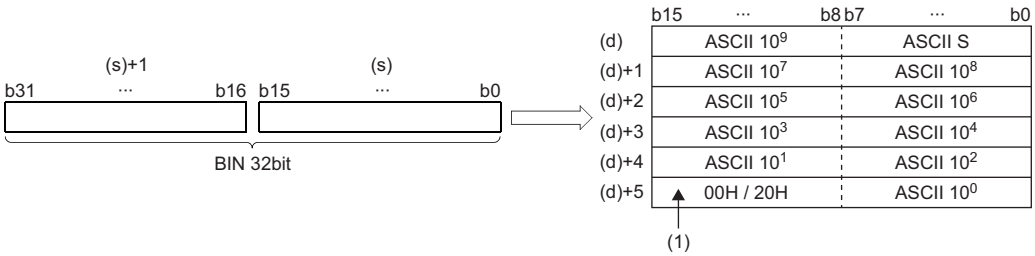
When the DBINDA(P) instruction is executed with a numeric value "-123456" stored in (s)



- When SM705 is off, the number of digits is fixed. The first character is a sign and it is 2DH(-) in the above example. (If (s) is 0 or positive, the first character is 20H (space).) The numeric part is right-justified. When the length of the numeric part is less than 10 digits, the ASCII code 20H (space) is stored for the ASCII code of the upper digit(s).
- When SM705 is on, data is left-justified. When the length of the numeric part is less than 10 digits, 00H is stored in the end.

■Operation of when SM705 (Number of conversion digits selection) is off

Decimal ASCII data is stored in a fixed number of digits in (d) to (d)+5.



ASCII S: Sign data of ASCII code*1

ASCII 10^9: Billions place of ASCII code*2

ASCII 10^8: Hundred-millions place of ASCII code*3

⋮

ASCII 10^1: Tens place of ASCII code*4

ASCII 10^0: Ones place of ASCII code

(1): 00H is stored in the upper byte of (d)+5 only when SM701 (Number of output characters selection) is off, and 20H (space) is stored when it is on.

*1 When the value is 0 or positive, 20H (space) is stored. When the value is negative, 2DH (-) is stored.

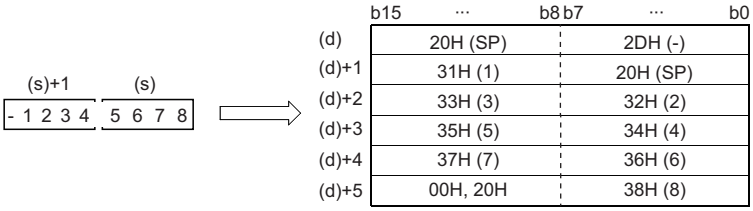
*2 When the length of the numeric part is 9 digits or less, 20H (space) is stored in ASCII 10^9.

*3 When the length of the numeric part is 8 digits or less, 20H (space) is stored in ASCII 10^8.

*4 When the length of the numeric part is 1 digit, 20H (space) is stored in ASCII 10^1.

Ex.

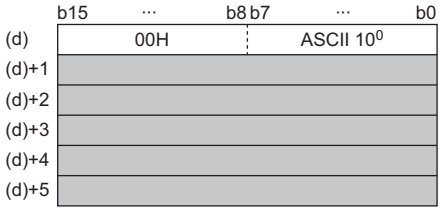
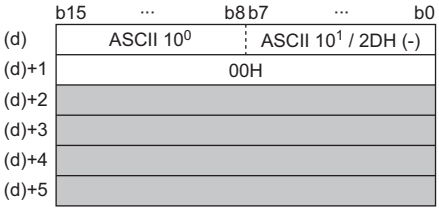
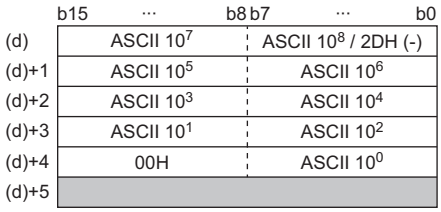
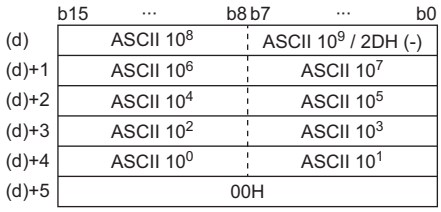
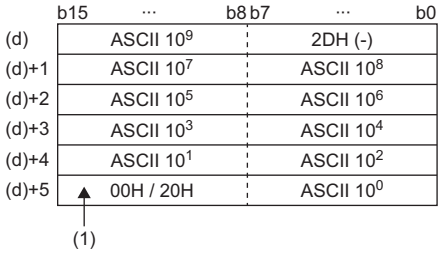
When -12345678 (signed) is specified in (s)



■Operation of when SM705 (Number of conversion digits selection) is on

Decimal ASCII data is stored right-justified in (d).

The following figures show an example of a value of (s) and a value stored in (d).

<table>
<tr><th>Value of (s)</th><th>Data of (d) to (d)+2</th><th>Value of (s)</th><th>Data of (d) to (d)+2</th></tr>
<tr><td>• 0
• Positive value (1 digit in numeric part)</td><td>• The upper byte of (d) is filled with 00H.
• (d)+1 and later remain unchanged.

</td><td>• Positive value (2 digits in numeric part)
• Negative value (1 digit in numeric part)</td><td>• (d)+1 is filled with 00H.
• (d)+2 and later remain unchanged.

</td></tr>
<tr><td colspan="4">⋮</td></tr>
<tr><td>• Positive value (9 digits in numeric part)
• Negative value (8 digits in numeric part)</td><td>• The upper byte of (d)+4 is filled with 00H.
• (d)+5 and later remain unchanged.

</td><td>• Positive value (10 digits in numeric part)
• Negative value (9 digits in numeric part)</td><td>• (d)+5 is filled with 00H.

</td></tr>
<tr><td>• Negative value (10 digits in numeric part)</td><td>• 00H is stored in the upper byte (1) of (d)+5 when SM701 (Number of output characters selection) is off, and 20H (space) is stored when it is on.

</td><td colspan="2">ASCII 10^9: Billions place of ASCII code
ASCII 10^8: Hundred-millions place of ASCII code
⋮
ASCII 10^1: Tens place of ASCII code
ASCII 10^0: Ones place of ASCII code</td></tr>
</table>

- When the number of operation digits is less than the maximum number of digits (a sign + 10 digits), 00H is stored in the end of the string regardless of the status (on/off) of SM701 (Number of output characters selection). If the end of the string is the lower byte, 00H is also stored in the upper byte.
- When the number of operation digits is equal to the maximum number of digits (a sign + 10 digits), 00H is stored in the upper byte of (d)+5 if SM701 is off. 20H (space) is stored in the upper byte of (d)+5 if SM701 is on.

Operation error

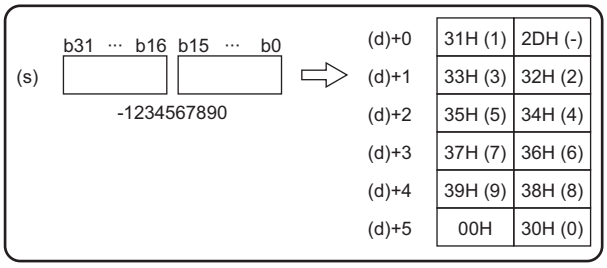
There is no operation error.

Precautions

When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the DBINDA(P)(_U) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation result may not be overwritten by the succeeding result and can remain in (d).

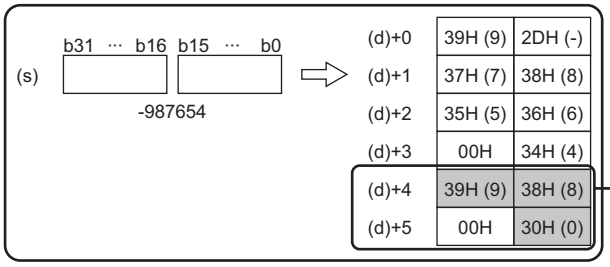
[Example] Executing the DBINDA(P) instruction when (s) is "-1234567890" and then executing another DBINDA(P) instruction when (s) is "-987654"

(1)



(d)+0	31H (1)	2DH (-)
(d)+1	33H (3)	32H (2)
(d)+2	35H (5)	34H (4)
(d)+3	37H (7)	36H (6)
(d)+4	39H (9)	38H (8)
(d)+5	00H	30H (0)

(2)



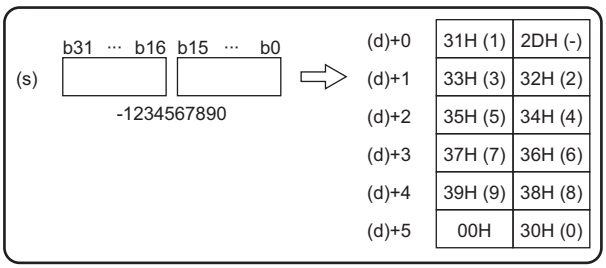
(d)+0	39H (9)	2DH (-)
(d)+1	37H (7)	38H (8)
(d)+2	35H (5)	36H (6)
(d)+3	00H	34H (4)
(d)+4	39H (9)	38H (8)
(d)+5	00H	30H (0)

(3)

(1) "-1234567890" is converted into a string.
(2) "-987654" is converted into a string.
(3) A part of the previous conversion result remains in (d)+4 and (d)+5.

To avoid this, create a program to clear the entire data storage areas (d)+0 to (d)+5 before executing the DBINDA(P)(_U) instruction.

(4)

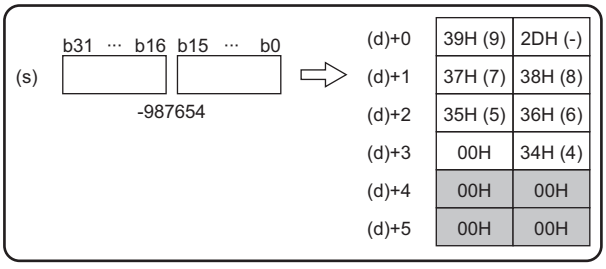


(d)+0	31H (1)	2DH (-)
(d)+1	33H (3)	32H (2)
(d)+2	35H (5)	34H (4)
(d)+3	37H (7)	36H (6)
(d)+4	39H (9)	38H (8)
(d)+5	00H	30H (0)

(5)

(d)+0	00H	00H
(d)+1	00H	00H
(d)+2	00H	00H
(d)+3	00H	00H
(d)+4	00H	00H
(d)+5	00H	00H

(6)



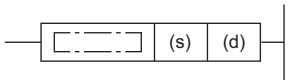
(d)+0	39H (9)	2DH (-)
(d)+1	37H (7)	38H (8)
(d)+2	35H (5)	36H (6)
(d)+3	00H	34H (4)
(d)+4	00H	00H
(d)+5	00H	00H

(4) "-1234567890" is converted into a string.
(5) (d)+0 to (d)+5 are cleared.
(6) "-987654" is converted into a string.

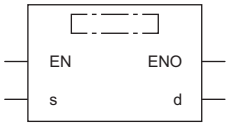
Converting 16-bit binary data to hexadecimal ASCII

BINHA(P)

These instructions convert 16-bit binary data to the hexadecimal ASCII code.

Ladder	ST
	ENO:=BINHA(EN,s,d); ENO:=BINHAP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BINHA	
BINHAP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data used for ASCII conversion	-32768 to 32767	16-bit signed binary	ANY16
(d)	Start device for storing the conversion result	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the digit in hexadecimal notation of the 16-bit binary data in the device specified by (s) to the ASCII code, and store the converted data in the device number specified by (d) and later.
- The format of the hexadecimal ASCII data to be stored in (d) depends on the status of SM705 (Number of conversion digits selection).

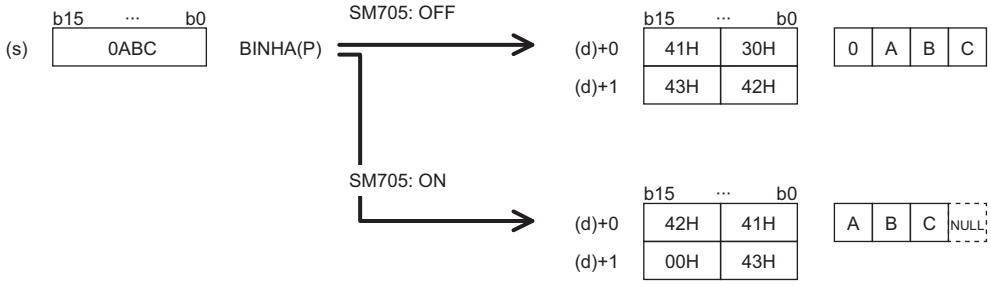
Status of SM705	Storage format of (d)	Reference
Off	Data is stored in a fixed number of digits (4 digits)	Page 760 Operation of when SM705 (Number of conversion digits selection) is off
On	Each digit is stored left-justified depending on the value of (s).	Page 760 Operation of when SM705 (Number of conversion digits selection) is on

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

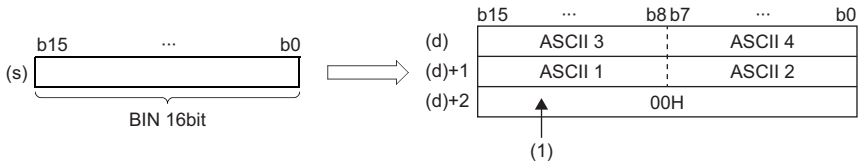
When the BINHA(P) instruction is executed with the 16-bit binary data "0ABCH" stored in (s)



- When SM705 is off, the number of digits is fixed. Four digits of "0ABC" are converted into ASCII data and stored.
- When SM705 is on, data is left-justified. "0ABC" with the leading "0" omitted ("ABC") are converted into ASCII data and stored, and 00H is stored in the end.

■Operation of when SM705 (Number of conversion digits selection) is off

Hexadecimal ASCII data is stored in a fixed number of digits (4 digits) in (d).



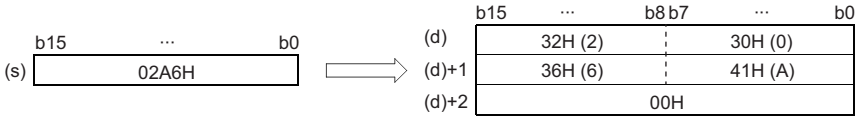
ASCII □: ASCII code (□th digit)

(1): 00H is stored in (d)+2 when SM701 (Number of output characters selection) is off. When it is on, the value remains unchanged.

- The operation result to be stored in the device specified by (d) is processed as a 4-digit hexadecimal number. Therefore, 0 at the left side of the effective number of digits is processed as "0". (Zero padding)

Ex.

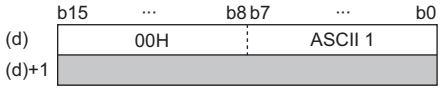
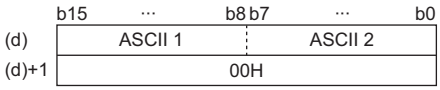
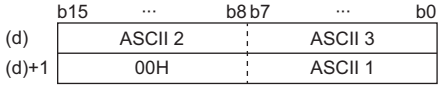
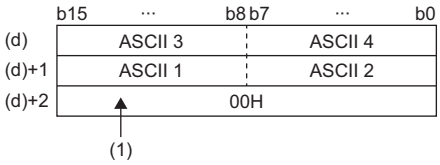
When 02A6H is stored in the device specified by (s)



■Operation of when SM705 (Number of conversion digits selection) is on

Hexadecimal ASCII data for the number of digits (up to 4 digits) without the leftmost "0" in the effective digits is stored right-justified in (d).

The following figures show an example of a value of (s) and a value stored in (d).

Value of (s)	Data of (d) to (d)+1	Value of (s)	Data of (d) to (d)+1
• 0H to FH	• The upper byte of (d) is filled with 00H. • (d)+1 and later remain unchanged. 	• 10H to FFH	• (d)+1 is filled with 00H. 
• 100H to FFFH	• The upper byte of (d)+1 are filled with 00H. 	• 1000H to FFFFH	• (1): (d)+2 is filled with 00H when SM701 (Number of output characters selection) is off. (d)+2 remains unchanged if SM701 is on. 

ASCII □: ASCII code (□th digit)

- When the number of digits is less than the maximum number of digits (4 digits), 00H is stored in the end of the string regardless of the status (on/off) of SM701 (Number of output characters selection). If the end of the string is the lower byte, 00H is also stored in the upper byte.
- When the number of digits is equal to the maximum number of digits (4 digits), 00H is stored in (d)+2 when SM701 is off. (d)+2 remains unchanged if SM701 is on.

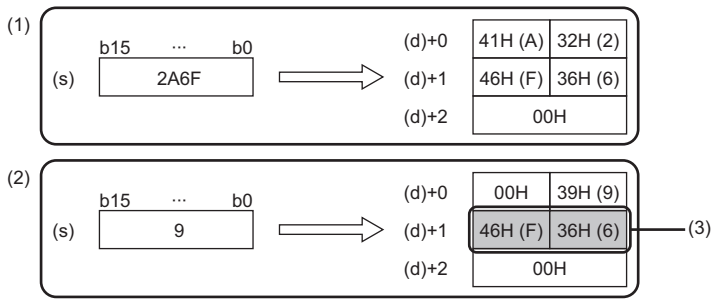
Operation error

There is no operation error.

Precautions

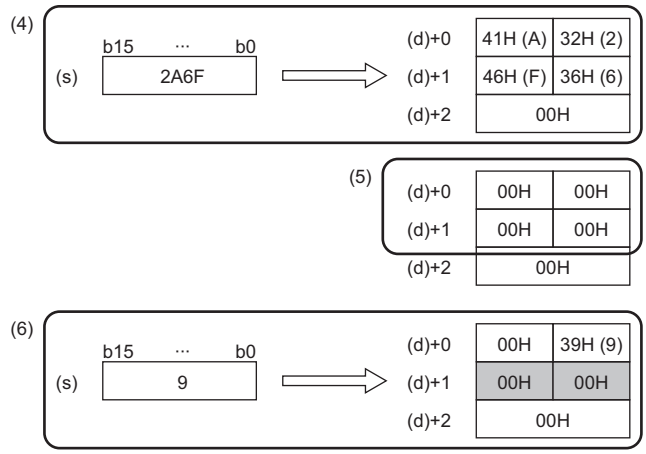
When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the BINHA(P) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation result may not be overwritten by the succeeding result and can remain in (d).

[Example] Executing the BINHA(P) instruction when (s) is "2A6F" and then executing another BINHA(P) instruction when (s) is "9"



(1) "2A6F" is converted into a string.
(2) "9" is converted into a string.
(3) A part of the previous conversion result remains in (d)+1.

To avoid this, create a program to clear the entire data storage areas (d)+0 and (d)+1 before executing the BINHA(P) instruction.

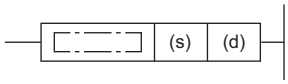


(4) "2A6F" is converted into a string.
(5) (d)+0 to (d)+1 are cleared.
(6) "9" is converted into a string.

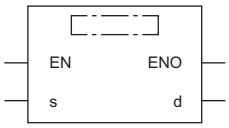
Converting 32-bit binary data to hexadecimal ASCII

DBINHA(P)

These instructions convert 32-bit binary data to the hexadecimal ASCII code.

Ladder	ST
	ENO:=DBINHA(EN,s,d); ENO:=DBINHAP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBINHA	
DBINHAP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Binary data used for ASCII conversion	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Start device for storing the conversion result	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the digit at each place in hexadecimal notation of the 32-bit binary data specified by (s) to the ASCII code, and store the converted data in the device number specified by (d) and later.
- The format of the decimal ASCII data to be stored in (d) depends on the status of SM705 (Number of conversion digits selection).

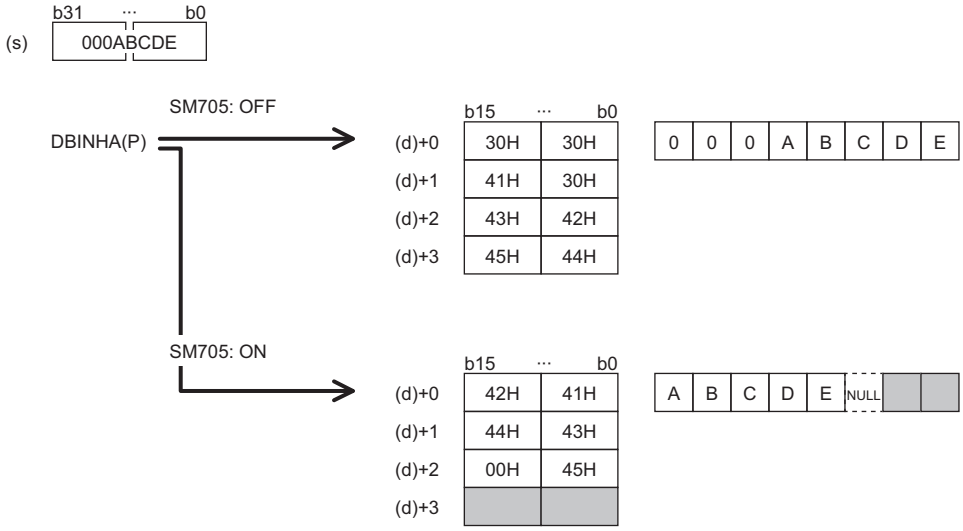
Status of SM705	Storage format of (d)	Reference
Off	Data is stored in a fixed number of digits (8 digits)	Page 764 Operation of when SM705 (Number of conversion digits selection) is off
On	Each digit is stored left-justified depending on the value of (s).	Page 764 Operation of when SM705 (Number of conversion digits selection) is on

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

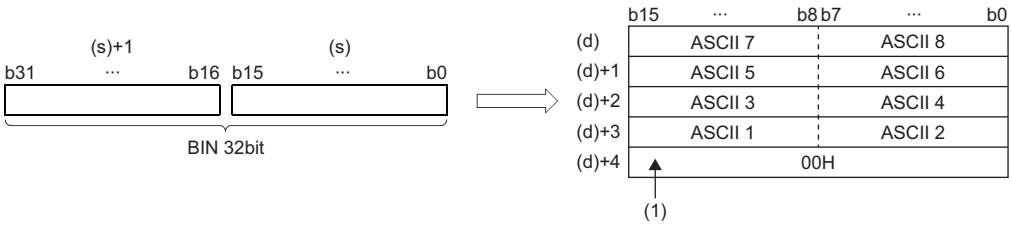
When the DBINHA(P) instruction is executed with a 32-bit binary data "000ABCDEH" stored in (s)



- When SM705 is off, the number of digits is fixed. Eight digits of "000ABCDE" are converted into ASCII data and stored.
- When SM705 is on, data is left-justified. The effective digits "000ABCDE" with the leading "0"s omitted ("ABCDE") are converted into ASCII data and stored, and 00H is stored in the end.

■Operation of when SM705 (Number of conversion digits selection) is off

Hexadecimal ASCII data is stored in a fixed number of digits (8 digits) in (d) to (d)+3.



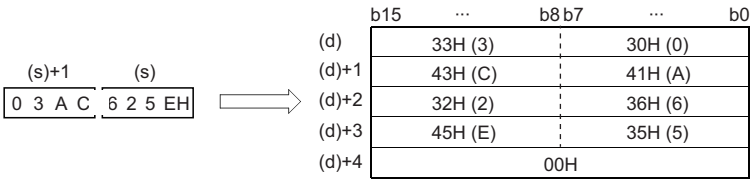
ASCII □: ASCII code (□th digit)

(1): 00H is stored in (d)+4 when SM701 (Number of output characters selection) is off. When it is on, the value remains unchanged.

- The operation result to be stored in the device specified by (d) is processed as an 8-digit hexadecimal number. Therefore, 0 at the left side of the effective number of digits is processed as "0". (Zero padding)

Ex.

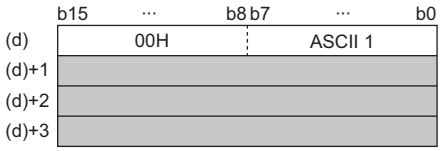
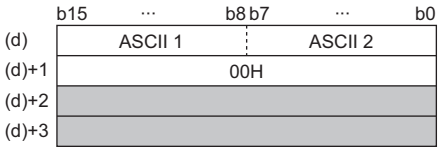
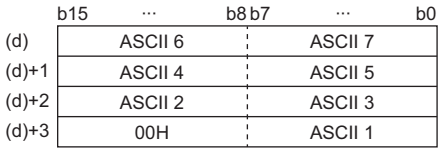
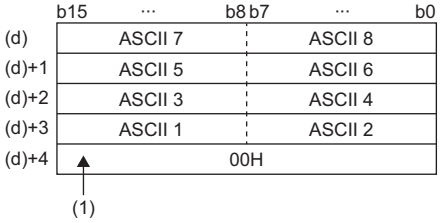
When 03AC625EH is stored in the device specified by (s)



■Operation of when SM705 (Number of conversion digits selection) is on

Hexadecimal ASCII data for the number of digits (up to 8 digits) without the leftmost "0" in the effective digits is stored right-justified in (d).

The following figures show an example of a value of (s) and a value stored in (d).

Value of (s)	Data of (d) to (d)+3	Value of (s)	Data of (d) to (d)+3
• 0H to FH	• The upper byte of (d) is filled with 00H. • (d)+1 and later remain unchanged. 	• 10H to FFH	• (d)+1 is filled with 00H. • (d)+2 and later remain unchanged. 
⋮			
• 1000000H to FFFFFFFH	• The upper byte of (d)+3 is filled with 00H. 	• 10000000H to FFFFFFFFH	• (1): (d)+4 is filled with 00H when SM701 (Number of output characters selection) is off. (d)+4 does not change if SM701 (Number of output characters selection) is on. 

ASCII □: ASCII code (□th digit)

- When the number of digits is less than the maximum number of digits (8 digits), 00H is stored in the end of the string regardless of the status (on/off) of SM701 (Number of output characters selection). If the end of the string is the lower byte, 00H is also stored in the upper byte.
- When the number of digits is equal to the maximum number of digits (8 digits), 00H is stored in (d)+4 when SM701 (Number of output characters selection) is off. (d)+4 does not change if SM701 (Number of output characters selection) is on.

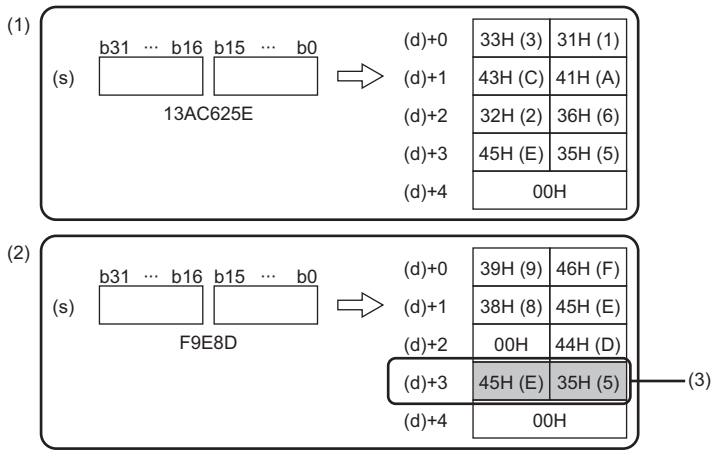
Operation error

There is no operation error.

Precautions

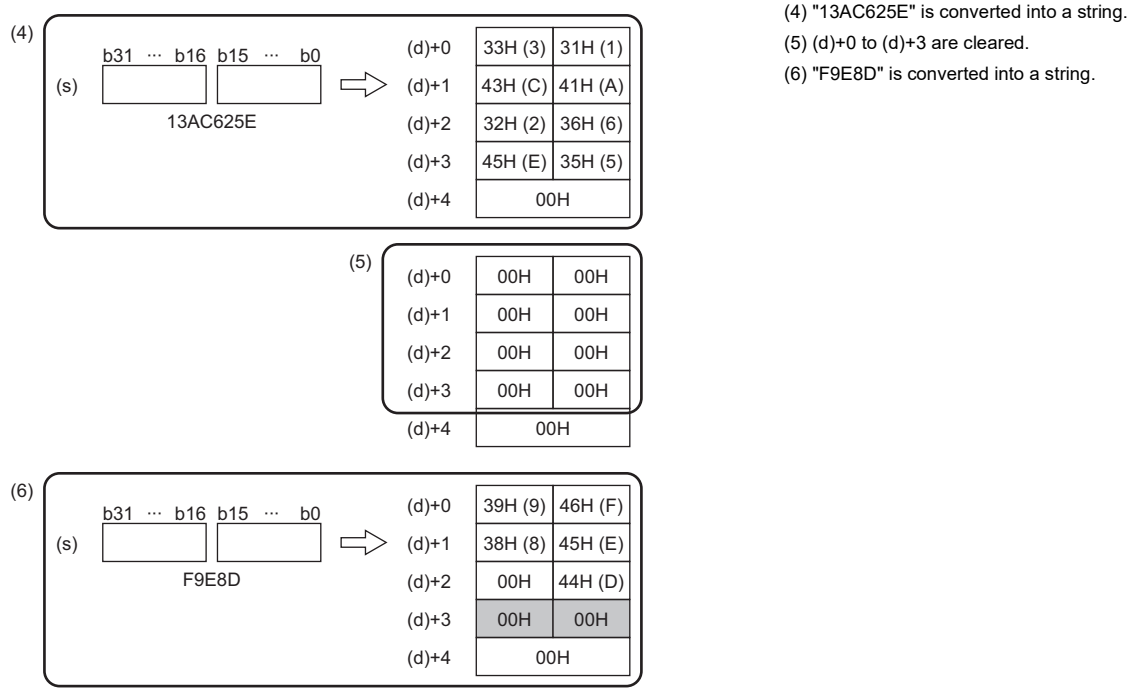
When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the DBINHA(P) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation result may not be overwritten by the succeeding result and can remain in (d).

[Example] Executing the DBINHA(P) instruction when (s) is "13AC625E" and then executing another DBINHA(P) instruction when (s) is "F9E8D"



(1) "13AC625E" is converted into a string.
(2) "F9E8D" is converted into a string.
(3) A part of the previous conversion result remains in (d)+3.

To avoid this, create a program to clear the entire data storage areas (d)+0 to (d)+3 before executing the DBINHA(P) instruction.

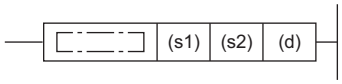


(4) "13AC625E" is converted into a string.
(5) (d)+0 to (d)+3 are cleared.
(6) "F9E8D" is converted into a string.

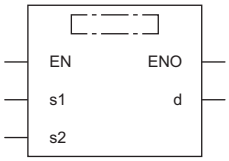
Converting 16-bit binary data to string data

STR(P)(_U)

These instructions convert 16-bit binary data to a string by adding a decimal point to the specified place of the data.

Ladder	ST	
	ENO:=STR(EN,s1,s2,d); ENO:=STRP(EN,s1,s2,d);	ENO:=STR_U(EN,s1,s2,d); ENO:=STRP_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
STR STR_U	
STRP STRP_U	

Setting data

■Description, range, data type

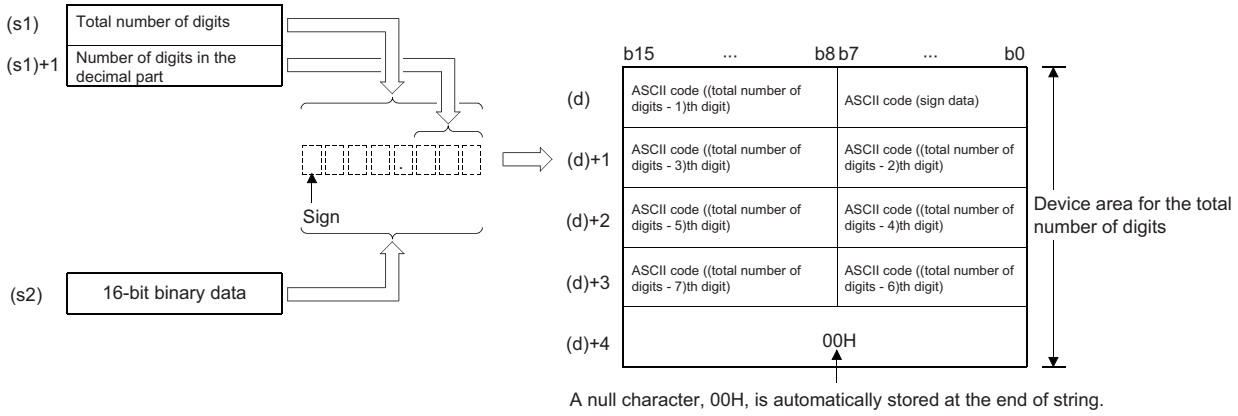
Operand		Description	Range	Data type	Data type (label)
(s1)	STR(P)	Start device where the number of digits of the conversion target data is stored	—	16-bit signed binary	ANY16_S_ARRAY (Number of elements: 2)
	STR(P)_U			16-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 2)
(s2)	STR(P)	Conversion target data	-32768 to 32767	16-bit signed binary	ANY16_S
	STR(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)		Start device for storing the converted string	—	String	ANYSTRING_SINGLE
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

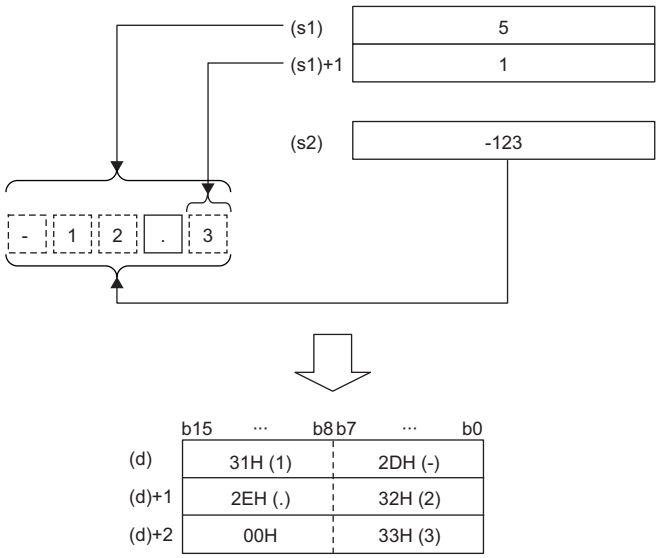
Processing details

- These instructions add a decimal point to the 16-bit binary data in the device specified by (s2) at the location specified by (s1), convert the data to string data, and store the converted data in the device number specified by (d) and later.



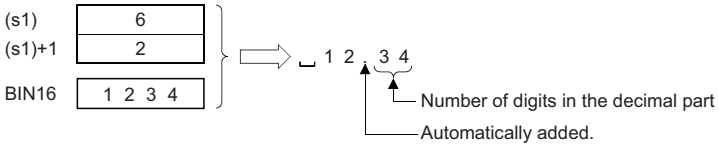
Ex.

When converting data "-123" in (s2) into a string assuming that the number of decimal places is one ("-12.3")

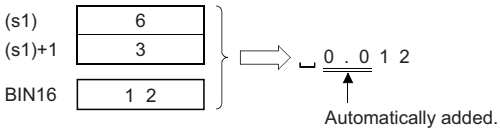


- When storing the decimal part in (d), the total number of digits that can be specified by (s1) is 4 to 8. The number of digits in the decimal part that can be specified by (s1)+1 is 1 to 5. Note that the number of digits in the decimal part must be equal to or smaller than the total number of digits minus 3.
- When not storing the decimal part in (d), the total number of digits that can be specified by (s1) is 2 to 8. The number of digits in the decimal part that can be specified by (s1)+1 cannot be set.

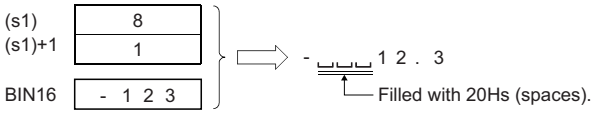
- The converted string data are stored in the device number specified by (d) and later as shown below.
  - As sign data, "20H" (space) is stored if the 16-bit binary data is positive, and "2DH" (-) is stored if the data is negative.
  - If the number of digits in the decimal part is set to other than 0, "2EH" (.) is automatically stored at the position of the specified number of digits plus 1. If the number of digits in the decimal part is 0, "2EH" (.) is not stored.



  - If the specified number of digits in the decimal part is greater than the number of digits of the 16-bit binary data, 0s are automatically added and the data is regarded as "0.□□□□".



  - If the total number of digits excluding the sign and the decimal point is greater than the number of digits of the 16-bit binary data, "20H" (space) is stored between the sign and the numeric value. If the number of digits of the 16-bit binary data is greater, an error occurs.



  - The value "00H" is automatically stored at the end of the converted string.

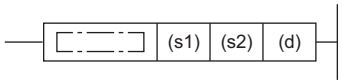
Operation error

Error code (SD0)	Description
3281H	Invalid data that cannot be converted is input to (s1). • The specified total number of digits is out of the range, 2 to 8. • The specified number of digits in the decimal part of (s1)+1 is out of the range, 0 to 5. • When the specified number of digits in the decimal part of (s1)+1 is in the range 1 to 5, the relationship between the total number of digits specified by (s1) and the number of digits in the decimal part specified by (s1)+1 does not satisfy the following. (Total number of digits)-3≥Number of digits in the decimal part • The number of digits specified by (s1) is smaller than the number of digits plus 2 of the 16-bit binary data specified by (s2). [Number of digits in (s1)] < [Number of digits of 16-bit binary data excluding the sign in (s2) + Number of digits in the sign (+ or -) + Number of digits of decimal point (.)]
3286H	The entire string that was converted in the assignment range of the relevant device/label for the device number specified by (d) and later cannot be stored.

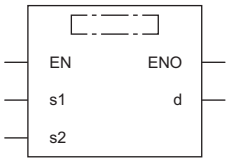
Converting 32-bit binary data to string data

DSTR(P)(_U)

These instructions convert 32-bit binary data to a string by adding a decimal point to the specified place of the data.

Ladder	ST	
	ENO:=DSTR(EN,s1,s2,d); ENO:=DSTRP(EN,s1,s2,d);	ENO:=DSTR_U(EN,s1,s2,d); ENO:=DSTRP_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSTR DSTR_U	
DSTRP DSTRP_U	

Setting data

■Description, range, data type

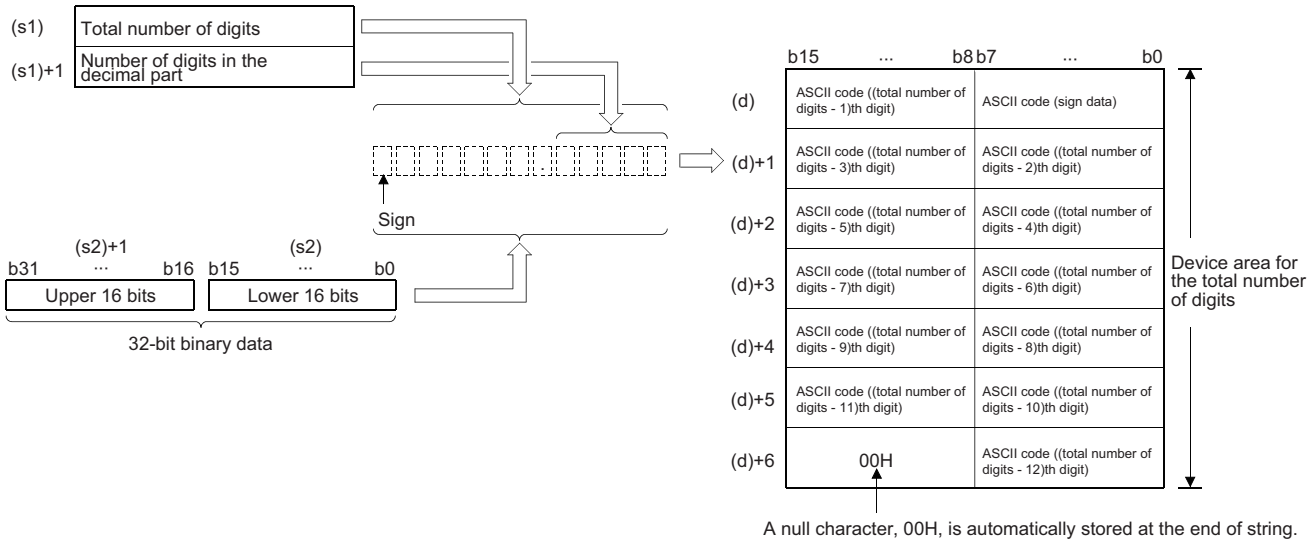
Operand		Description	Range	Data type	Data type (label)
(s1)	DSTR(P)	Start device where the number of digits of the conversion target data is stored	—	16-bit signed binary	ANY16_S_ARRAY (Number of elements: 2)
	DSTR(P)_U			16-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 2)
(s2)	DSTR(P)	Conversion target data	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DSTR(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)		Start device for storing the converted string	—	String	ANYSTRING_SINGLE
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

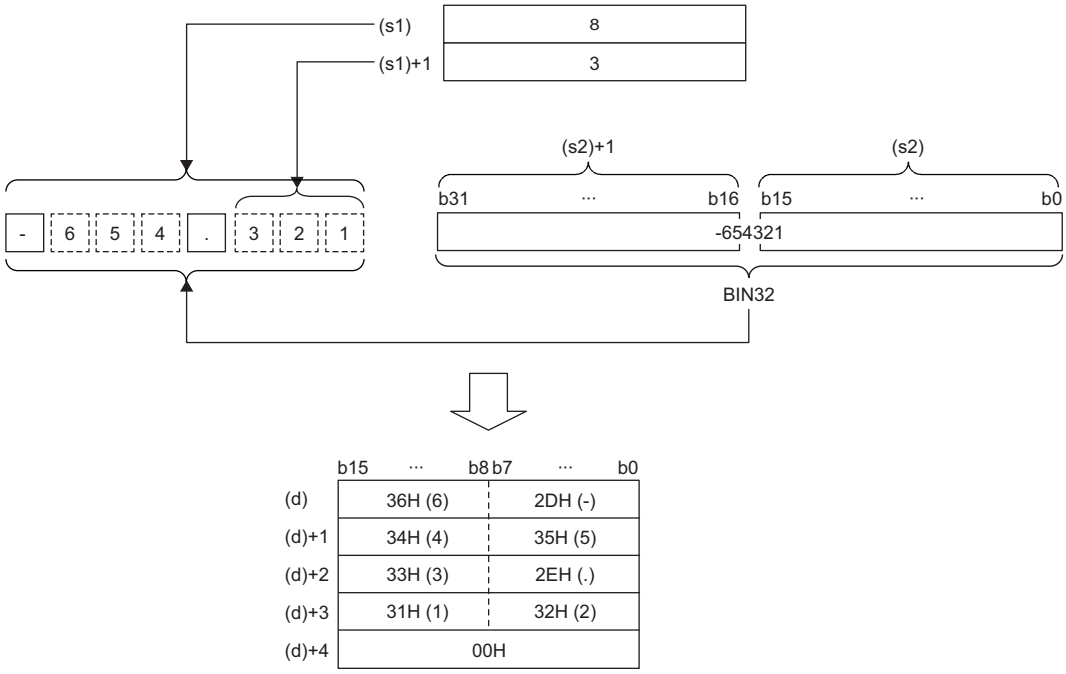
Processing details

- These instructions add a decimal point to the 32-bit binary data in the device specified by (s2) at the location specified by (s1), convert the data to string data, and store the converted data in the device number specified by (d) and later.



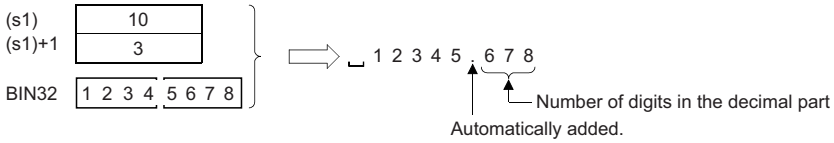
Ex.

When converting data "-654321" in (s2) into a string assuming that the number of decimal places is three ("-654.321")

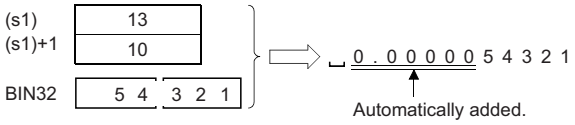


- When storing the decimal part in (d), the total number of digits that can be specified by (s1) is 4 to 13. The number of digits in the decimal part that can be specified by (s1)+1 is 1 to 10. Note that the number of digits in the decimal part must be equal to or smaller than the total number of digits minus 3.
- When not storing the decimal part in (d), the total number of digits that can be specified by (s1) is 2 to 13. The number of digits in the decimal part that can be specified by (s1)+1 cannot be set.

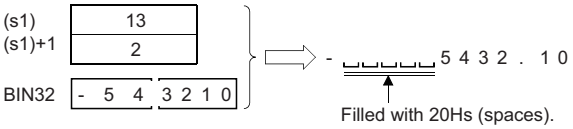
- The converted string data are stored in the device number specified by (d) and later as shown below.
  - As sign data, 20H (space) is stored if the 32-bit binary data is positive, and 2DH (-) is stored if the data is negative.
  - If the number of digits in the decimal part is set to other than 0, "2EH" (.) is automatically stored at the position of the specified number of digits plus 1. If the number of digits in the decimal part is 0, "2EH" (.) is not stored.



  - If the specified number of digits in the decimal part is greater than the number of digits of the 32-bit binary data, 0s are automatically added and the data is regarded as "0.□□□□".



  - If the total number of digits excluding the sign and the decimal point is greater than the number of digits of the 32-bit binary data, 20H (space) is stored between the sign and the numeric value. If the number of digits of the 32-bit binary data is greater, an error occurs.



  - The value "00H" is automatically stored at the end of the converted string.

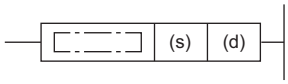
Operation error

Error code (SD0)	Description
3281H	Invalid data that cannot be converted is input to (s1). • The specified total number of digits is out of the range, 2 to 13. • The specified number of digits in the decimal part of (s1)+1 is out of the range, 0 to 10. • When the specified number of digits in the decimal part of (s1)+1 is in the range 0 to 10, the relationship between the total number of digits specified by (s1) and the number of digits in the decimal part specified by (s1)+1 does not satisfy the following. (Total number of digits)-3≥Number of digits in the decimal part • The number of digits specified by (s1) is smaller than the number of digits plus 2 of the 32-bit binary data specified by (s2). [Number of digits in (s1)] < [Number of digits of 32-bit binary data excluding the sign in (s2) + Number of digits in the sign (+ or -) + Number of digits of decimal point (.)]
3286H	The entire string that was converted in the assignment range of the relevant device/label for the device number specified by (d) and later cannot be stored.

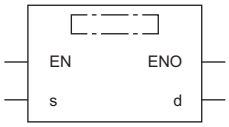
Converting BCD 4-digit data to decimal ASCII code

BCDDA(P)

These instructions convert BCD 4-digit data to the ASCII code.

Ladder	ST
	ENO:=BCDDA(EN,s,d); ENO:=BCDDAP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BCDDA	
BCDDAP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	BCD data used for ASCII conversion	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the conversion result	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the numerical value of each digit of the BCD 4-digit data in the device specified by (s) to the ASCII code, and store the converted data in the device number specified by (d) and later.
- The format of the hexadecimal ASCII data to be stored in (d) depends on the status of SM705 (Number of conversion digits selection).

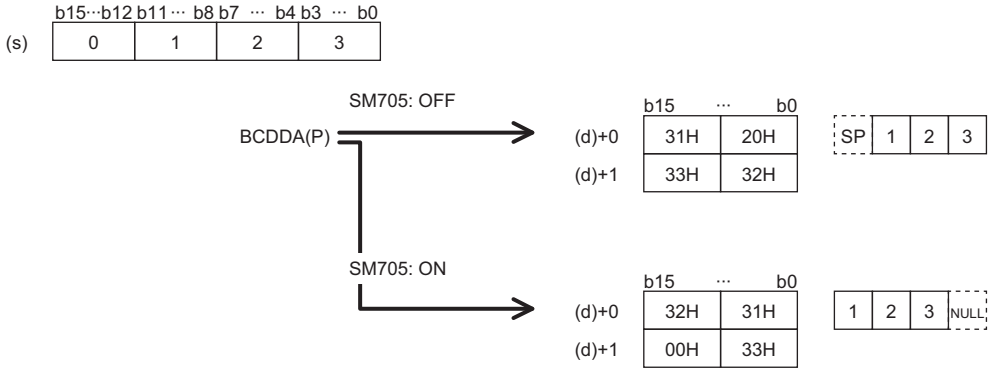
Status of SM705	Storage format of (d)	Reference
Off	Data is stored in a fixed number of digits (4 digits)	Page 774 Operation of when SM705 (Number of conversion digits selection) is off
On	Each digit is stored left-justified depending on the value of (s).	Page 774 Operation of when SM705 (Number of conversion digits selection) is on

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

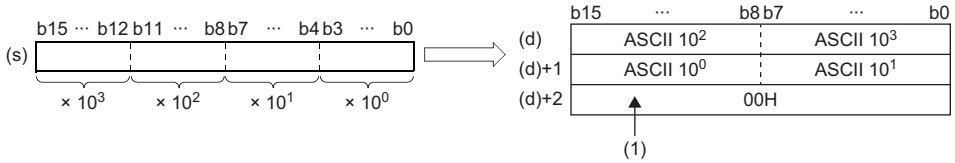
When the BCDDA(P) instruction is executed with BCD 4-digit data "0123" stored in (s)



- When SM705 is off, the number of digits is fixed. The leading "0" of "0123" is converted into 20H (space) and stored.
- When SM705 is on, data is left-justified. "0123" with the leading "0" omitted ("123") is converted into ASCII data and stored, and 00H is stored in the end.

■Operation of when SM705 (Number of conversion digits selection) is off

Decimal ASCII data is stored in a fixed number of digits (4 digits) in (d) to (d)+1.



ASCII 10^3: Thousands place of ASCII code

ASCII 10^2: Hundreds place of ASCII code

ASCII 10^1: Tens place of ASCII code

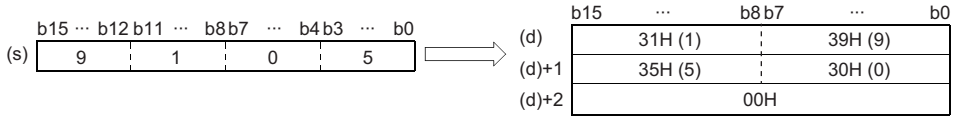
ASCII 10^0: Ones place of ASCII code

(1): 00H is stored in (d)+2 when SM701 (Number of output characters selection) is off. When it is on, the value remains unchanged.

- 20H (space) is stored for the leading "0"s at the left of the effective number of digits of the operation result stored in the device specified by (d). (Zero-suppression) For "0050", "00" becomes 20H (space) and "50" is the effective number of digits.

Ex.

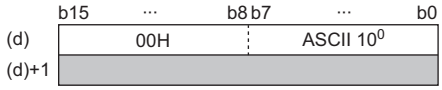
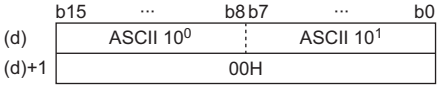
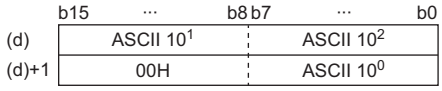
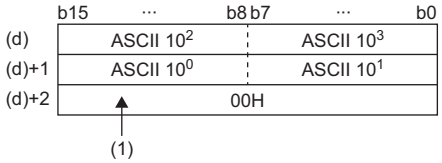
When 9105 is specified in (s)



■Operation of when SM705 (Number of conversion digits selection) is on

Decimal ASCII data for the number of digits (up to 4 digits) without the leftmost "0" in the effective digits is stored in (d).

The following figures show an example of a value of (s) and a value stored in (d).

Value of (s)	Data of (d) to (d)+1	Value of (s)	Data of (d) to (d)+1
• 0H to 9H	• The upper byte of (d) is filled with 00H. • (d)+1 and later remain unchanged. 	• 10H to 99H	• (d)+1 is filled with 00H. 
• 100H to 999H	• The upper byte of (d)+1 are filled with 00H. 	• 1000H to 9999H	• (1): (d)+2 is filled with 00H when SM701 (Number of output characters selection) is off. (d)+2 remains unchanged if SM701 is on. 

ASCII 10^3: Thousands place of ASCII code

ASCII 10^2: Hundreds place of ASCII code

ASCII 10^1: Tens place of ASCII code

ASCII 10^0: Ones place of ASCII code

- When the number of digits is less than the maximum number of digits (4 digits), 00H is stored in the end of the string regardless of the status (on/off) of SM701 (Number of output characters selection). If the end of the string is the lower byte, 00H is also stored in the upper byte.
- When the number of digits is equal to the maximum number of digits (4 digits), 00H is stored in (d)+2 when SM701 is off. (d)+2 remains unchanged if SM701 is on.

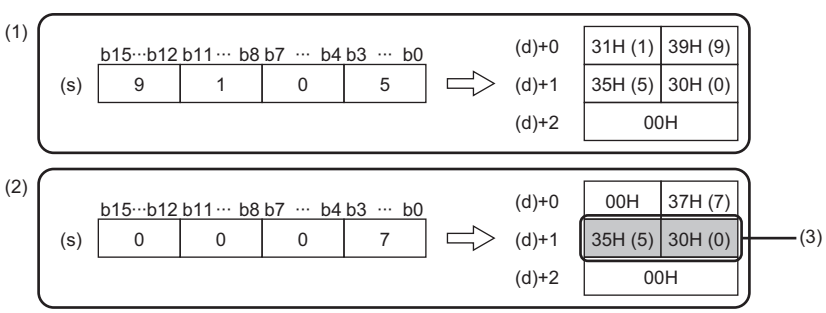
Operation error

Error code (SD0)	Description
3281H	Data in the device specified by (s) is out of the range, 0 to 9999.
3286H	The entire string that was converted in the assignment range of the relevant device/label for the device number specified by (d) and later cannot be stored.

Precautions

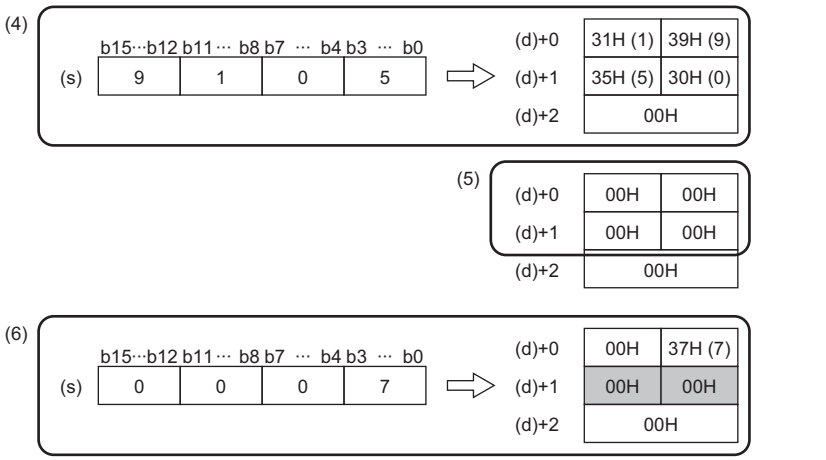
When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the BCDDA(P) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation result may not be overwritten by the succeeding result and can remain in (d).

[Example] Executing the BCDDA(P) instruction when (s) is "9105H" and then executing another BCDDA(P) instruction when (s) is "0007H"



(1) "9105" is converted into a string.
(2) "7" is converted into a string.
(3) A part of the previous conversion result remains in (d)+1.

To avoid this, create a program to clear the entire data storage areas (d)+0 and (d)+1 before executing the BCDDA(P) instruction.

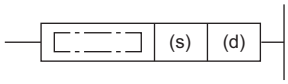


(4) "9105" is converted into a string.
(5) (d)+0 to (d)+1 are cleared.
(6) "7" is converted into a string.

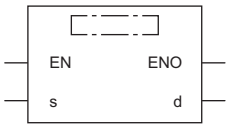
Converting BCD 8-digit data to decimal ASCII code

DBCDDA(P)

These instructions convert BCD 8-digit data to the ASCII code.

Ladder	ST
	ENO:=DBCDDA(EN,s,d); ENO:=DBCDDAP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBCDDA	
DBCDDAP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	BCD data used for ASCII conversion	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the conversion result	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the numerical value of each digit of the BCD 8-digit data in the device specified by (s) to the ASCII code, and store the converted data in the device number specified by (d) and later.
- The format of the decimal ASCII data to be stored in (d) depends on the status of SM705 (Number of conversion digits selection).

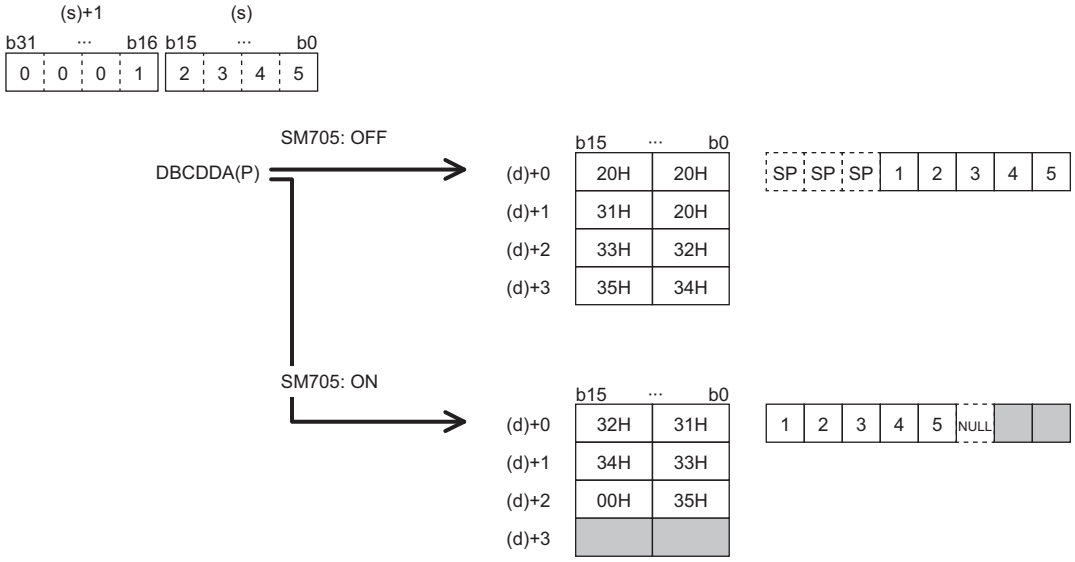
Status of SM705	Storage format of (d)	Reference
Off	Data is stored in a fixed number of digits (8 digits)	Page 778 Operation of when SM705 (Number of conversion digits selection) is off
On	Each digit is stored left-justified depending on the value of (s).	Page 779 Operation of when SM705 (Number of conversion digits selection) is on

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

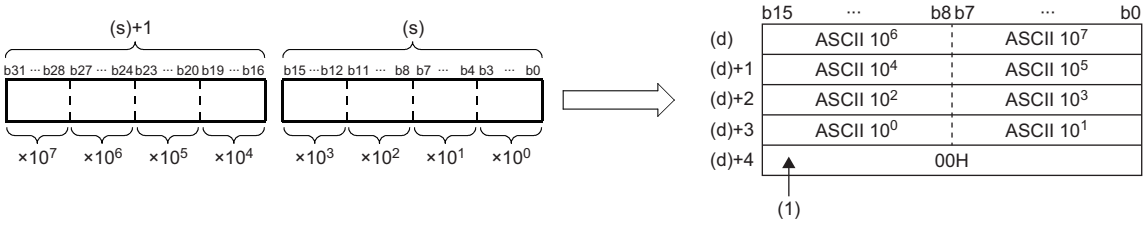
When the DBCDDA(P) instruction is executed with BCD 8-digit data "00012345" stored in (s)



- When SM705 is off, the number of digits is fixed. Each of the leading "0"s of "00012345" is converted into 20H (space) and stored.
- When SM705 is on, data is left-justified. "00012345" with the leading "0"s omitted ("12345") is converted into ASCII data and stored, and 00H is stored in the end.

■Operation of when SM705 (Number of conversion digits selection) is off

Decimal ASCII data is stored in a fixed number of digits (8 digits) in (d) to (d)+3.



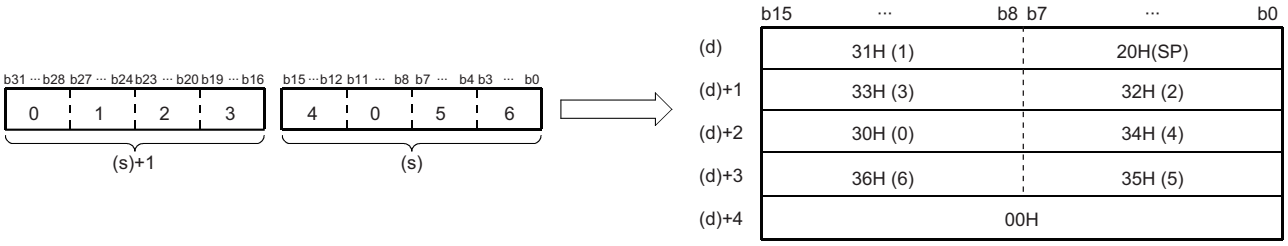
ASCII 10^7: Ten-millions place of ASCII code
ASCII 10^6: Millions place of ASCII code
⋮
ASCII 10^1: Tens place of ASCII code
ASCII 10^0: Ones place of ASCII code

(1): 00H is stored in (d)+4 when SM701 (Number of output characters selection) is off. When it is on, the value remains unchanged.

- 20H (space) is stored for the leading "0"s at the left of the effective number of digits of the operation result stored in the device specified by (d). (Zero-suppression) For "00012098", "000" becomes 20H (space) and "12098" is the effective number of digits.

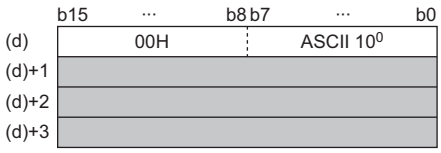
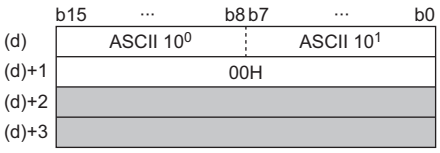
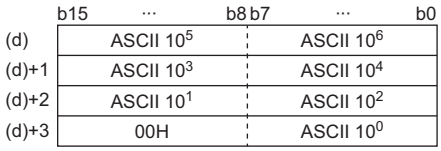
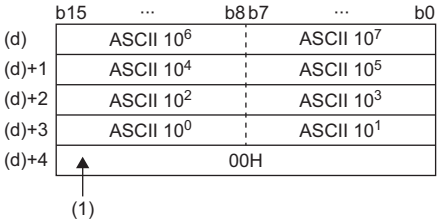
Ex.

When 01234056 is specified in (s)



■Operation of when SM705 (Number of conversion digits selection) is on

Decimal ASCII data for the number of digits (up to 8 digits) without the leftmost "0" in the effective digits is stored in (d).
The following figures show an example of a value of (s) and a value stored in (d).

<table>
<tr><th>Value of (s)</th><th>Data of (d) to (d)+3</th><th>Value of (s)</th><th>Data of (d) to (d)+3</th></tr>
<tr><td>• 0H to 9H</td><td>• The upper byte of (d) is filled with 00H.
• (d)+1 and later remain unchanged.

</td><td>• 10H to 99H</td><td>• (d)+1 is filled with 00H.
• (d)+2 and later remain unchanged.

</td></tr>
<tr><td colspan="4">⋮</td></tr>
<tr><td>• 1000000H to 9999999H</td><td>• The upper byte of (d)+3 is filled with 00H.

</td><td>• 10000000H to 99999999H</td><td>• (1): (d)+4 is filled with 00H when SM701 (Number of output characters selection) is off. (d)+4 does not change if SM701 (Number of output characters selection) is on.

</td></tr>
</table>

ASCII 10^7: Ten-millions place of ASCII code
ASCII 10^6: Millions place of ASCII code
⋮
ASCII 10^1: Tens place of ASCII code
ASCII 10^0: Ones place of ASCII code

- When the number of digits is less than the maximum number of digits (8 digits), 00H is stored in the end of the string regardless of the status (on/off) of SM701 (Number of output characters selection). If the end of the string is the lower byte, 00H is also stored in the upper byte.
- When the number of digits is equal to the maximum number of digits (8 digits), 00H is stored in (d)+4 when SM701 (Number of output characters selection) is off. (d)+4 does not change if SM701 (Number of output characters selection) is on.

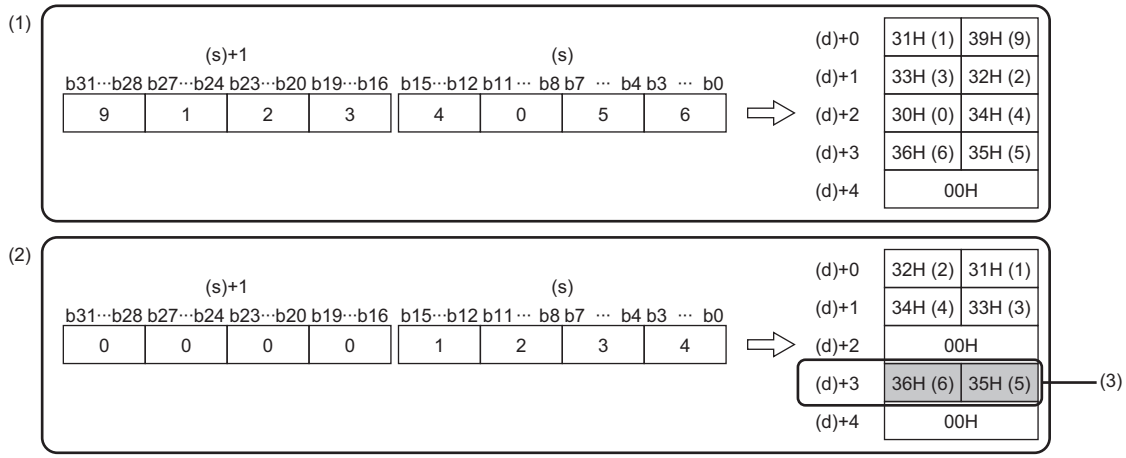
Operation error

Error code (SD0)	Description
3281H	Data in the device specified by (s) is out of the range, 0 to 99999999.
3286H	The entire string that was converted in the assignment range of the relevant device/label for the device number specified by (d) and later cannot be stored.

Precautions

When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the DBCDDA(P) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation result may not be overwritten by the succeeding result and can remain in (d).

[Example] Executing the DBCDDA(P) instruction when (s) is "91234056H" and then executing another DBCDDA(P) instruction when (s) is "00001234H"

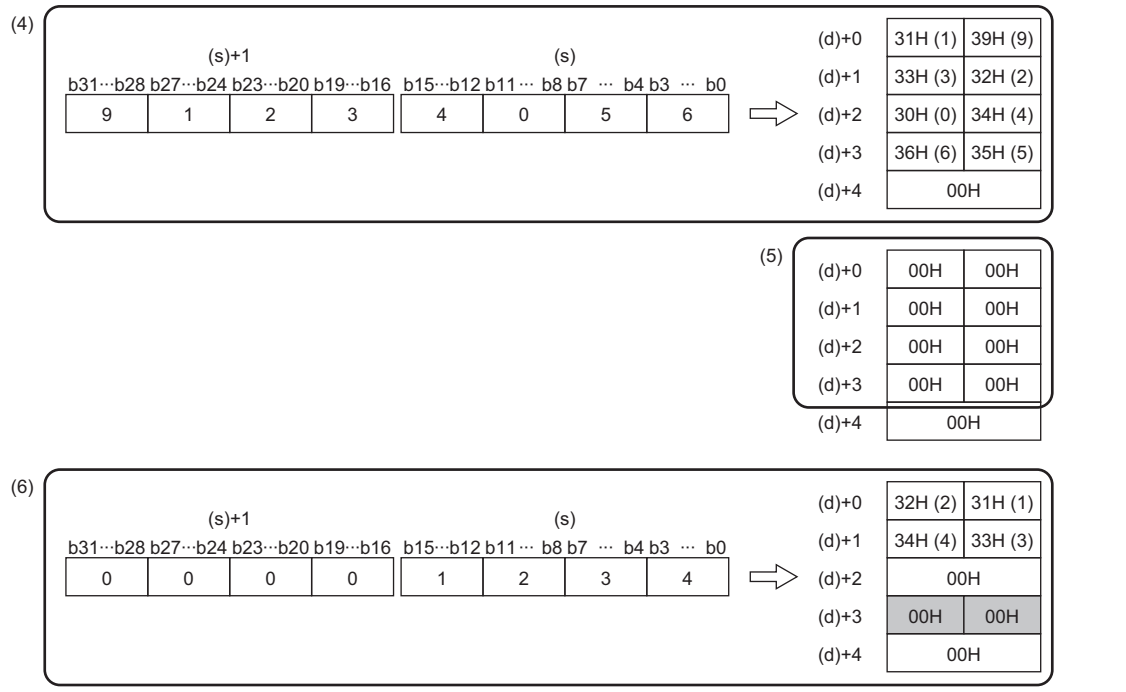


(1) "91234056" is converted into a string.

(2) "1234" is converted into a string.

(3) A part of the previous conversion result remains in (d)+3.

To avoid this, create a program to clear the entire data storage areas (d)+0 to (d)+3 before executing the DBCDDA(P) instruction.



(4) "91234056" is converted into a string.

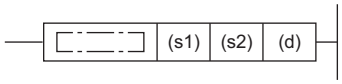
(5) (d)+0 to (d)+3 are cleared.

(6) "1234" is converted into a string.

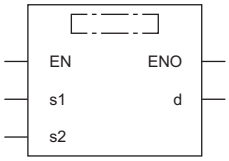
Converting single-precision real number to string data

ESTR(P)

These instructions convert single-precision real number data to a string according to the display specification.

Ladder	ST
	ENO:=ESTR(EN,s1,s2,d); ENO:=ESTRP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ESTR	
ESTRP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Single-precision real number data to be converted, or the start device containing the data	0, 2^{-126}<|(s1)|<2^{128}	Single-precision real number	ANYREAL_32
(s2)	Start device containing the display specification of the real number to be converted	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(d)	Start device for storing the converted string	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

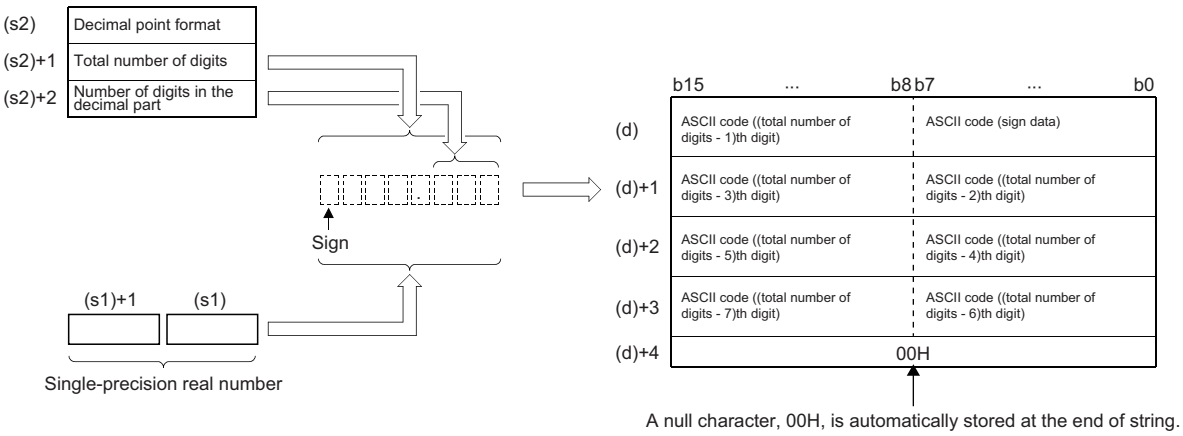
Processing details

- These instructions convert the single-precision real number data stored in the device specified by (s1) to a string according to the display specification stored in the device number specified by (s2) and later, and store the converted data in the device number specified by (d) and later.
- The type of the converted data varies depending on the display specification stored in the device specified by (s2).

(s2)	0: Decimal point format 1: Exponent format
(s2)+1	Total number of digits
(s2)+2	Number of digits in the decimal part

■Decimal point format

- When 0 is specified in (s2), the decimal point format is used.



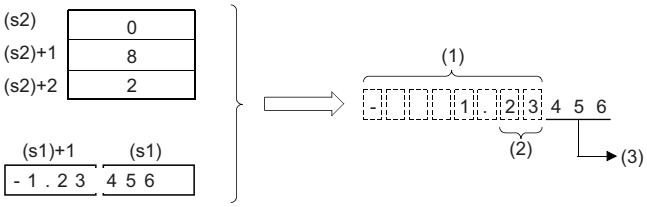
- When storing the decimal part in (d), the total number of digits that can be specified by (s2) + 1 is the total number of digits (maximum of 24) ≥ (number of digits in the integral part[*1] + number of digits in the decimal part + 2). The number of digits in the decimal part that can be specified by (s2)+2 is 0 to 7. Note that the number of digits in the decimal part must be equal to or smaller than the total number of digits minus 3.

*1 Indicates the number of digits in the integral part of the 32-bit floating point real number data in the device specified by (s1).

- When not storing the decimal part in (d), the total number of digits that can be specified by (s2) + 1 is the total number of digits (maximum of 24) ≥ (number of digits in the integral part[*2] + 1). The number of digits in the decimal part that can be specified by (s2)+2 cannot be set.

*2 Indicates the number of digits in the integral part of the 32-bit floating point real number data in the device specified by (s1).

- The converted string data are stored in the device number specified by (d) and later as shown below.
  - As sign data, 20H (space) is stored if the single-precision real number data is positive, and 2DH (-) is stored if the data is negative.
  - If the decimal part of the single-precision real number data is not stored within the range of the number of digits in the decimal part, the lower decimal digits are rounded off.

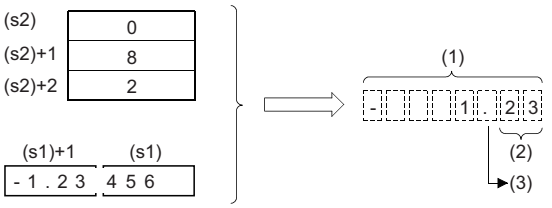


(1) Total number of digits

(2) Number of digits in the decimal part

(3) Rounded off.

  - If the number of digits in the decimal part is set to a value other than 0, 2EH (.) is automatically stored at the position of the specified number of digits in the decimal part plus 1. If the number of digits in the decimal part is 0, "2EH" (.) is not stored.

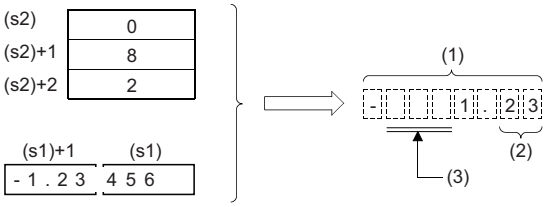


(1) Total number of digits

(2) Number of digits in the decimal part

(3) Automatically added.

  - If the number of digits excluding the sign, decimal point, and decimal part from the total number of digits is greater than the number of digits in the integral part of single-precision real number data, 20H (space) is stored between the sign and integral part.



(1) Total number of digits

(2) Number of digits in the decimal part

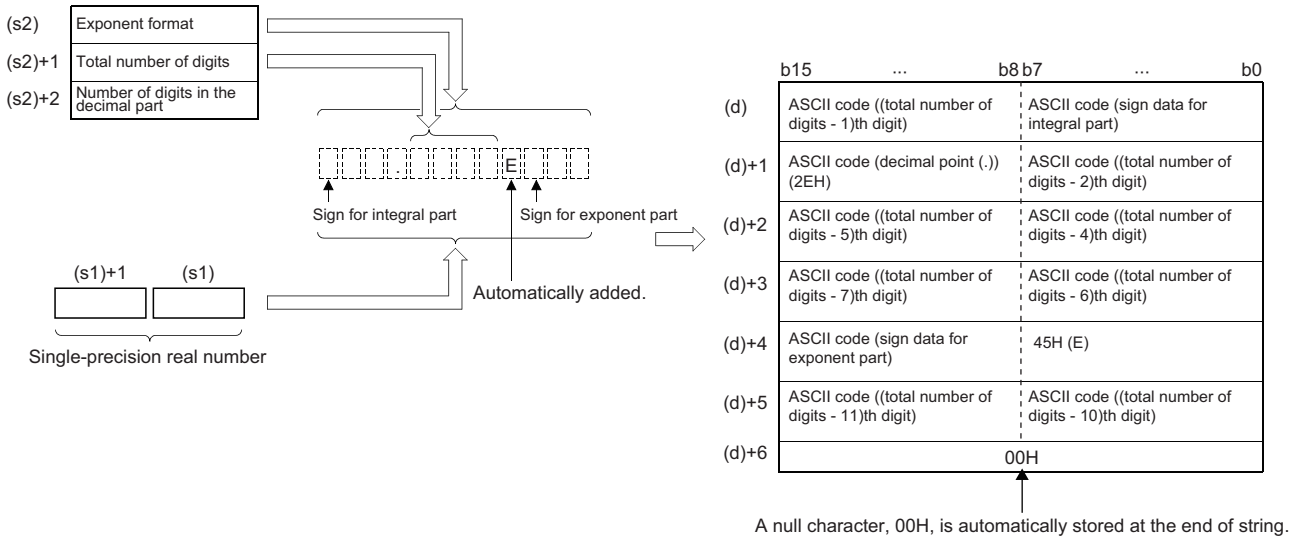
(3) Filled with 20Hs (spaces).

  - The value "00H" is automatically stored at the end of the converted string.

- The number of digits in the integral part of the 32-bit floating point real number data in the device specified by (s1) can be 1 to 16.

■Exponential format

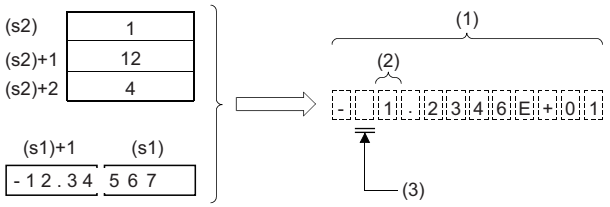
• When 1 is specified in (s2), the exponential format is used.



• When storing the decimal part in (d), the total number of digits that can be specified by (s2) + 1 is the number of digits (maximum of 24) ≥ (number of digits in the decimal part+7). The number of digits in the decimal part that can be specified by (s2)+2 is 0 to 7. Note that the number of digits in the decimal part must be equal to or smaller than the total number of digits minus 7.

• When not storing the decimal part in (d), the total number of digits that can specified by (s2) + 1 is the number of digits (maximum of 24) ≥ 6. The number of digits in the decimal part that can be specified by (s2)+2 cannot be set.

- The converted string data are stored in the device number specified by (d) and later as shown below.
  - As sign data in the integral part, 20H (space) is stored if the single-precision real number data is positive, and 2DH (-) is stored if the data is negative.
  - The integral part is fixed to one digit. 20H (space) is stored between the integral part and sign.

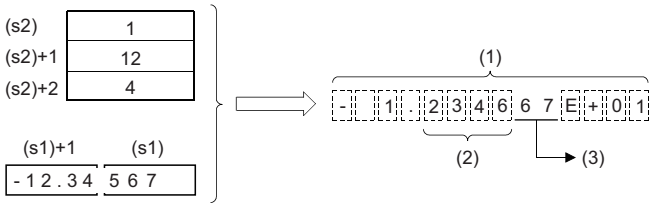


(1) Total number of digits

(2) Fixed to 1 digit

(3) Filled with 20Hs (spaces).

- If the decimal part of the single-precision real number data is not stored within the range of the number of digits in the decimal part, the lower decimal digits are rounded off.

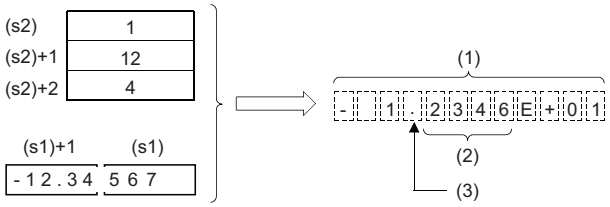


(1) Total number of digits

(2) Number of digits in the decimal part

(3) Rounded down.

- If the number of digits in the decimal part is set to a value other than 0, 2EH (.) is automatically stored at the position of the specified number of digits in the decimal part plus 1. If the number of digits in the decimal part is 0, "2EH" (.) is not stored.

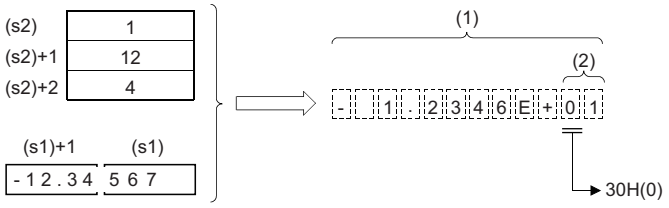


(1) Total number of digits

(2) Number of digits in the decimal part

(3) Automatically added.

- As sign data in the exponent part, 2BH (+) is stored if the exponent is positive, and 2DH (-) is stored if it is negative.
- The exponent part is fixed to two digits. When the exponent part is one digit, 30H(0) is stored between the exponent part and sign.



(1) Total number of digits

(2) Fixed to 2 digits

- The value "00H" is automatically stored at the end of the converted string.

- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Operation error

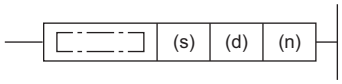
Error code (SD0)	Description
3281H	The value in the device specified by (s1) is not in the following range. 0, $2^{-126} \leq |(s1)| < 2^{128}$
	Invalid data that cannot be converted is set to (s2). • The format specification in the device specified by (s2) is a value other than 0 and 1. • In the decimal point format, the total number of digits specified by (s2)+1 is outside the following range. The number of digits in the decimal part is 0: Total number of digits≥number of digits in the integral part*1+1 The number of digits in the decimal part is not 0: Total number of digits≥number of digits in the integral part*1+ number of digits in the decimal point+2 • In the decimal point format, the total number of digits specified by (s2)+1 and the number of digits in the decimal part specified by (s2)+2 are outside the following range. The number of digits in the decimal part is 0: Total number of digits≥2 The number of digits in the decimal part is not 0: Total number of digits≥(number of digits in the decimal point+3) • In the exponential format, the total number of digits specified by (s2)+1 is outside the following range. The number of digits in the decimal part is 0: Total number of digits≥6 The number of digits in the decimal part is not 0: Total number of digits≥(number of digits in the decimal part+7) • In the exponential point format, the decimal part digit specification in the device specified by (s2)+2 is outside the following range. Decimal point format: Number of digits in the decimal part≤(total number of digits - 3) Exponential format: Number of digits in the decimal part≤(Total number of digits - 7) • In the decimal point format, the number of digits in the integral part of 32-bit floating point real number data in the device specified by (s1) exceeds 16. • The number of digits in the decimal part specified by (s2)+2 is out of the range from 0 to 7.
	The specified value consists of more than 24 digits in total.
3282H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
3286H	The entire string that was converted in the assignment range of the relevant device/label for the device number specified by (d) and later cannot be stored.

*1 Indicates the number of digits in the integral part of the 32-bit floating point real number data in the device specified by (s1).

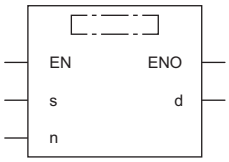
Converting hexadecimal binary data to hexadecimal ASCII code

INT2ASC(P)

These instructions convert 16-bit binary data to the hexadecimal ASCII code and store it in any specified range.

Ladder	ST
	ENO:=INT2ASC(EN,s,n,d); ENO:=INT2ASCP(EN,s,n,d)

FBD/LD



■Execution condition

Instruction	Execution condition
INT2ASC	
INT2ASCP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the binary data to be converted to a string	—	16-bit signed binary	ANY16*1
(d)	Start device for storing the converted string	—	String	ANYSTRING_SINGLE
(n)	Number of characters to be stored	0 to 16383	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

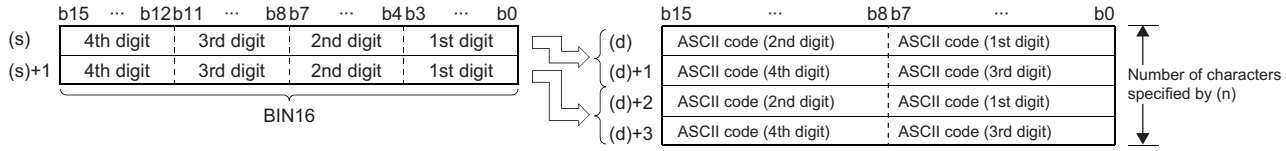
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

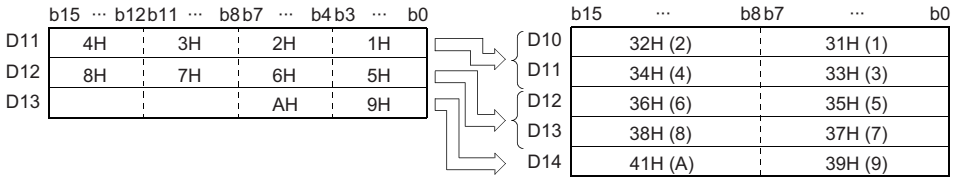
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

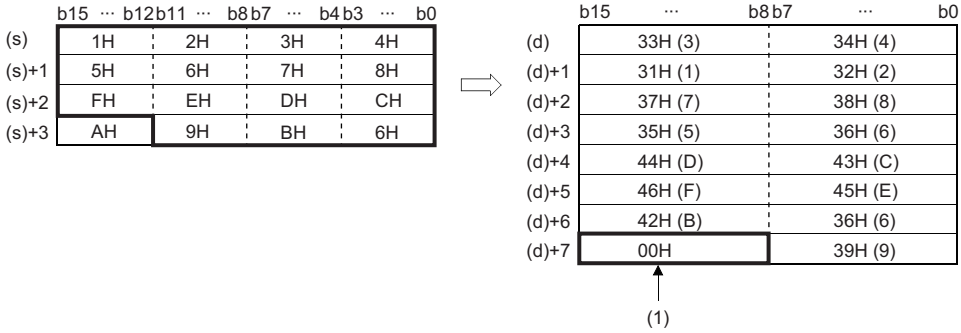
- The 16-bit binary data in the device number specified by (s) and later are converted to hexadecimal ASCII, and the converted data by the number of characters in the device specified by (n) are stored in the device number specified by (d) and later.



- Setting the number of characters by (n) automatically determines the range of binary data in the device specified by (s) and the range of the device specified by (d) for storing the string data.
- Processing is performed normally even if the device range in which the binary data to be converted and the device range for storing the converted binary data are overlapping.



- If the number of characters in the device specified by (n) is an odd number, 00H is automatically stored in the upper 8 bits of the last device number among device numbers for storing the converted string data.



(1) 00H is automatically stored.

- If the number of characters in the device specified by (n) is 0, no processing is performed.

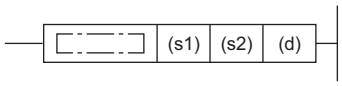
Operation error

Error code (SD0)	Description
3285H	An out-of-range value is set to (n). • The specified number of characters is not between 0 and 16383.

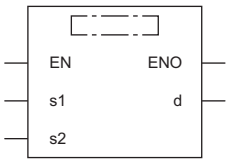
Converting and concatenating multiple data strings

SPF(P)

Multiple data is converted to a string according to the specified display format, the string is concatenated with any string, and they are output as a single string.

Ladder	ST
	ENO:=SPF(EN,s1,s2,d); ENO:=SPFP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SPF	
SPFP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	String data and display format, or the start device containing them	—	String	ANYSTRING_SINGLE
(s2)	Start device where the conversion target data is stored	—	16-bit signed binary	ANY16
(d)	Start device for storing the converted and concatenated string	—	String	ANYSTRING_SINGLE

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- This instruction converts data stored in the device number specified by (s2) and later into strings according to the display format specified by (s1). Then, the strings are concatenated with a given string stored in (s1), and they are stored as a single string in the device number specified by (d) and later. The given string can be omitted, allowing only the data stored in the device number specified by (s2) and later to be concatenated.
- The display format is as follows: Specifiers are combined in the order starting from "%", followed by [Flag], [Minimum field width], [Accuracy], [Acquisition data size], and [Conversion specifier]. The specifiers are stored together with the concatenation target string in (s1). The following table lists the display format selection items that can be used. If the order of the display format items is incorrect, an error occurs.

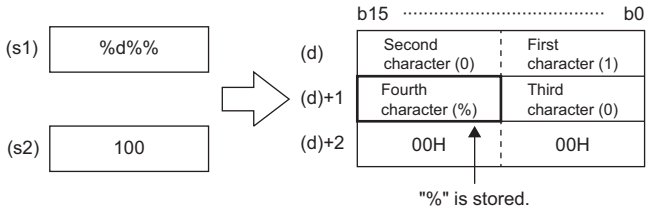
Display format	Specifier*1	Description	Details
Flag	-, +, " " (space), #, and 0	Modifies the string to be output to (d) according to the specified specifier.	☞ Page 792 Flag
Minimum field width	Any numerical value	Specifies the total number of characters to be output to (d), including the sign of the converted string, decimal point, exponent part, and " " (space). When this item is omitted, the number of characters for the retrieved data is specified for the output.	☞ Page 793 Minimum field width
Accuracy	"." (period) numerical value	Specifies the number of characters to be output to (d) with [Conversion specifier] from the data stored in (s2).	☞ Page 794 Accuracy
Acquisition data size	w, q	Specifies how many words of the data stored in (s2) are to be converted and output to (d). When this item is omitted, the data is retrieved in units of one word.	☞ Page 796 Acquisition data size
Conversion specifier	d, u, x, X, o, s, c, f, e, E, g, G	Specifies the output format of the string to be output to (d).	☞ Page 797 Conversion specifier

*1 Specifiers are case-sensitive.

- [Flag], [Minimum field width], [Accuracy], and [Acquisition data size] can be omitted.
- Only if "%" is output as a character, [Conversion specifier] can be omitted. When "%%" is stored in (s1) and output as a string, "%" will be stored in (d) as a string. The data stored following "%%" is output as a character up to the next display format.

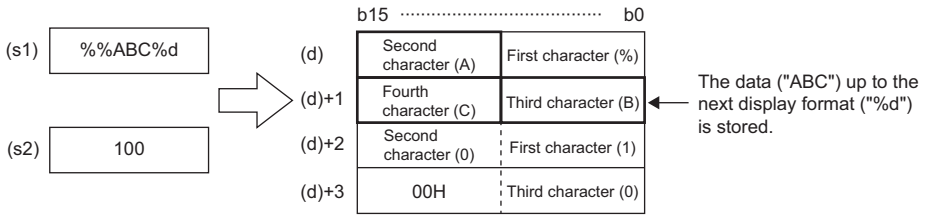
Ex.

When "%d%%" is stored in (s1) and "100" is stored in the device specified by (s2), "100%" is stored in the device specified by (d).



Ex.

When "%%ABC%d" is stored in (s1) and "100" is stored in the device specified by (s2), "%ABC100" is stored in the device specified by (d).

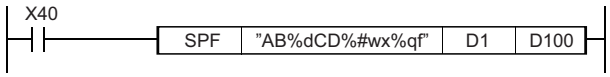


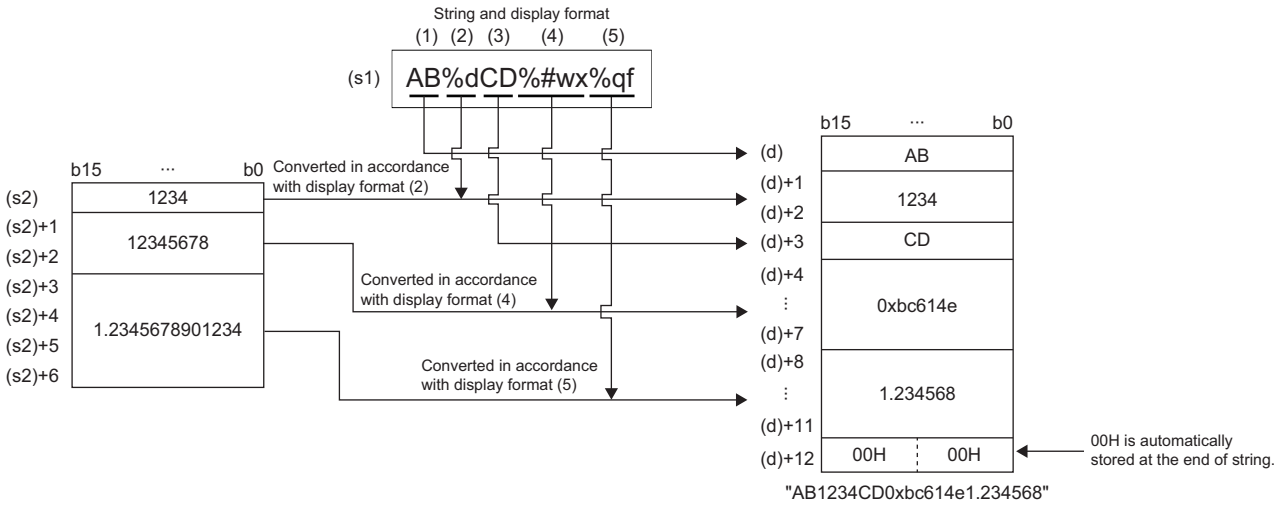
- If the converted string (d) ends with an even number of bytes, the NULL code "00H" is stored in the device (upper 8 bits and lower 8 bits) following the last device of the string. If the converted string (d) ends with an odd number of bytes, the NULL code "00H" is stored in the last device (upper 8 bits) of the string.

Ex.

When the SPF(P) instruction is executed while the data listed below is stored, the string stored in D100 and later is "AB1234CD0xbc614e1.234568".

Operand	Stored data	
(s1)	AB%dCD%#wx%qf (string) • The following display format is specified for the stored string. AB (outputting the string) %d (D1 display format) CD (outputting the string) %#wx (D2 and D3 display format) %qf (D4 to D7 display format)	
(s2)	D1	1234 (decimal)
	D2 to D3	12345678 (decimal)
	D4 to D7	1.2345678901234 (double-precision real number)
(d)	D100	





(1): The string "AB" is output.

(2): The display format: [Acquisition data size] = (Omitted), [Conversion specifier] = d Data is obtained from (s2) in units of one word, and output as a signed decimal number.

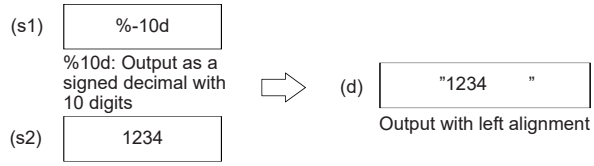
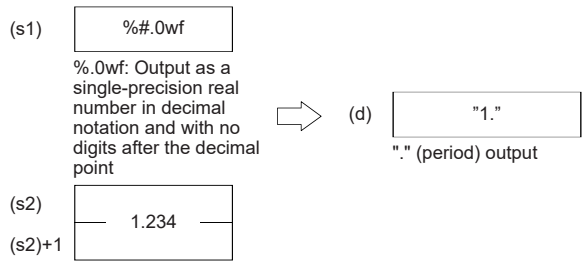
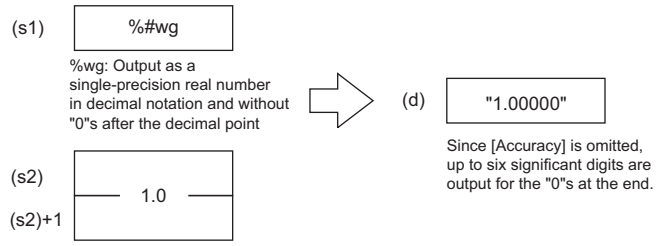
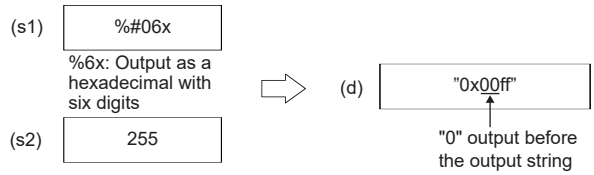
(3): The string "CD" is output.

(4): The display format: [Flag] = #, [Acquisition data size] = w, [Conversion specifier] = x Data is obtained from (s2)+1 in units of two words, prefixed with "0x", and output as an unsigned hexadecimal number.

(5): The display format: [Accuracy] = (Omitted), [Acquisition data size] = q, [Conversion specifier] = f Data is obtained from (s2)+3 in units of four words, and output as a double-precision real number in decimal notation. Since the description of [Accuracy] is omitted, a value with six digits after the decimal point is output, and the last digit of the output string is rounded off.

■Flag

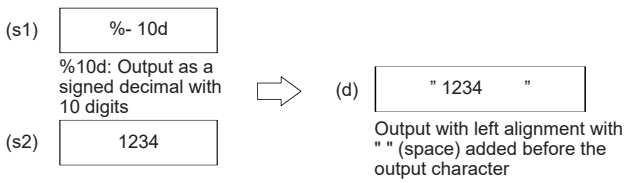
This item modifies the string to be output to (d) according to conditions.

Specifier	Description
(Omitted)	Outputs a string with right alignment.
-	When [Minimum field width] is larger than the number of converted characters obtained from (s2), the string is output with left alignment. 
+	To output 0 or a positive number, add "+".
" " (space)	To output 0 or a positive number, add " " (space).
#	Outputs a string in the following output formats according to the [Conversion specifier] setting. • "o": Adds "0" to the beginning. • "x" or "X": Adds "0x" or "0X" to the beginning. • "f", "e", or "E": Always outputs a decimal point when a real number is output.  • "g" or "G": Always outputs a decimal point when a real number is output, and appends "0" for the number of significant digits to the end. 
0	When [Minimum field width] is larger than the number of converted characters obtained from (s2), the output string is prefixed with "0" for output. 

Multiple specifiers can be specified for [Flag]. There are no restrictions on the order to be used. However, when "+" and " " (space) are concurrently used, the content of the specifier "+" takes a priority for the output. If the same specifier is used, the second and subsequent specifications are ignored. If the specifiers "-" and "0" are specified simultaneously, the specifier "0" is ignored.

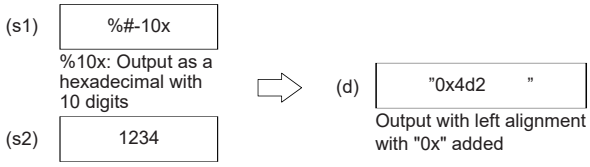
Ex.

When "-" and " " (space) are used



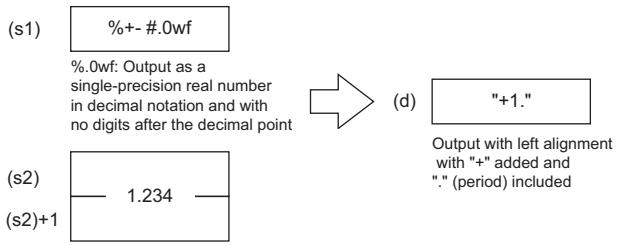
Ex.

When "#" and "-" are used



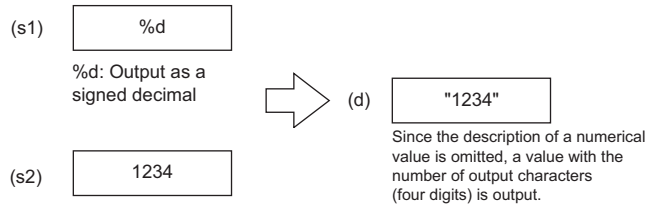
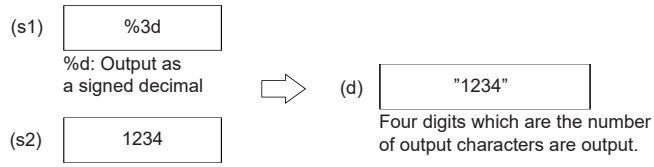
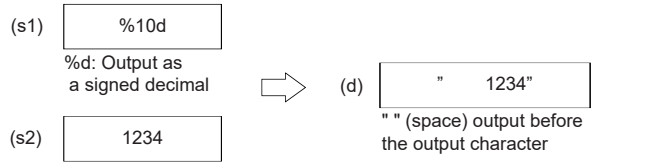
Ex.

When "+", "-", " " (space), and "#" are used



■Minimum field width

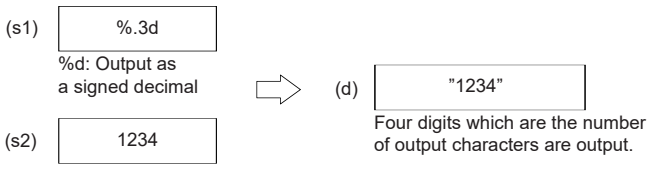
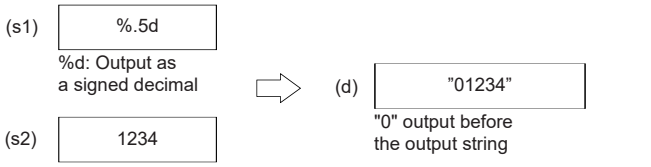
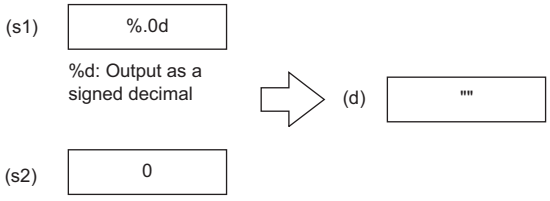
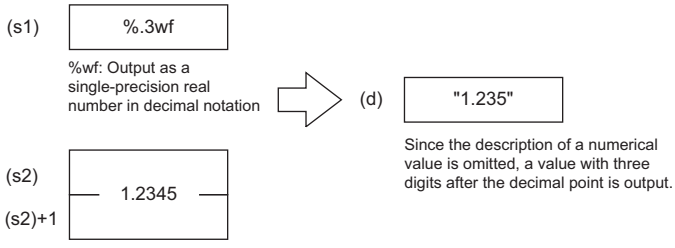
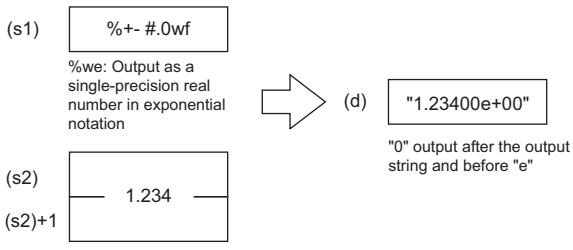
This item specifies the total number of characters to be output to (d), including the sign of the converted string, decimal point, exponent part, and " " (space).

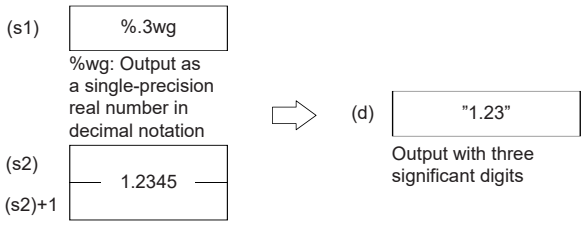
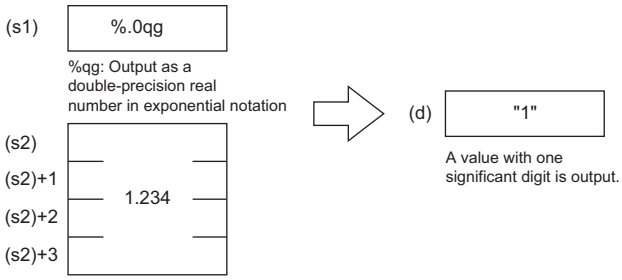
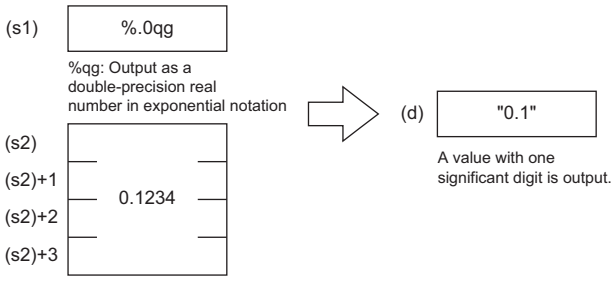
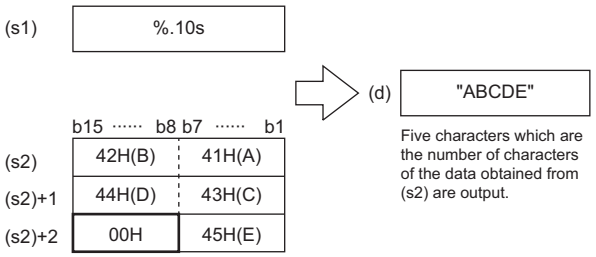
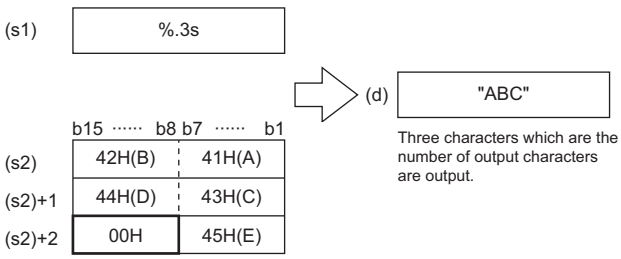
Specifier	Description
Numerical value	• When this item is omitted, the number of characters for the retrieved data is specified for the output.  • When [Minimum field width] is equal to or less than the number of converted characters obtained from (s2), the specified [Minimum field width] is ignored.  • When [Minimum field width] is larger than the number of converted characters obtained from (s2), " " (space) is output before the numerical value. 

■Accuracy

Specifies the number of characters to be output to (d) with [Conversion specifier] from the data stored in (s2).

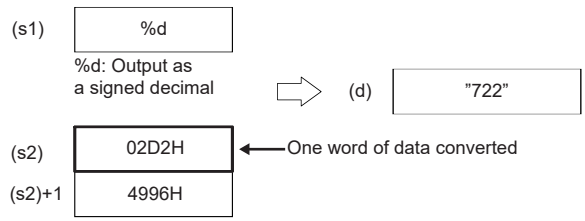
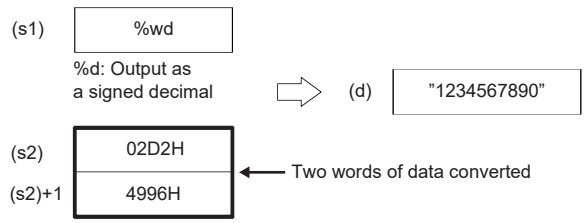
To indicate the accuracy, add the numerical value after "." (period). When only "." (period) is described and the numerical value is omitted, the operation is the same as when ".0" is specified.

Specifier	Description
"." (period) numerical value	■When [Conversion specifier] is "c" The specified [Accuracy] is ignored. ■When [Conversion specifier] is "d", "u", "x", "X", or "o" The number of output characters is specified. • When the specified numerical value is equal to or less than the number of converted characters obtained from (s2), the specified [Accuracy] is ignored.  • When the specified numerical value is larger than the number of converted characters obtained from (s2), the output string is prefixed with "0" for output.  • When [Accuracy] is specified as "0" and the conversion data specified in (s2) is "0", no string is output.  ■When [Conversion specifier] is "f", "e", or "E" The number of output characters for digits after the decimal point is specified. • When the description of a numerical value is omitted, a value with six digits after the decimal point is output, and the last digit of the output string is rounded off.  • When the specified numerical value is larger than the number of converted characters obtained from (s2), "0" is added for output. However, a character other than "0" may be output depending on the conversion data in the device specified by (s2). In addition, 0 will be appended after the output string when decimal notation is used, and appended after the output string and before "e" or "E" when exponential notation is used. 

Specifier	Description
"." (period) numerical value	■When [Conversion specifier] is "g" or "G" The number of significant digits is specified. • When the description of a numerical value is omitted, six significant digits are output, and the last digit of the output string is rounded off.  • When [Accuracy] is specified as ".0", it is equal to when ".1" is specified. When 1.234 is specified in (s2), it is output as shown below.  When 0.1234 is specified in (s2), it is output as shown below.  ■When [Conversion specifier] is "s" The string for the specified number of characters is output in the order of the string data stored in (s2). • When the number of output characters is larger than the number of characters for the string data retrieved from (s2), the string is output for the number of characters for the string data retrieved by (s2).  • When the number of output characters is smaller than the number of characters for the string data retrieved from (s2), the string is output for the number of output characters. 

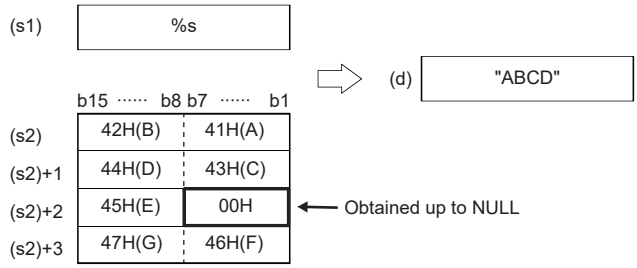
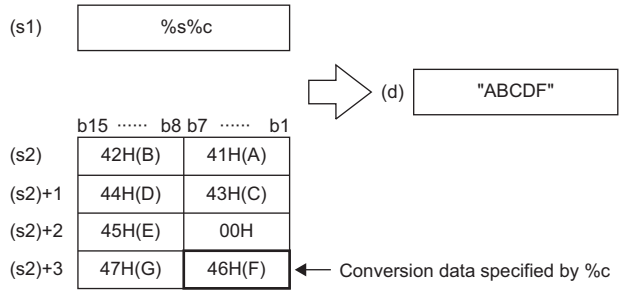
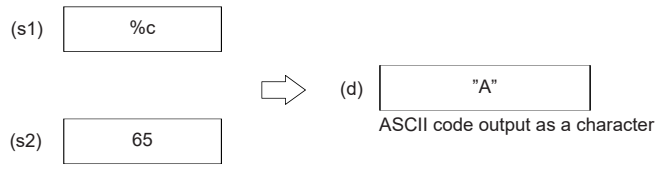
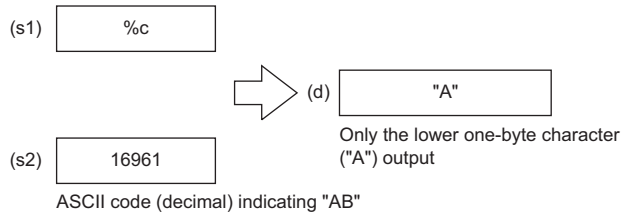
■Acquisition data size

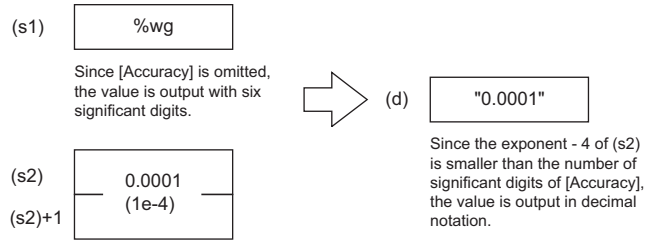
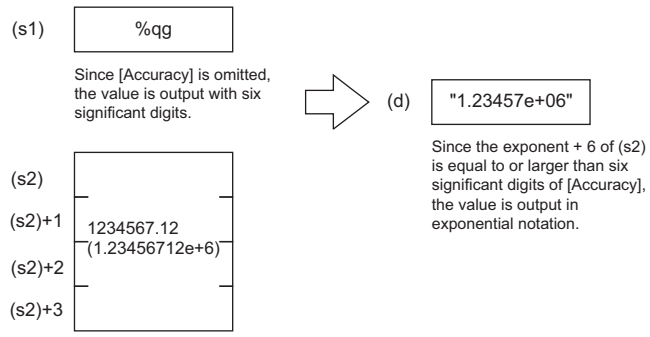
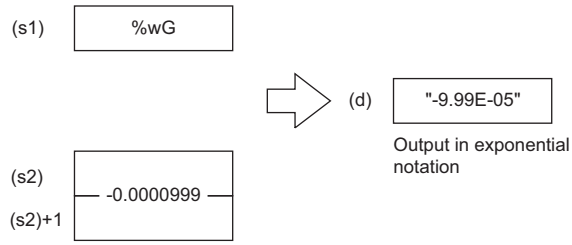
This item specifies how many words of the data stored in (s2) are to be converted and output to (d).

Specifier	Description
(Omitted)	Retrieves (s2) in units of one word. 
w	Retrieves (s2) in units of two words. 
q	Retrieves (s2) in units of four words.

■Conversion specifier

This item specifies the output format of the string to be output to (d).

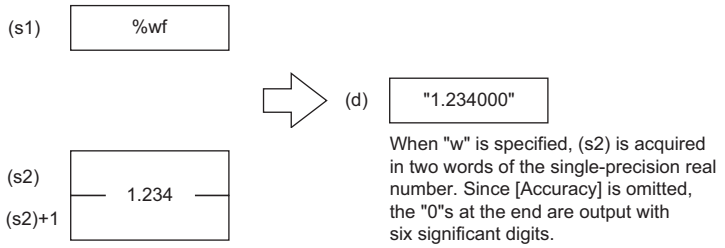
Specifier	Description
d	Outputs a string as a signed decimal number.
u	Outputs a string as an unsigned decimal number.
x	Outputs a string as an unsigned hexadecimal number. Numbers 10 to 15 are displayed as "a" to "f".
X	Outputs a string as an unsigned hexadecimal number. Numbers 10 to 15 are displayed as "A" to "F".
o	Outputs a string as an unsigned octal number.
s	Retrieves the string data stored in (s2) up to 00H (NULL) and outputs it. If 00H (NULL) does not exist, an error (error code: 2820H) occurs.  When 00H (NULL) is stored in the lower 1 byte (b1 to b7), the next conversion data is valid from the data specified in the subsequent device. 
c	Retrieves the string data stored in (s2) as ASCII code (decimal) in units of one word, converts it to a string, and outputs it.  The conversion data stored in (s2) is valid only for the lower one byte (b1 to b7). 
f	Outputs a string in decimal notation. The maximum number of significant digits will be 7 for single-precision real numbers and 15 for double-precision real numbers.
e	Outputs a string in exponential notation, with the exponent indicated by "e". The maximum number of significant digits will be 7 for single-precision real numbers and 15 for double-precision real numbers.
E	Outputs a string in exponential notation, with the exponent indicated by "E". The maximum number of significant digits will be 7 for single-precision real numbers and 15 for double-precision real numbers.

Specifier	Description
g	Outputs a value in either decimal notation or exponential notation, depending on the magnitude of the numerical value. The exponent is indicated by "e". However, if the value in the exponent part of the conversion data specified in (s2) is equal to or larger than the numerical value specified in [Accuracy], the value is output in exponential notation. • Absolute value of numerical value 0.0001 or more: Decimal notation • Numerical value less than 0.0001: Exponential notation The maximum number of significant digits will be 7 for single-precision real numbers and 15 for double-precision real numbers. ■When the conversion data specified in (s2) is 0.0001 or more and the value is output in decimal notation  ■When the conversion data specified in (s2) is 0.0001 or more and the value is output in exponential notation 
G	Outputs a value in either decimal notation or exponential notation, depending on the magnitude of the numerical value. The exponent is indicated by "E". However, if the value in the exponent part of the conversion data specified in (s2) is equal to or larger than the numerical value specified in [Accuracy], the value is output in exponential notation. • Absolute value of numerical value 0.0001 or more: Decimal notation • Numerical value less than 0.0001: Exponential notation The maximum number of significant digits will be 7 for single-precision real numbers and 15 for double-precision real numbers. 

When "f", "e", "E", "g", or "G" is used in [Conversion specifier], the output format can be selected from single-precision real number or double-precision real number by the specifier of [Acquisition data size]. When [Acquisition data size] is specified as "w", a single-precision real number is output, and when [Acquisition data size] is specified as "q", a double-precision real number is output.

Ex.

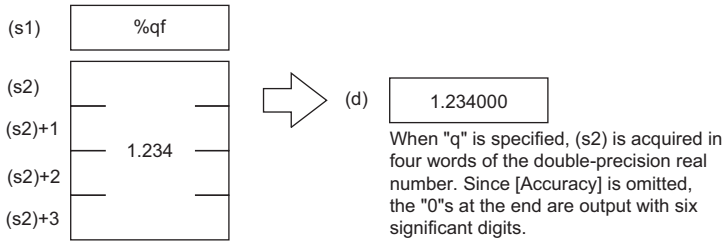
To output as a single-precision real number



When "w" is specified, (s2) is acquired in two words of the single-precision real number. Since [Accuracy] is omitted, the "0"s at the end are output with six significant digits.

Ex.

To output as a double-precision real number



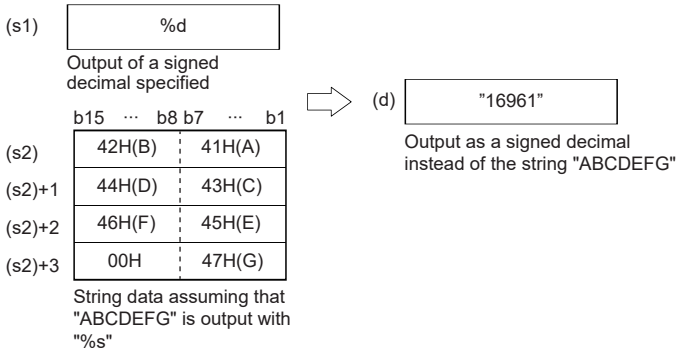
When "q" is specified, (s2) is acquired in four words of the double-precision real number. Since [Accuracy] is omitted, the "0"s at the end are output with six significant digits.

Precautions

The combination of display format items and the format of the data to be converted must always match. Otherwise, an unintended string will be output.

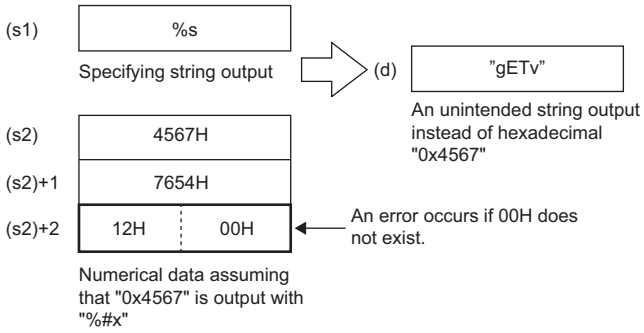
Ex.

When the conversion target data is string data but the display format is numerical data



Ex.

When the conversion target data is numerical data but the display format is string data

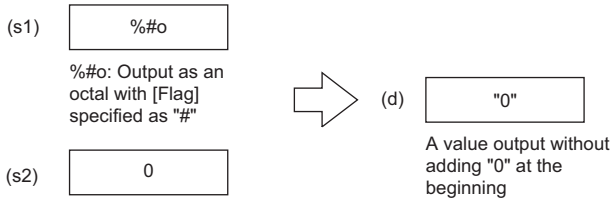


■Flag

- When [Conversion specifier] is "s", the specified [Flag] is ignored.
- When [Minimum field width] is equal to or less than the number of converted characters obtained from (s2), the specifier "-" is ignored.
- When [Conversion specifier] is "u", "x", "X", "o", "s", or "c", the specifiers "+" and " " (space) are ignored.
- When [Conversion specifier] is "d", "u", "s", or "c", or when [Conversion specifier] is specified as "o", "x", or "X" and "0" is specified in (s2), the specifier "#" is ignored.

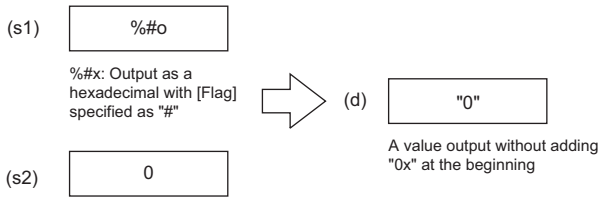
Ex.

When "o" is specified in [Conversion specifier] and "#" is specified in [Flag], the output is as follows.



Ex.

When "x" is specified in [Conversion specifier] and "#" is specified in [Flag], the output is as follows.



- Ignored if a value other than 0 or a positive number is output when "+" and " " (space) are used.
- "-" is added at the beginning when "+" is not specified and a negative number is output.

■Minimum field value

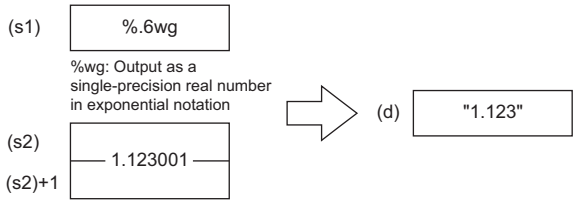
If the specified [Minimum field width] exceeds 255 characters, an error (error code: 329CH) occurs.

■Accuracy

- If the specified [Accuracy] exceeds 255 characters, an error (error code: 329CH) occurs.
- When [Conversion specifier] is "g" or "G", "0"s at the end after the decimal point are not output. "." (period) is also not displayed if there is no numerical value after the decimal point.

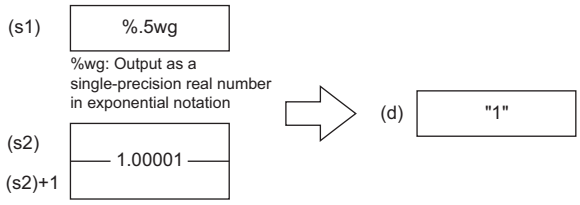
Ex.

When the number of significant digits is six, 0s at the end after the decimal point (0s at the fourth and fifth places after the decimal point) are not output.



Ex.

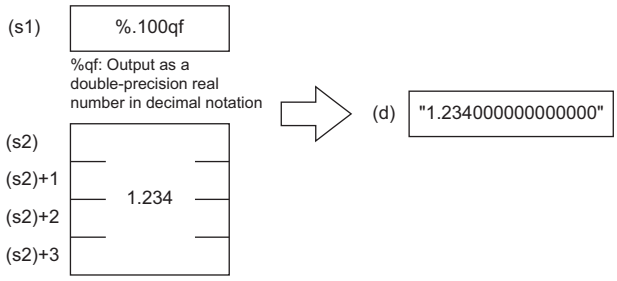
When the number of significant digits is five, "." (period) is not output because all the digits after the decimal point are 0s (no numerical values).



- When [Conversion specifier] is "f", "e", "E", "g", or "G" and the numerical value specified in [Accuracy] is larger than the maximum number of significant digits of the output result, the value is output with the maximum number of significant digits.

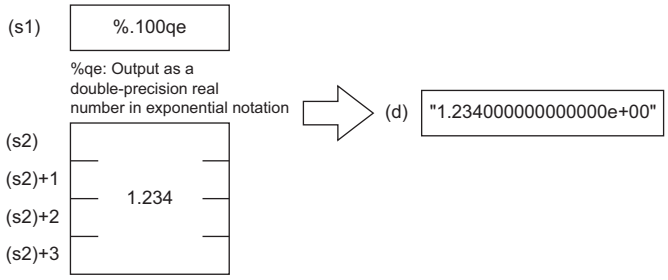
Ex.

When "f" is specified in [Conversion specifier] and the numerical value specified in [Accuracy] is larger than the maximum number of significant digits of the double-precision real number, the value is output with the maximum number of significant digits (15 digits) of the double-precision real number.



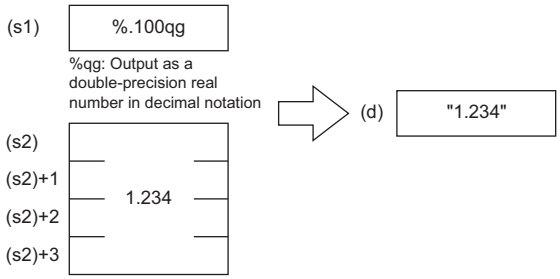
Ex.

When "e" is specified in [Conversion specifier] and the numerical value specified in [Accuracy] is larger than the maximum number of significant digits of the double-precision real number, the value is output with the maximum number of significant digits (15 digits) of the double-precision real number.



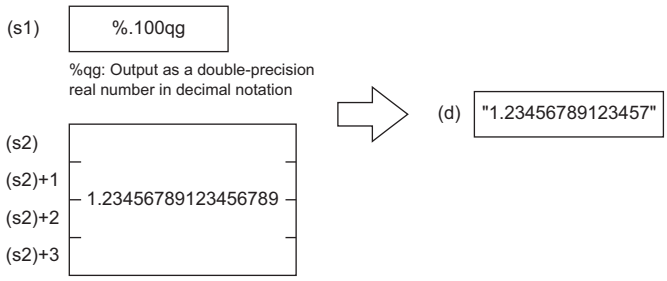
Ex.

When "g" is specified in [Conversion specifier], the numerical value specified in [Accuracy] is larger than the maximum number of significant digits of the double-precision real number, and the number of digits of the numerical value stored in (s2) is equal to or less than the maximum number of significant digits, "0"s at the end after the decimal point are not output.



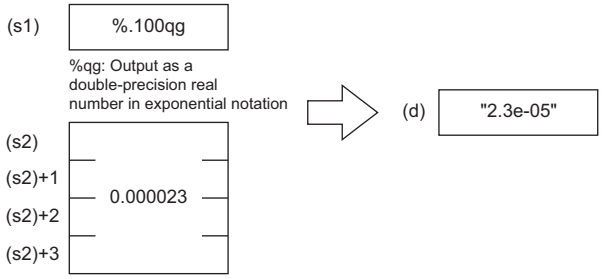
Ex.

When "g" is specified in [Conversion specifier], the numerical value specified in [Accuracy] is larger than the maximum number of significant digits of the double-precision real number, and the number of digits of the numerical value stored in (s2) is equal to or larger than the maximum number of significant digits, the value is output with the maximum number of significant digits (15 digits) which is obtained by rounding off the 16th digit.



Ex.

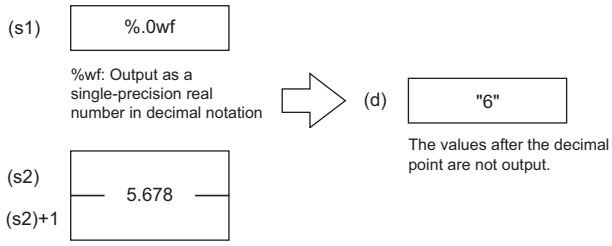
When "g" is specified in [Conversion specifier], the numerical value specified in [Accuracy] is larger than the maximum number of significant digits of the double-precision real number, and the value is output in exponential notation, "0"s at the end after the decimal point are not output.



- When "f", "e", or "E" is specified in [Conversion specifier], ".0" is specified in [Accuracy], and "#" is not specified in [Flag], the digits after the decimal point are not output.

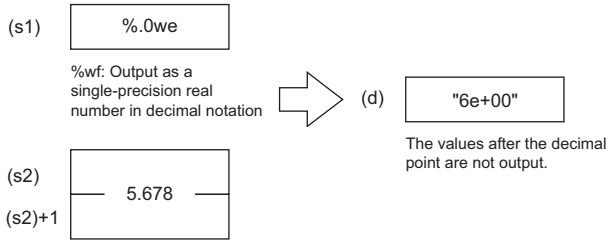
Ex.

When [Conversion specifier] is "f", the expression is as follows if ".0" is specified in [Flag].



Ex.

When [Conversion specifier] is "e", the expression is as follows if ".0" is specified in [Flag].



- When [Conversion specifier] is "d", "f", "e", "E", "g", or "G", the sign of the converted string is not included in the number of output characters specified in [Accuracy].

■Acquisition data size

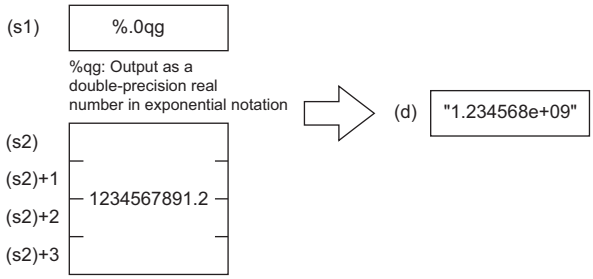
- When [Conversion specifier] is "s" or "c", the specified [Acquisition data size] is ignored.
- When [Conversion specifier] is "f", "e", "E", "g", or "G", always specify "w" or "q" in [Acquisition data size]. If the description is omitted, an error (error code: 3299H) occurs.

■Conversion specifier

When a string is output in exponential notation, an exponent with two or more digits is output.

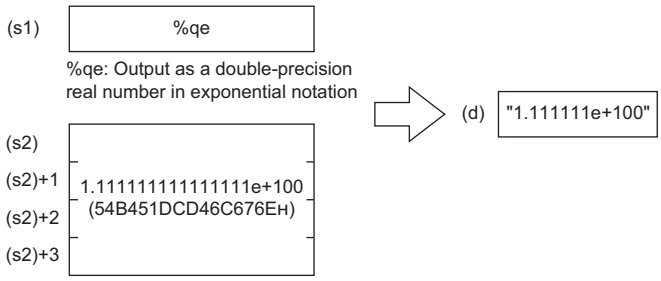
Ex.

When the data stored in (s2) is expressed exponentially and the number of digits of the exponent part is two, the exponent part is output as two digits as follows.



Ex.

When the data stored in (s2) is expressed exponentially and the number of digits of the exponent part is three, the exponent part is output as three digits as follows.



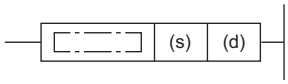
Operation error

Error code (SD0)	Description
2820H	00H does not exist between the device number specified by (s2) and the last device number of the relevant device. (When [Conversion specifier] is specified as "s")
	The size specified by (s2) + [Acquisition data size] exceeds the end device number of (s2). (When [Conversion specifier] is not specified as "s")
	00H does not exist between the device number specified by (s1) and the last device number of the relevant device.
2821H	The device numbers for storing the strings in the devices specified by (s1) and (d) are overlapping.
3285H	The number of characters of the string in the device specified by (s1) exceeds 255.
3298H	The following display format is used for (s1). • A specifier that cannot be used is used. • The order of the display format is not the specified one. • [Accuracy] or [Acquisition data size] is used multiple times for a single display format.
3299H	The display format is not specified for (s1).
329AH	The number of characters of the string in the device specified by (s2) exceeds 255. (When [Conversion specifier] is specified as "s")
3286H	The string stored in the device specified by (d) is out of the output enable range. • The number of characters in the concatenated string exceeds 255. • The entire string that was converted in the assignment range of the relevant device/label for the specified device number and later cannot be stored.
329CH	The specified [Minimum field width] or [Accuracy] exceeds 255 characters.

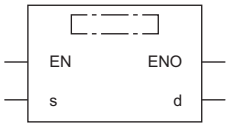
Converting Unicode string to Shift JIS string

WS2SJIS(P)

These instructions convert a Unicode string to a Shift JIS string.

Ladder	ST
	ENO:=WS2SJIS(EN,s,d); ENO:=WS2SJISP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
WS2SJIS	
WS2SJISP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the string to be converted	—	Unicode string	ANYSTRING_DOUBLE
(d)	Start device for storing the converted string	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

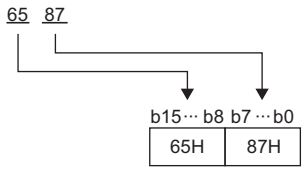
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the Unicode string in the device specified by (s) to the shift JIS string, and stores the converted data in the device specified by (d).
- The Unicode string in the device specified by (s) should be specified in little endian.

Ex.

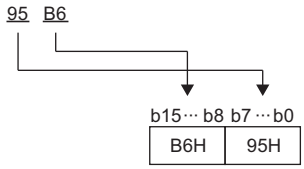
A Unicode string "6587H" is specified with "6587H".



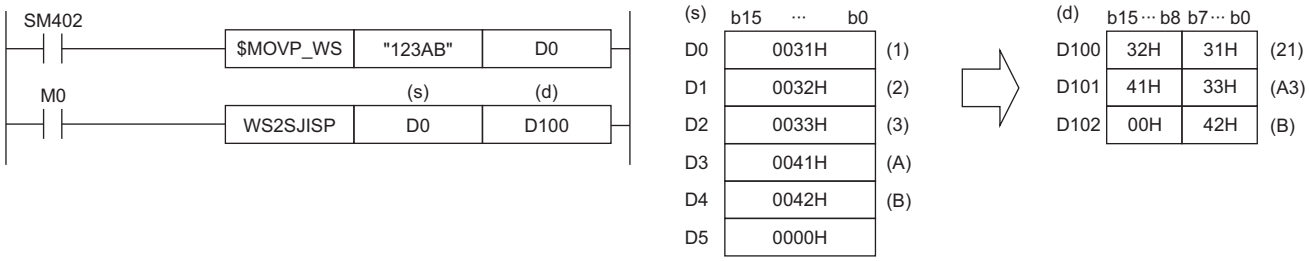
- The Shift JIS string in (d) is stored in big endian.

Ex.

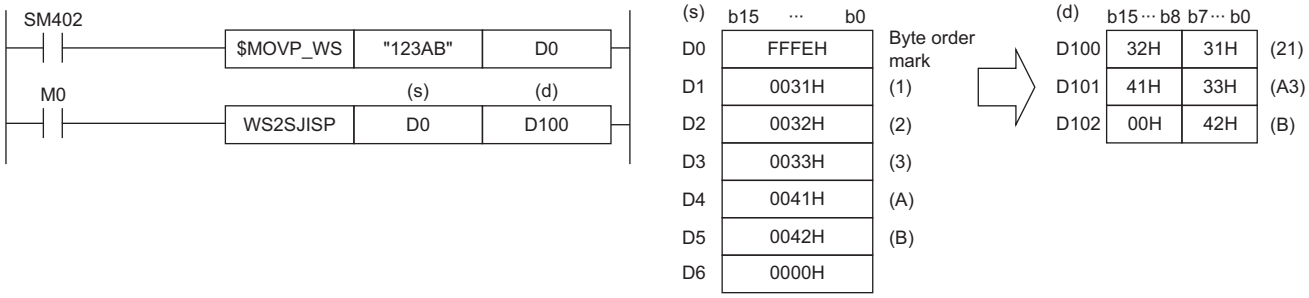
A Shift JIS string "95B6H" is specified with "B695H".



- When a byte order mark is not used, conversion from Unicode to shift JIS occurs as follows.



- When a byte order mark (FEFFH) is added, conversion from Unicode to shift JIS occurs as follows.



Operation error

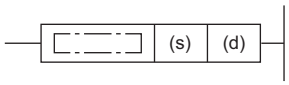
Error code (SD0)	Description
2821H	The ranges of data in the devices specified by (s) and (d) are overlapping.
3281H	Byte order mark FEFFH (big endian) is added to the string in the device specified by (s).
	The range of data in the device specified by (s) includes a character code that cannot be converted.
3285H	The string in the device specified by (s) exceeds 16383 characters.*1
3286H	The entire string that was converted in the assignment range of the relevant device/label for the device number specified by (d) and later cannot be stored.

*1 A two-byte character such as a kanji character represented in shift JIS code should be counted as 2.

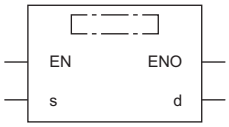
Converting Shift JIS string to Unicode string (without byte order mark)

SJIS2WS(P)

These instructions convert a Shift JIS string to a Unicode string.

Ladder	ST
	ENO:=SJIS2WS(EN,s,d); ENO:=SJIS2WSP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SJIS2WS	
SJIS2WSP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the string to be converted	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the converted string	—	Unicode string	ANYSTRING_DOUBLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

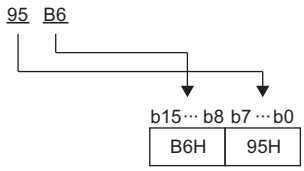
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the shift JIS string in the device specified by (s) to a Unicode string, and store the converted data in the device specified by (d).
- The shift JIS string in the device specified by (s) should be specified in big endian.

Ex.

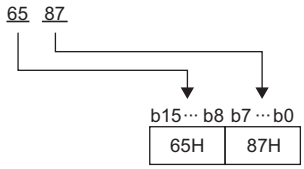
A Shift JIS string "95B6H" is specified with "B695H".



- The Unicode string in (d) is stored in little endian.

Ex.

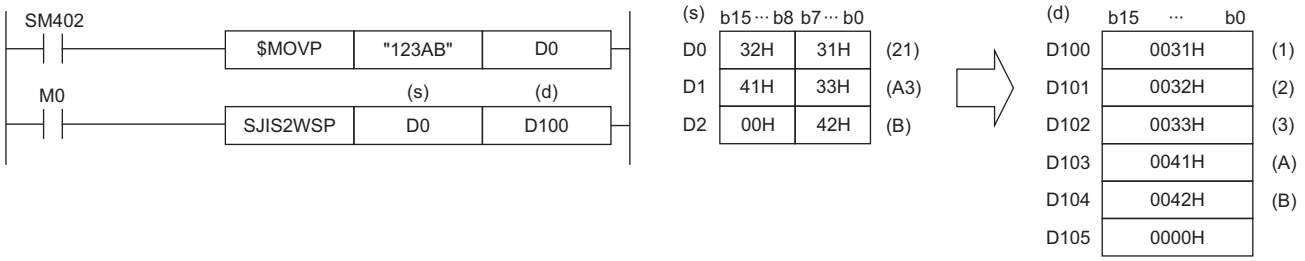
A Unicode string "6587H" is specified with "6587H".



- The SJIS2WS(P) instruction does not add a byte order mark to the beginning of the data in the device specified by (d). To add a byte order mark, use the SJIS2WSB(P) instruction.

☞ Page 810 SJIS2WSB(P)

- The following figure shows the operation for converting shift JIS to Unicode.



Operation error

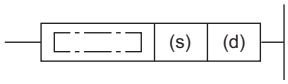
Error code (SD0)	Description
2821H	The ranges of data in the devices specified by (s) and (d) are overlapping.
3281H	The range of data in the device specified by (s) includes a character code that cannot be converted.
3285H	The string in the device specified by (s) exceeds 16383 characters.*1
3286H	The entire string that was converted in the assignment range of the relevant device/label for the device number specified by (d) and later cannot be stored.

*1 A two-byte character such as a kanji character represented in shift JIS code should be counted as 2.

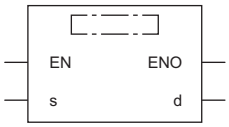
Converting shift JIS string to Unicode (with byte order mark)

SJIS2WSB(P)

These instructions convert a shift JIS string to a Unicode string, and adds a byte order mark to the start of the converted data.

Ladder	ST
	ENO:=SJIS2WSB(EN,s,d); ENO:=SJIS2WSBP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SJIS2WSB	
SJIS2WSBP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the string to be converted	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the converted string	—	Unicode string	ANYSTRING_DOUBLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

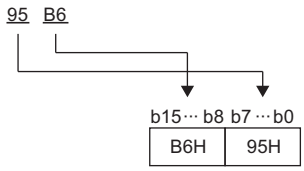
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the shift JIS string in the device specified by (s) to the Unicode string, add a byte order mark to the start of the converted data, and store it in the device specified by (d).
- The shift JIS string in the device specified by (s) should be specified in big endian.

Ex.

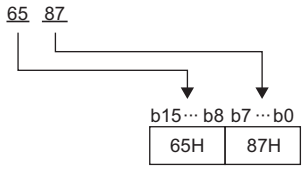
A Shift JIS string "95B6H" is specified with "B695H".



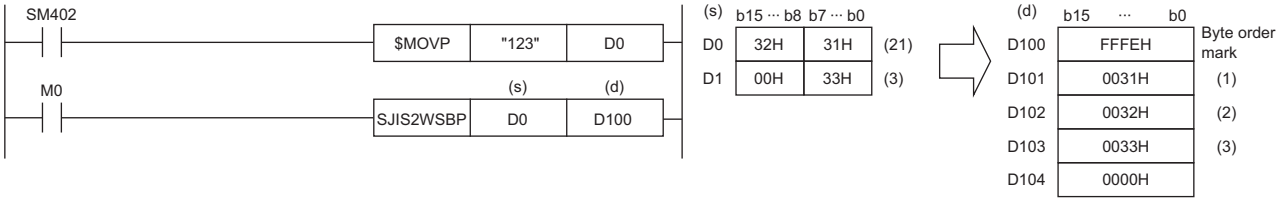
- The Unicode string in (d) is stored in little endian.

Ex.

A Unicode string "6587H" is specified with "6587H".



- The following figure shows the operation for converting shift JIS to Unicode.



Operation error

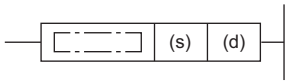
Error code (SD0)	Description
2821H	The ranges of data in the devices specified by (s) and (d) are overlapping.
3281H	The range of data in the device specified by (s) includes a character code that cannot be converted.
3285H	The string in the device specified by (s) exceeds 16383 characters.*1
3286H	The entire string that was converted in the assignment range of the relevant device/label for the device number specified by (d) and later cannot be stored.

*1 A two-byte character such as a kanji character represented in shift JIS code should be counted as 2.

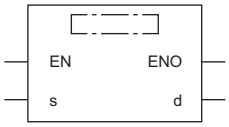
Detecting a string length

LEN(P)

These instructions detect the length of the specified string.

Ladder	ST*1
	ENO:=LENP(EN,s,d);

FBD/LD*1



*1 The LEN instruction does not support the ST and FBD/LD. Use the standard function, LEN.
☞ Page 1327 LEN(_E)

■Execution condition

Instruction	Execution condition
LEN	
LENP	

Setting data

■Description, range, data type

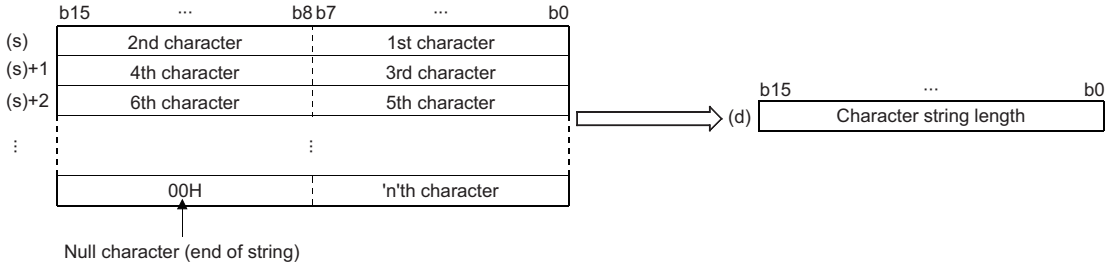
Operand	Description	Range	Data type	Data type (label)
(s)	String or the start device containing the string	—	String	ANYSTRING_SINGLE
(d)	Number of the device for storing the length of the detected string	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

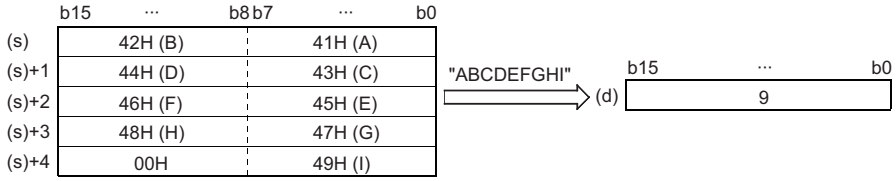
Processing details

- These instructions detect the length of the string in the device specified by (s) and store it in the device number specified by (d) and later. The data stored in device numbers starting from the one specified by (s) to the one containing 00H is processed as a string.



Ex.

When "ABCDEFGHI" is stored in the device specified by (s) and later



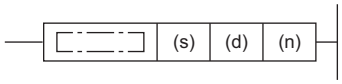
Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) set in the assignment range of the relevant device/label for the device number specified by (s) and later.
3285H	The number of characters in the string specified by (s) exceeds 16383.

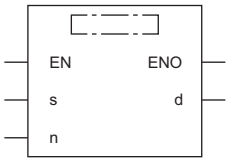
Extracting string data from the right

RIGHT(P)

These instructions extract (n) characters of data from the right of string data.

Ladder	ST*1
	ENO:=RIGHTP(EN,s,n,d);

FBD/LD*1



*1 The RIGHT instruction does not support the ST and FBD/LD. Use the standard function, RIGHT.
☞ Page 1329 LEFT(_E), RIGHT(_E)

■Execution condition

Instruction	Execution condition
RIGHT	
RIGHTP	

Setting data

■Descriptions, ranges, and data types

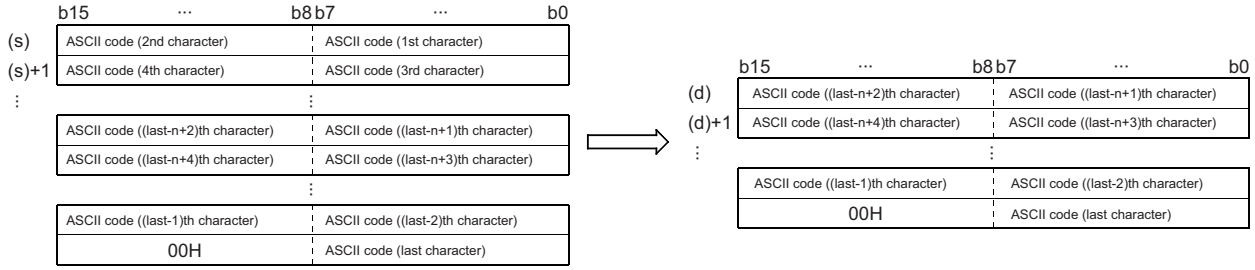
Operand	Description	Range	Data type	Data type (label)
(s)	String or the start device containing the string	—	String	ANYSTRING_SINGLE
(d)	Start device for storing (n) characters of string extracted from the right of the data in the device specified by (s)	—	String	ANYSTRING_SINGLE
(n)	Number of characters to be extracted	1 to 16383	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

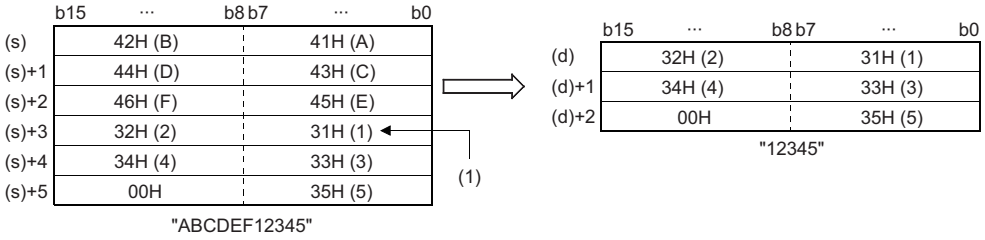
Processing details

- These instructions extract (n) characters of data from the right of the string data (the end of the string) stored in the device number specified by (s) and later, and store the extracted data in the device number specified by (d) and later.



Ex.

When (n)=5



(1) ASCII code (5th character)

- The NULL code (00H) indicating the end of a string is automatically added to the end of the string data.
- When the number of characters in the device specified by (n) is 0, NULL code (00H) is stored in the device specified by (n).

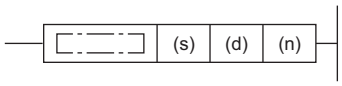
Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) set in the assignment range of the relevant device/label for the device number specified by (s) and later.
3285H	An out-of-range value is set to (s). • The number of characters of the string in the device specified by (s) exceeds 16383. • The number of characters of the string in the device specified by (s) is 0.
	The number of characters in the device specify by (n) exceeds that in the device specified by (s).
3286H	The entire string that was extracted in the assignment range of the relevant device/label for the device number specified by (d) and later cannot be stored.

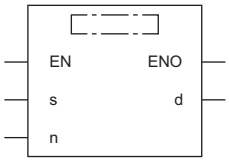
Extracting string data from the left

LEFT(P)

These instructions extract (n) characters of data from the left of the string data, and store the extracted data in the device number specified by (d) and later.

Ladder	ST*1
	ENO:=LEFTP(EN,s,n,d);

FBD/LD*1



*1 The LEFT instruction does not support the ST and FBD/LD. Use the standard function, LEFT.
☞ Page 1329 LEFT(_E), RIGHT(_E)

■Execution condition

Instruction	Execution condition
LEFT	
LEFTP	

Setting data

■Descriptions, ranges, and data types

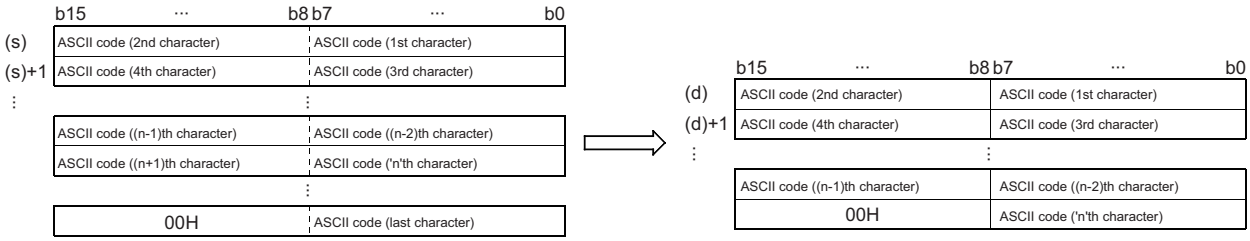
Operand	Description	Range	Data type	Data type (label)
(s)	String or the start device containing the string	—	String	ANYSTRING_SINGLE
(d)	Start device for storing (n) characters of string extracted from the left of the data in the device specified by (s)	—	String	ANYSTRING_SINGLE
(n)	Number of characters to be extracted	1 to 16383	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

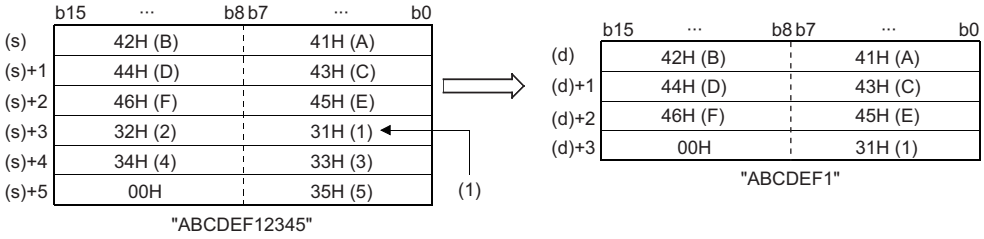
Processing details

- These instructions extract (n) characters of data from the left of the string data (the start of the string) stored in the device number specified by (s) and later, and store the extracted data in the device number specified by (d) and later.



Ex.

When (n)=7



(1) ASCII code (7th character)

- The NULL code (00H) indicating the end of a string is automatically added to the end of the string data.
- When the number of characters in the device specified by (n) is 0, NULL code (00H) is stored in the device specified by (n).

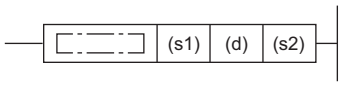
Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) set in the assignment range of the relevant device/label for the device number specified by (s) and later.
3285H	An out-of-range value is set to (s). • The number of characters of the string in the device specified by (s) exceeds 16383. • The number of characters of the string in the device specified by (s) is 0.
	The number of characters in the device specify by (n) exceeds that in the device specified by (s).
3286H	The entire string that was extracted in the assignment range of the relevant device/label for the device number specified by (d) and later cannot be stored.

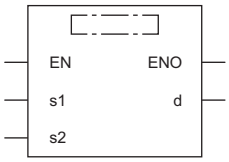
Extracting the specified string data

MIDR(P)

These instructions extract data at any position in string data, and store the extracted data in the device number specified by (d) and later.

Ladder	ST
	ENO:=MIDR(EN,s1,s2,d); ENO:=MIDRP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
MIDR	
MIDRP	

Setting data

■Description, range, data type

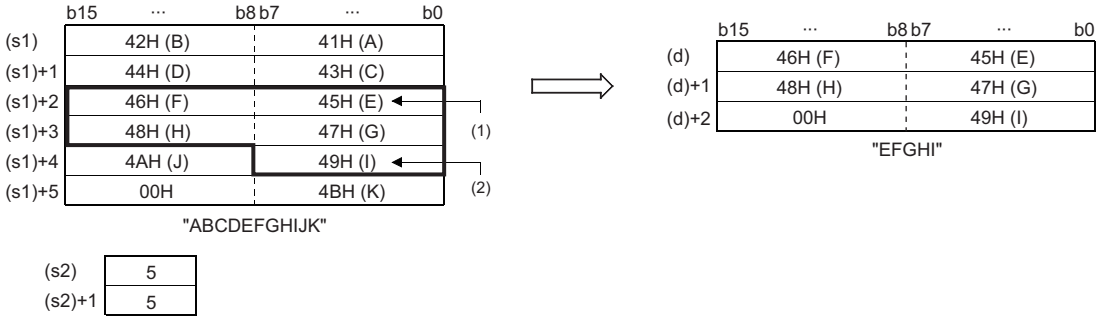
Operand	Description	Range	Data type	Data type (label)
(s1)	String or the start device containing the string	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the string data of the operation result	—	String	ANYSTRING_SINGLE
(s2)	Start device for storing the location of the start character and the number of characters (s2): Location of start character, (s2)+1: Number of characters	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

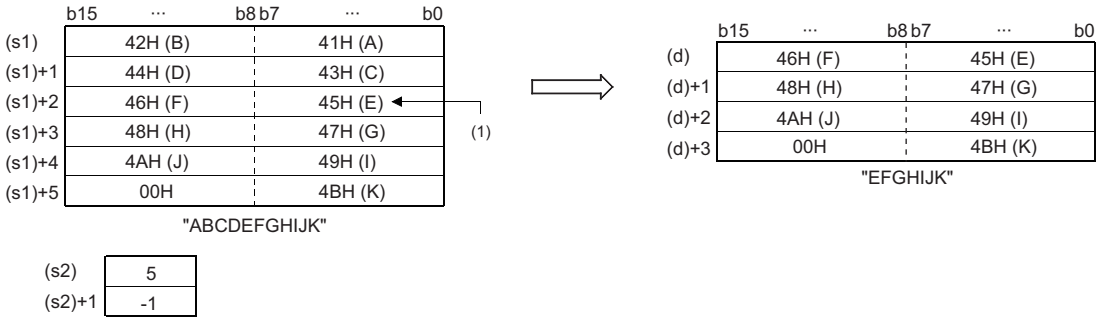
- These instructions extract the data by the number of characters specified by (s2)+1 from the location specified by (s2) in the string data stored in the device number specified by (s1) and later, and store the extracted data in the device number specified by (d) and later.



(1) Position specified by (s2): 5th character

(2) ASCII code for 5th character (s2)+1

- The NULL code (00H) indicating the end of a string is automatically added to the end of the string data.
- If the number of characters in the device specified by (s2)+1 is 0, no processing is performed.
- When the number of characters of the data in the device specified by (s2)+1 is -1, the data till the last character data in the device specified by (s1) is stored in the device specified by (d) and later.



(1) Position specified by (s2): 5th character

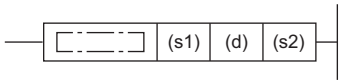
Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the assignment range of the relevant device/label for the device number specified by (s1) and later.
3285H	The number of characters of the string in the device specified by (s1) exceeds 16383.
	An out-of-range value is set to (s2). • The value in the device specified by (s2) is 0 or less. • The value in the device specified by (s2)+1 is other than the valid values (-1, 0, 1 or bigger). • The value in the device specify by (s2) exceeds the number of characters in the device specified by (s1). • The value obtained by adding (s2) and (s2)+1 exceeds the number of characters in the device specified by (s1) plus 1.
3286H	The entire string that was extracted in the assignment range of the relevant device/label for the device number specified by (d) and later cannot be stored.

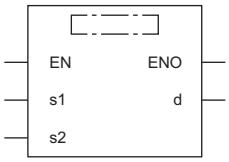
Replacing the specified string data

MIDW(P)

These instructions replace the data at the specified location in the string data with the specified string.

Ladder	ST
	ENO:=MIDW(EN,s1,s2,d); ENO:=MIDWP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
MIDW	
MIDWP	

Setting data

■Description, range, data type

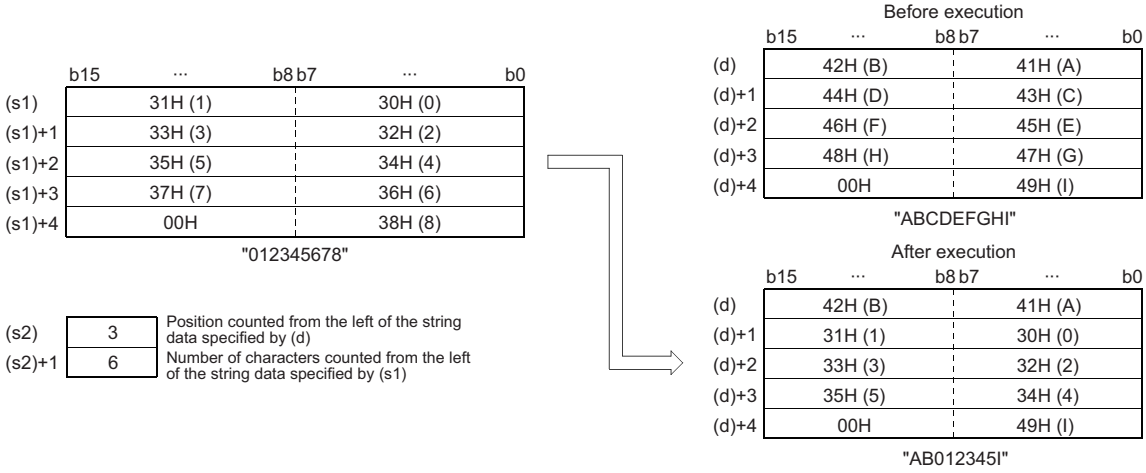
Operand	Description	Range	Data type	Data type (label)
(s1)	String or the start device containing the string	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the string data of the operation result	—	String	ANYSTRING_SINGLE
(s2)	Start device for storing the location of the start character and the number of characters (s2): Location of start character, (s2)+1: Number of characters	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

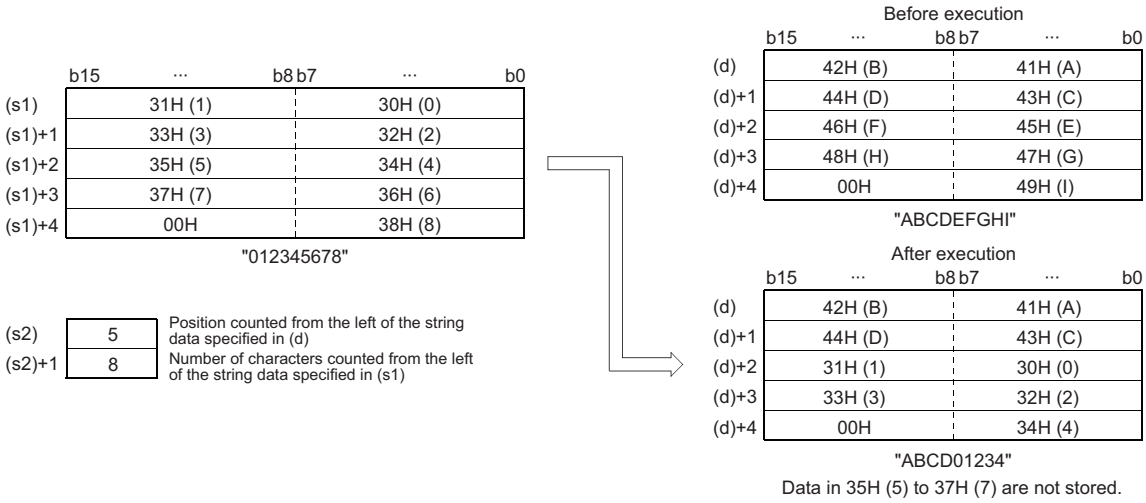
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

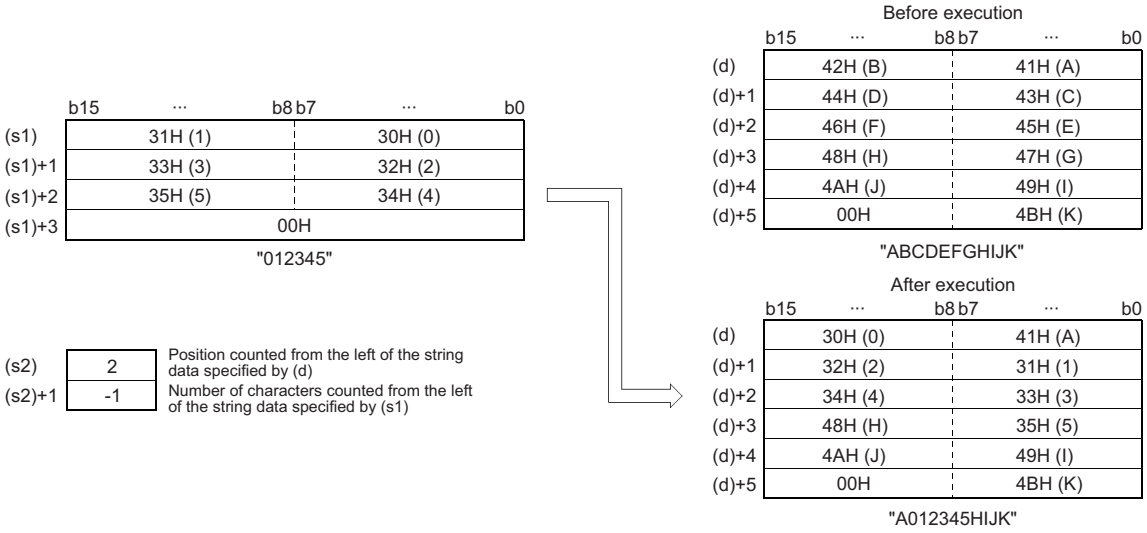
- These instructions read data by the number of characters stored in the device specified by (s2)+1 from the string data stored in the device number specified by (s1) and later, and store the read data at the location in the device specified by (s2) and later in the string stored in the device number specified by (d) and later.



- The NULL code (00H) indicating the end of a string is automatically added to the end of the string data.
- If the number of characters in the device specified by (s2)+1 is 0, no processing is performed.
- If the number of characters in the device specified by (s2)+1 exceeds the last character of the string data in the device specified by (d), the data is stored up to the last character.



- When the number of characters of the data in the device specified by (s2)+1 is -1, the data till the last character data in the device specified by (s1) is stored in the device specified by (d) and later.



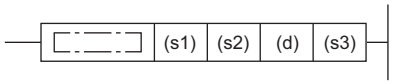
Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the assignment range of the relevant device/label for the device number specified by (s1) and later.
	There is no NULL code (00H) in the assignment range of the relevant device/label for the device number specified by (d) and later.
3285H	The number of characters of the string in the device specified by (s1) exceeds 16383.
	The number of characters of the string in the device specified by (d) exceeds 16383.
	An out-of-range value is set to (s2). • The value in the device specified by (s2) is 0 or less. • The value in the device specified by (s2)+1 is other than the valid values (-1, 0, 1 or bigger). • The value in the device specify by (s2) exceeds the number of characters in the device specified by (d). • The value in the device specify by (s2)+1 exceeds the number of characters in the device specified by (s1).

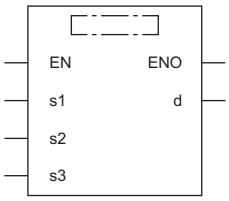
Searching string data

INSTR(P)

These instructions search string data for the specified string.

Ladder	ST
	ENO:=INSTR(EN,s1,s2,s3,d); ENO:=INSTRP(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
INSTR	
INSTRP	

Setting data

■Descriptions, ranges, and data types

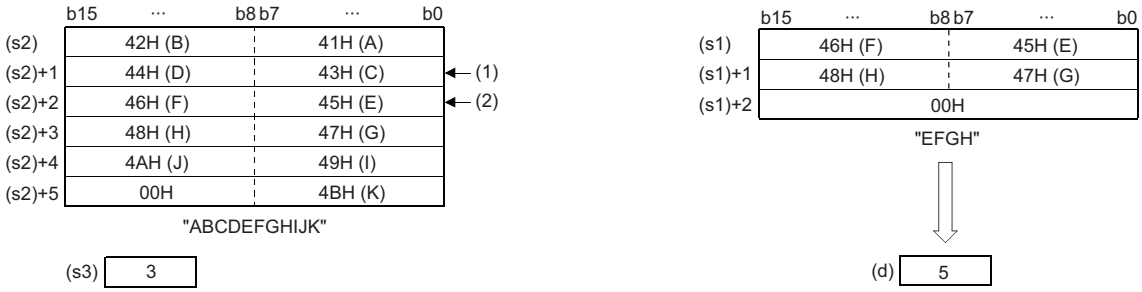
Operand	Description	Range	Data type	Data type (label)
(s1)	Strings to be searched or the start device containing these strings	—	String	ANYSTRING_SINGLE
(s2)	String to be searched for or the start device containing the string to be searched for	—	String	ANYSTRING_SINGLE
(d)	Device for storing the search result	—	16-bit signed binary	ANY16
(s3)	Search start position	1 to 16383	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(s3)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search the string data stored in the device number specified by (s2) and later starting from the (s3)th character from the left for the string data stored in the device number specified by (s1) and later, and store the search result in the device specified by (d). The search result stored shows the number of characters from the start character of the string data in the device specified by (s2).



(1) Starting position specified by (s3): 3rd character
(2) 5th character from the beginning of the data

- If no string data is matching, 0 is stored in the device specified by (d).

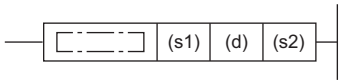
Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the assignment range of the relevant device/label for the device specified by (s1) and later.
	There is no NULL code (00H) in the assignment range of the relevant device/label for the device specified by (s2) and later.
3285H	An out-of-range value is set to (s1). • The number of characters of the string that has been set is 0. • The number of characters of the string that has been set exceeds 16383.
	The number of characters of the string that has been set in the device specified by (s2) exceeds 16383.
	An out-of-range value is set to (s3). • The value in the device specify by (s3) exceeds the number of characters in the device specified by (s2). • The value in the device specified by (s3) is negative or 0.

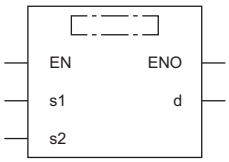
Inserting string data

STRINS(P)

These instructions insert the specified string data into the specified position of the string data.

Ladder	ST
	ENO:=STRINS(EN,s1,s2,d); ENO:=STRINSP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
STRINS	
STRINSP	

Setting data

■Descriptions, ranges, and data types

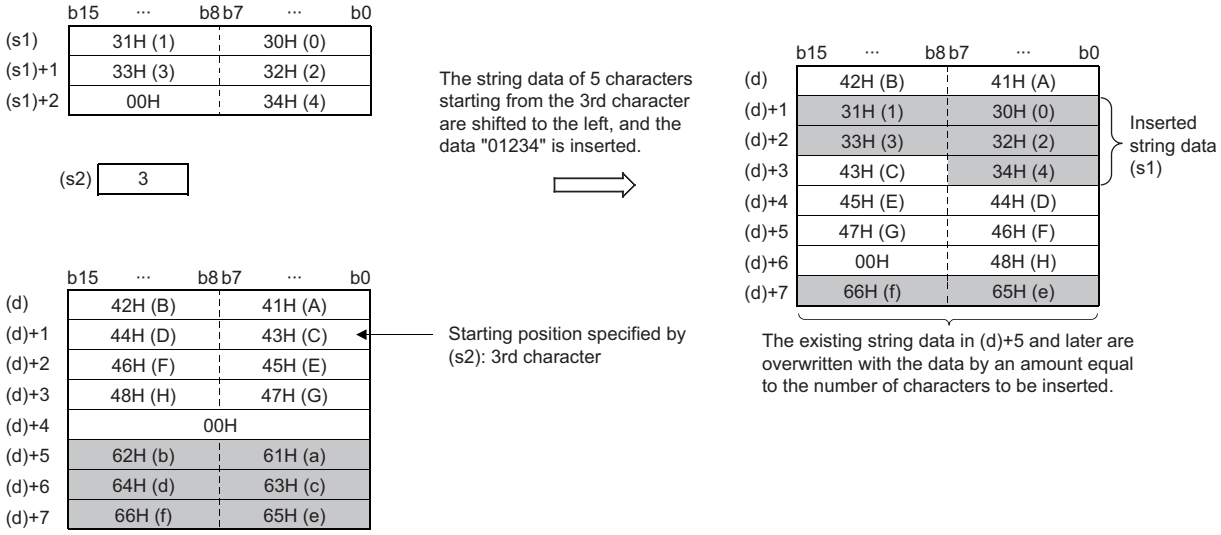
Operand	Description	Range	Data type	Data type (label)
(s1)	String to be inserted or the start device containing the string to be inserted	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the insertion result string	—	String	ANYSTRING_SINGLE
(s2)	Insertion position (bytes)	1 to 16383	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions insert the string data in the device specified by (s1) to the (s2)th character (insertion position) from the start of the string data in the device specified by (d).



- If the string after insertion in the device specified by (s1)+(d) is even, the NULL code (00H) is stored in the device (1 word) next to the last one containing the string.
- If the string after insertion in the device specified by (s1)+(d) is odd, the NULL code (00H) is stored in the last device (upper 8 bits) of the string.
- When the number of characters in the device specified by (d) plus 1 is specified in (s2), the string in the device specified by (s1) is concatenated to the end of the string in the device specified by (d).

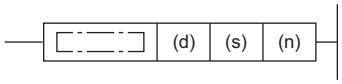
Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the assignment range of the relevant device/label for the device specified by (s1) and later.
	There is no NULL code (00H) in the assignment range of the relevant device/label for the device specified by (d) and later.
2821H	The devices specified by (s1) and (d), both containing strings, are overlapping even partly.
	The device specified by (s1)+(d) containing the string after insertion overlaps with the string storage device specified by (s1).
3285H	The number of characters of the string in the device specified by (s1) exceeds 16383.
	An out-of-range value is set to (s2). • The specified value exceeds the number of characters plus 1 of the string in the device specified by (d). • The specified value is not within the following range. 1≤(s2)≤16383
	The number of characters of the string in the device specified by (d) exceeds 16383.
3286H	The string after insertion stored in the device specified by (s1)+(d) becomes data outside the output enable range. • The number of characters of the string after insertion exceeds 16383. • The entire string after insertion in the assignment range of the relevant device/label for the device number specified by (d) and later cannot be stored.

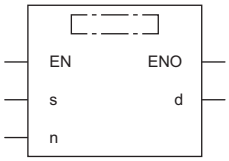
Deleting string data

STRDEL(P)

These instructions delete (n) characters starting from the specified position of string data.

Ladder	ST
	ENO:=STRDEL(EN,s,n,d); ENO:=STRDELP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
STRDEL	
STRDELP	

Setting data

■Descriptions, ranges, and data types

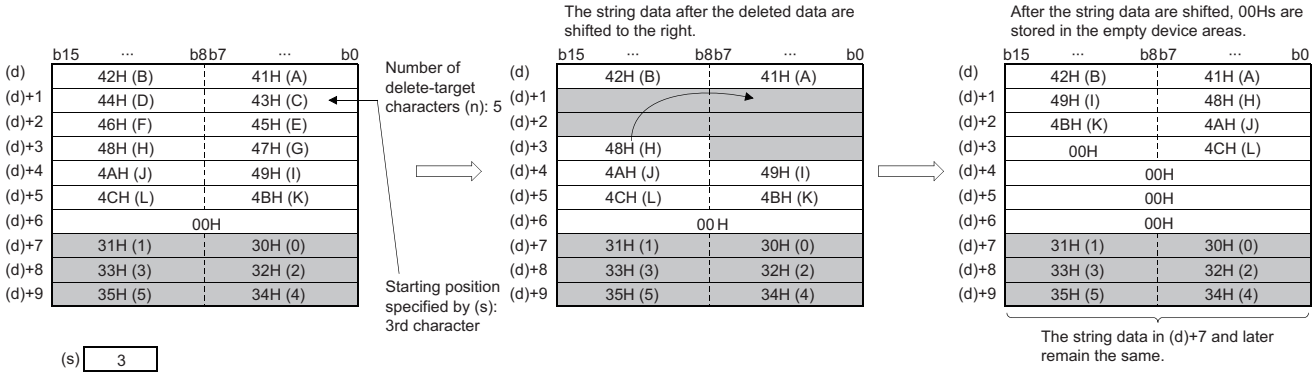
Operand	Description	Range	Data type	Data type (label)
(d)	Start device containing the string to be deleted	—	String	ANYSTRING_SINGLE
(s)	Deletion start position	1 to 16383	16-bit signed binary	ANY16
(n)	Number of characters to be deleted	0 to 16384-(s)	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s)	—	○	○	○	○	—	—	○	○	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions delete (n) characters starting from the position (deletion start position) specified by the (s)th character from the start of the string data in the device specified by (d).



- If the string after deletion in the device specified by (d) is even, the NULL code (00H) is stored in the device (1 word) next to the last one containing the string.
- If the string after deletion in the device specified by (d) is odd, the NULL code (00H) is stored in the last device (upper 8 bits) of the string.
- The string following the deleted one is shifted by (n) characters to the right, and the NULL code (00H) is stored in the device that has been emptied.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the assignment range of the relevant device/label for the device specified by (d) and later.
3285H	The number of characters of the string in the device specified by (d) exceeds 16383.
	An out-of-range value is set to (s). • The specified value is not within the following range. 1≤(s)≤16383 • The specified value exceeds the number of characters of the string in the device specified by (d).
	An out-of-range value is set to (n). • The specified value exceeds the number of characters from the data in the device specified by (s) of the string in the device specified by (d) to the last character. • The specified value is negative.

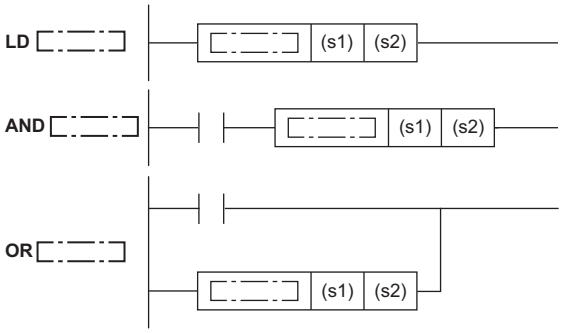
17 REAL VALUE PROCESSING

17.1 Floating-point Instruction

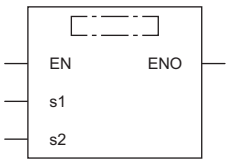
Comparing single-precision real numbers

LDE□, ANDE□, ORE□

These instructions perform a comparison operation of a single-precision real number. (Devices are used as a normally open contact.)

Ladder	ST
 (□ is replaced by E=, E<>, E>, E<=, E<, E>=.)	ENO:=LDE_□(EN,s1,s2); ENO:=ANDE_□(EN,s1,s2); ENO:=ORE_□(EN,s1,s2); (□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)*1

FBD/LD



(□ is replaced by a combination of LDE_, ANDE_, or ORE_ and EQ, NE, GT, LE, LT, or GE.)*1

*1 EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
LDE□, ANDE□, ORE□	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Comparison data or the start device where the comparison data is stored	0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Comparison data or the start device where the comparison data is stored	0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	—	—	○	○	○	○	○	○	—	○	—	—

Processing details

- These instructions perform a comparison operation between the single-precision real number in the device specified by (s1) and the single-precision real number in the device specified by (s2). (Devices are used as a normally open contact.)
- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
E=, EQ	(s1)=(s2)	Continuity state (ENO is on.)
E<>, NE	(s1)≠(s2)	
E>, GT	(s1)>(s2)	
E<=, LE	(s1)≤(s2)	
E<, LT	(s1)<(s2)	
E>=, GE	(s1)≥(s2)	
E=, EQ	(s1)≠(s2)	Non-continuity state (ENO is off.)
E<>, NE	(s1)=(s2)	
E>, GT	(s1)≤(s2)	
E<=, LE	(s1)>(s2)	
E<, LT	(s1)≥(s2)	
E>=, GE	(s1)<(s2)	

- If the data in the device specified by (s1) or (s2) is out of the range of setting data, the operation result will be non-continuity (ENO OFF).
- If the LDE_□ instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.
- If the ORE_□ instruction is used in the program written in FBD/LD and EN is set to TRUE, ENO turns on. EN will not be an execution condition.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

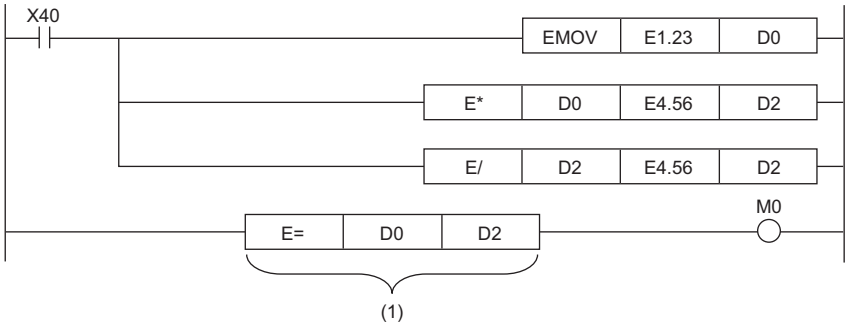
☞ Page 40 Precautions

Operation error

There is no operation error.

Point

Note that two values may not be equal due to an error when the E= instruction is used.



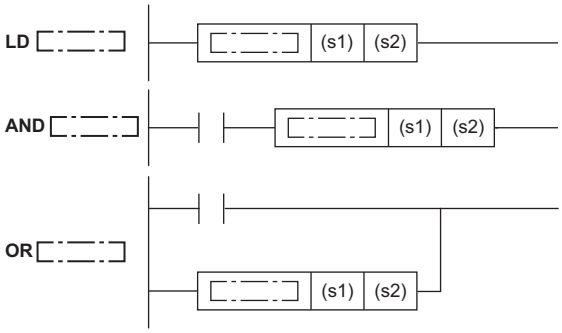
(1) Two values may not be equal.

Comparing double-precision real numbers

LDED□, ANDED□, ORED□

These instructions perform a comparison operation of a double-precision real number. (Devices are used as a normally open contact.)

Ladder

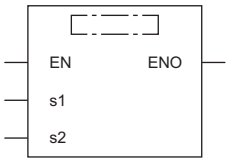


(□ is replaced by ED=, ED<>, ED>, ED<=, ED<, ED>=.)

ST

ENO:=LDED_□(EN,s1,s2);
ENO:=ANDED_□(EN,s1,s2);
ENO:=ORED_□(EN,s1,s2);
(□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)*1

FBD/LD



(□ is replaced by a combination of LDED_, ANDED_, or ORED_ and EQ, NE, GT, LE, LT, or GE.)*1

*1 EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
LDED□, ANDED□, ORED□	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Comparison data or the start device where the comparison data is stored	0, $2^{-1022} \leq \lvert(s1)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Comparison data or the start device where the comparison data is stored	0, $2^{-1022} \leq \lvert(s2)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—

Processing details

- These instructions perform a comparison operation between the double-precision real number in the device specified by (s1) and the double-precision real number in the device specified by (s2). (Devices are used as a normally open contact.)
- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
ED=, EQ	(s1)=(s2)	Continuity state (ENO is on.)
ED<>, NE	(s1)≠(s2)	
ED>, GT	(s1)>(s2)	
ED<=, LE	(s1)≤(s2)	
ED<, LT	(s1)<(s2)	
ED>=, GE	(s1)≥(s2)	
ED=, EQ	(s1)≠(s2)	Non-continuity state (ENO is off.)
ED<>, NE	(s1)=(s2)	
ED>, GT	(s1)≤(s2)	
ED<=, LE	(s1)>(s2)	
ED<, LT	(s1)≥(s2)	
ED>=, GE	(s1)<(s2)	

- If the data in the device specified by (s1) or (s2) is out of the range of setting data, the operation result will be non-continuity (ENO OFF).
- If the LDED_☐ instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.
- If the ORED_☐ instruction is used in the program written in FBD/LD and EN is set to TRUE, ENO turns on. EN will not be an execution condition.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Operation error

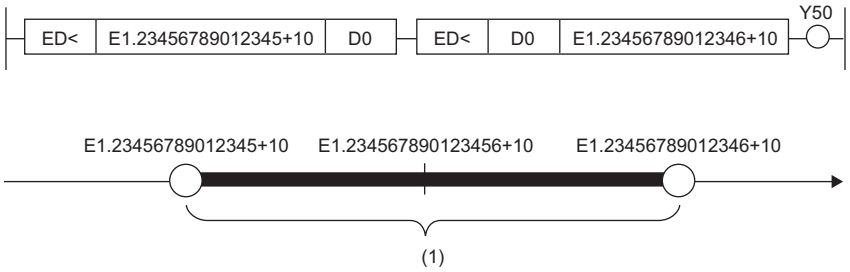
There is no operation error.

Precautions

The maximum number of digits of a real number which can be input using the engineering tool is 15, and therefore these instructions cannot perform comparison with a real number consisting of 16 or more effective digits. When these instructions are used to determine the match or mismatch with a real number consisting of 16 or more effective digits, the instructions need to compare the size with the approximate values before and after the real number to be compared.

Ex.

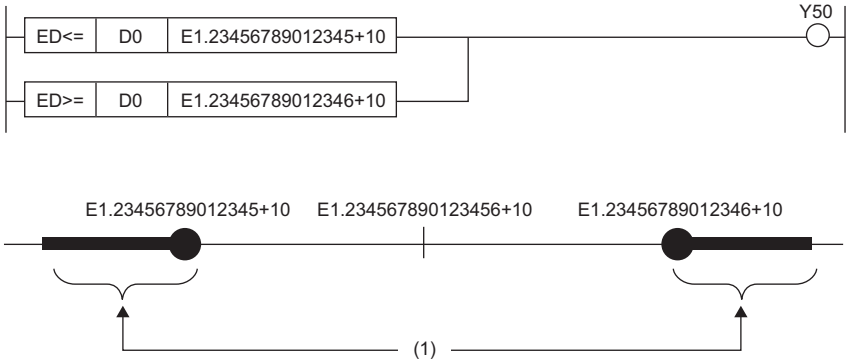
To determine the match between E1.234567890123456+10 (16 effective digits) and a double-precision real number



(1) Whether data in D0 to D3 are within this range is checked. (The boundary values are not included.)

Ex.

To determine the mismatch between E1.234567890123456+10 (16 effective digits) and a double-precision real number

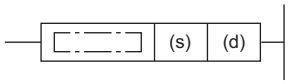


(1) Whether data in D0 to D3 are within this range is checked. (The boundary values are included.)

Adding single-precision real numbers

E+(P) [when two operands are set]

These instructions add single-precision real numbers.

Ladder	ST
	Not supported (☞ Page 836 E+(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 836 E+(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
E+	
E+P	

Setting data

■Description, range, data type

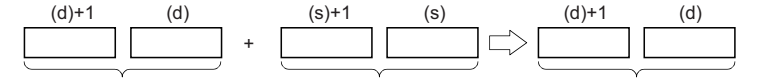
Operand	Description	Range	Data type	Data type (label)
(s)	Second addend data or the start device where the second addend data is stored	0, $2^{-126}\leq\lvert(s)\rvert<2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device where the first addend data is stored	0, $2^{-126}\leq\lvert(d)\rvert<2^{128}$	Single-precision real number	ANYREAL_32

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions add the single-precision real number in the device specified by (s) to the single-precision real number in the device specified by (d), and store the result in the device specified by (d).



- Value 0 or $2^{-126} \leq$|specified value (stored value)| $< 2^{128}$ can be specified or stored in the devices specified by (s) and (d).
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

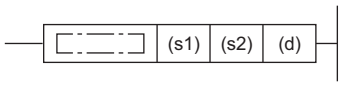
☞ Page 40 Precautions

Operation error

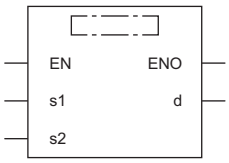
Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{128}$

E+(P) [when three operands are set]

These instructions add single-precision real numbers.

Ladder	ST
	ENO:=EPLUS(EN,s1,s2,d); ENO:=EPLUSP(EN,s1,s2,d);

FBD/LD



(☐ is replaced by either of the following: EPLUS, EPLUSP.)

■Execution condition

Instruction	Execution condition
E+	
E+P	

Setting data

■Description, range, data type

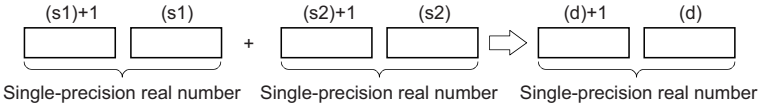
Operand	Description	Range	Data type	Data type (label)
(s1)	First addend data or the start device where the first addend data is stored	0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Second addend data or the start device where the second addend data is stored	0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions add the single-precision real number in the device specified by (s2) to the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).



- Value 0 or $2^{-126} \leq$|specified value (stored value)| $< 2^{128}$ can be specified or stored in the devices specified by (s1), (s2), and (d).

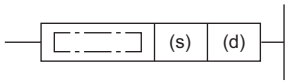
Operation error

Error code (SD0)	Description
3282H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{128}$

Subtracting single-precision real numbers

E-(P) [when two operands are set]

These instructions perform subtraction between single-precision real numbers.

Ladder	ST
	Not supported (☞ Page 840 E-(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 840 E-(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
E-	
E-P	

Setting data

■Description, range, data type

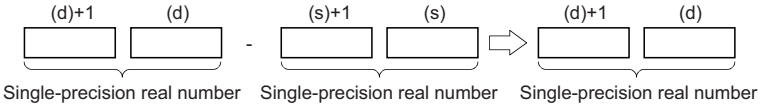
Operand	Description	Range	Data type	Data type (label)
(s)	Subtrahend data or the start device where subtrahend data is stored	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device where minuend data is stored	0, $2^{-126} \leq \lvert(d)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions subtract the single-precision real number in the device specified by (s) from the single-precision real number in the device specified by (d), and store the result in the device specified by (d).



- Value 0 or $2^{-126} \leq$|specified value (stored value)| $< 2^{128}$ can be specified or stored in the devices specified by (s) and (d).
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

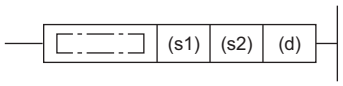
☞ Page 40 Precautions

Operation error

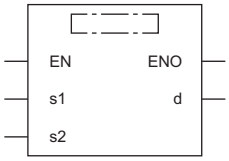
Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{128}$

E-(P) [when three operands are set]

These instructions perform subtraction between single-precision real numbers.

Ladder	ST
	ENO:=EMINUS(EN,s1,s2,d); ENO:=EMINUSP(EN,s1,s2,d);

FBD/LD



(☐ is replaced by either of the following: EMINUS, EMINUSP.)

■Execution condition

Instruction	Execution condition
E-	
E-P	

Setting data

■Description, range, data type

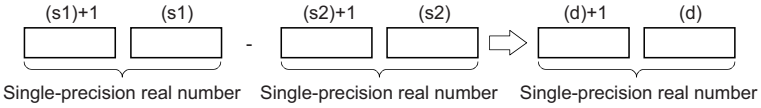
Operand	Description	Range	Data type	Data type (label)
(s1)	Minuend data or the start device where minuend data is stored	0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Subtrahend data or the start device where subtrahend data is stored	0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions subtract the single-precision real number in the device specified by (s2) from the single-precision real number in the device specified by (s1), and store the result in the device specified by (d).



- Value 0 or $2^{-126} \leq$|specified value (stored value)| $< 2^{128}$ can be specified or stored in the devices specified by (s1), (s2), and (d).
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

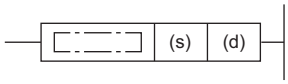
Operation error

Error code (SD0)	Description
3282H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{128}$

Adding double-precision real numbers

ED+(P) [when two operands are set]

These instructions add double-precision real numbers.

Ladder	ST
	Not supported (☞ Page 844 ED+(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 844 ED+(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
ED+	
ED+P	

Setting data

■Description, range, data type

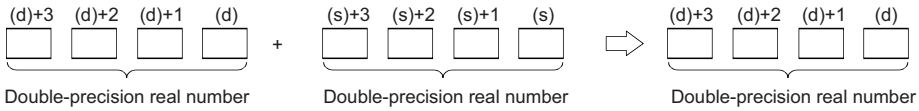
Operand	Description	Range	Data type	Data type (label)
(s)	Second addend data or the start device where the second addend data is stored	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device where the first addend data is stored	0, $2^{-1022} \leq \lvert(d)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions add the double-precision real number in the device specified by (d) to the double-precision real number in the device specified by (s), and store the result in the device specified by (d).



- Value 0 or $2^{-1022} \leq$ |specified value (stored value)| $< 2^{1024}$ can be specified or stored in the devices specified by (s) and (d).
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

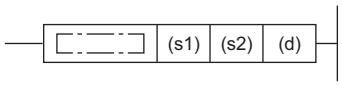
☞ Page 40 Precautions

Operation error

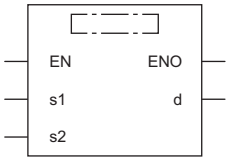
Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{1024}$

ED+(P) [when three operands are set]

These instructions add double-precision real numbers.

Ladder	ST
	ENO:=EDPLUS(EN,s1,s2,d); ENO:=EDPLUSP(EN,s1,s2,d);

FBD/LD



(□ is replaced by either of the following: EDPLUS, EDPLUSP.)

■Execution condition

Instruction	Execution condition
ED+	
ED+P	

Setting data

■Description, range, data type

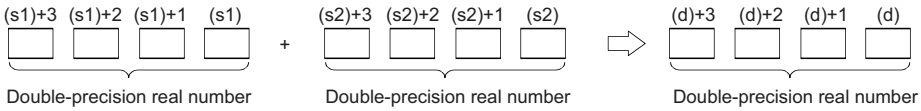
Operand	Description	Range	Data type	Data type (label)
(s1)	First addend data or the start device where the first addend data is stored	0, $2^{-1022} \leq \lvert(s1)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Second addend data or the start device where the second addend data is stored	0, $2^{-1022} \leq \lvert(s2)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions add the double-precision real number in the device specified by (s1) to the double-precision real number in the device specified by (s2), and store the result in the device specified by (d).



- Value 0 or $2^{-1022} \leq$|specified value (stored value)| $< 2^{1024}$ can be specified or stored in the devices specified by (s1), (s2), and (d).
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

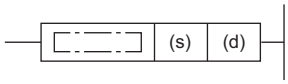
Operation error

Error code (SD0)	Description
3282H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{1024}$

Subtracting double-precision real numbers

ED-(P) [when two operands are set]

These instructions perform subtraction between double-precision real numbers.

Ladder	ST
	Not supported (☞ Page 848 ED-(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 848 ED-(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
ED-	
ED-P	

Setting data

■Description, range, data type

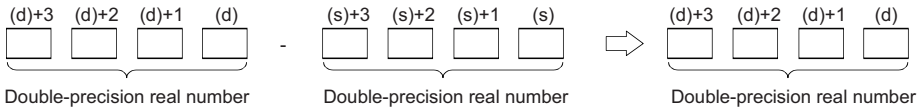
Operand	Description	Range	Data type	Data type (label)
(s)	Subtrahend data or the start device where subtrahend data is stored	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device where minuend data is stored	0, $2^{-1022} \leq \lvert(d)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions subtract the double-precision real number in the device specified by (s) from the double-precision real number in the device specified by (d), and store the result in the device specified by (d).



- Value 0 or $2^{-1022} \leq$|specified value (stored value)| $< 2^{1024}$ can be specified or stored in the devices specified by (s) and (d).
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

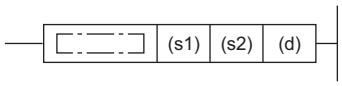
☞ Page 40 Precautions

Operation error

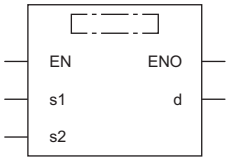
Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{1024}$

ED-(P) [when three operands are set]

These instructions perform subtraction between double-precision real numbers.

Ladder	ST
	ENO:=EDMINUS(EN,s1,s2,d); ENO:=EDMINUSP(EN,s1,s2,d);

FBD/LD



(□ is replaced by either of the following: EDMINUS, EDMINUSP.)

■Execution condition

Instruction	Execution condition
ED-	
ED-P	

Setting data

■Description, range, data type

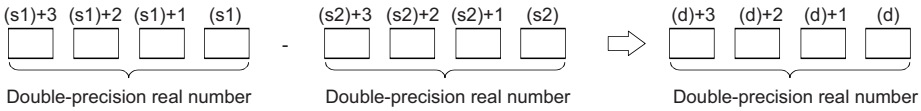
Operand	Description	Range	Data type	Data type (label)
(s1)	Minuend data or the start device where minuend data is stored	0, $2^{-1022} \leq \lvert(s1)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Subtrahend data or the start device where subtrahend data is stored	0, $2^{-1022} \leq \lvert(s2)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions subtract the double-precision real number in the device specified by (s2) from the double-precision real number in the device specified by (s1), and store the result in the device specified by (d).



- Value 0 or $2^{-1022} \leq$|specified value (stored value)| $< 2^{1024}$ can be specified or stored in the devices specified by (s1), (s2), and (d).
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

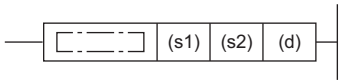
Operation error

Error code (SD0)	Description
3282H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{1024}$

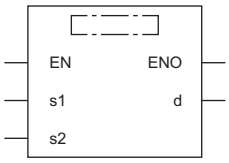
Multiplying single-precision real numbers

E*(P)

These instructions multiply single-precision real numbers.

Ladder	ST
	ENO:=EMULTI(EN,s1,s2,d); ENO:=EMULTIP(EN,s1,s2,d);

FBD/LD



(□ is replaced by either of the following: EMULTI, EMULTIP.)

■Execution condition

Instruction	Execution condition
E*	
E*P	

Setting data

■Description, range, data type

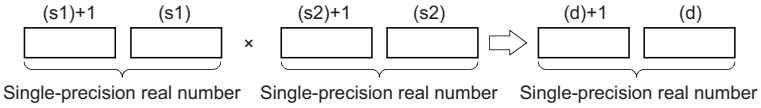
Operand	Description	Range	Data type	Data type (label)
(s1)	Multiplicand data or the start device where multiplicand data is stored	0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Multiplier data or the start device where multiplier data is stored	0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions multiply the single-precision real number in the device specified by (s1) by the single-precision real number in the device specified by (s2), and store the multiplication result in the device specified by (d).



- Value 0 or $2^{-126} \leq$|specified value (stored value)| $< 2^{128}$ can be specified or stored in the devices specified by (s1), (s2), and (d).
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

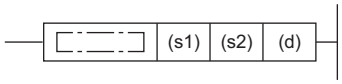
Operation error

Error code (SD0)	Description
3282H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{128}$

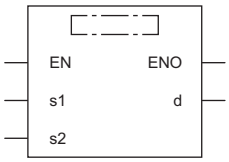
Dividing single-precision real numbers

E/(P)

These instructions perform division between single-precision real numbers.

Ladder	ST
	ENO:=EDIVISION(EN,s1,s2,d); ENO:=EDIVISIONP(EN,s1,s2,d);

FBD/LD



(☐ is replaced by either of the following: EDIVISION, EDIVISIONP.)

■Execution condition

Instruction	Execution condition
E/	
E/P	

Setting data

■Description, range, data type

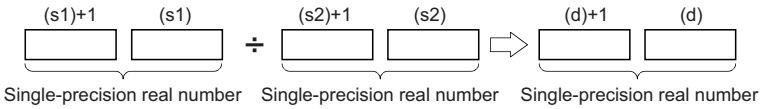
Operand	Description	Range	Data type	Data type (label)
(s1)	Dividend data or the start device where dividend data is stored	0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Divisor data or the start device where divisor data is stored	0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions divide the single-precision real number in the device specified by (s1) by the single-precision real number in the device specified by (s2), and store the division result in the device specified by (d).



- Value 0 or $2^{-126} \leq |$specified value (stored value)$| < 2^{128}$ can be specified or stored in the devices specified by (s1), (s2), and (d).
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

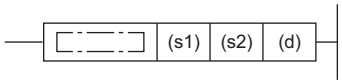
Operation error

Error code (SD0)	Description
3280H	The data (divisor) in the device specified by (s2) is 0.
3282H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{128}$

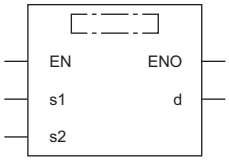
Multiplying double-precision real numbers

ED*(P)

These instructions multiply double-precision real numbers.

Ladder	ST
	ENO:=EDMULTI(EN,s1,s2,d); ENO:=EDMULTIP(EN,s1,s2,d);

FBD/LD



(☐ is replaced by either of the following: EDMULTI, EDMULTIP.)

■Execution condition

Instruction	Execution condition
ED*	
ED*P	

Setting data

■Description, range, data type

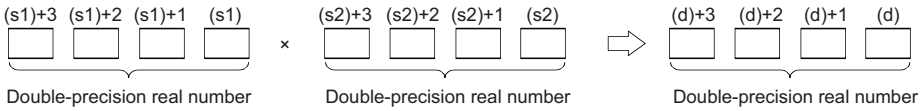
Operand	Description	Range	Data type	Data type (label)
(s1)	Multiplicand data or the start device where multiplicand data is stored	0, $2^{-1022} \leq \lvert(s1)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Multiplier data or the start device where multiplier data is stored	0, $2^{-1022} \leq \lvert(s2)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions multiply the double-precision real number in the device specified by (s1) by the double-precision real number in the device specified by (s2), and store the multiplication result in the device specified by (d).



- Value 0 or $2^{-1022} \leq$ |specified value (stored value)| $< 2^{1024}$ can be specified or stored in the devices specified by (s1), (s2), and (d).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

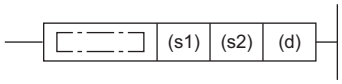
Operation error

Error code (SD0)	Description
3282H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{1024}$

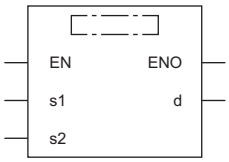
Dividing double-precision real numbers

ED/(P)

These instructions perform division between double-precision real numbers.

Ladder	ST
	ENO:=EDDIVISION(EN,s1,s2,d); ENO:=EDDIVISIONP(EN,s1,s2,d);

FBD/LD



(☐ is replaced by either of the following: EDDIVISION, EDDIVISIONP.)

■Execution condition

Instruction	Execution condition
ED/	
ED/P	

Setting data

■Description, range, data type

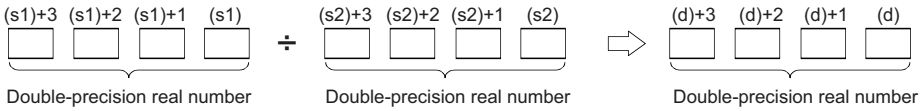
Operand	Description	Range	Data type	Data type (label)
(s1)	Dividend data or the start device where dividend data is stored	0, $2^{-1022} \leq \lvert(s1)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Divisor data or the start device where divisor data is stored	0, $2^{-1022} \leq \lvert(s2)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions divide the double-precision real number in the device specified by (s1) by the double-precision real number in the device specified by (s2), and store the division result in the device specified by (d).



- Value 0 or $2^{-1022} \leq$ |specified value (stored value)| $< 2^{1024}$ can be specified or stored in the devices specified by (s1), (s2), and (d).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

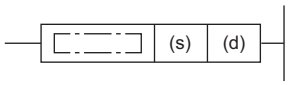
Operation error

Error code (SD0)	Description
3280H	The data (divisor) in the device specified by (s2) is 0.
3282H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{1024}$

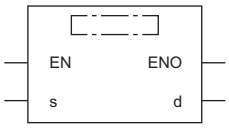
Converting 16-bit signed binary data to single-precision real number

INT2FLT(P)

These instructions convert 16-bit signed binary data to a single-precision real number.

Ladder	ST
	ENO:=INT2FLT(EN,s,d); ENO:=INT2FLTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
INT2FLT	
INT2FLTP	

Setting data

■Description, range, data type

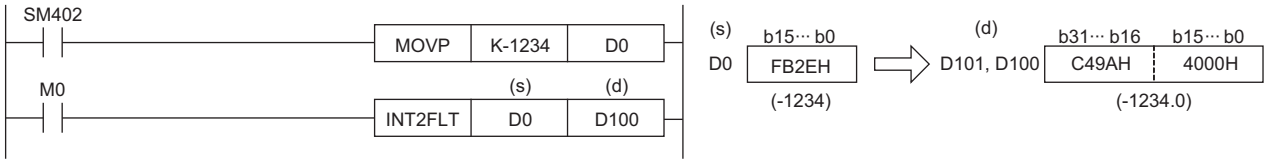
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the device containing integral data	-32768 to 32767	16-bit signed binary	ANY16_S
(d)	Start device for storing the converted single-precision real number	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 16-bit signed binary data in the device specified by (s) to a single-precision real number, and store the converted data in the device specified by (d).



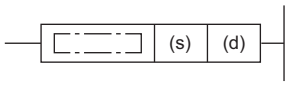
Operation error

There is no operation error.

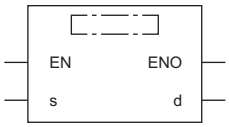
Converting 16-bit unsigned binary data to single-precision real number

UINT2FLT(P)

These instructions convert 16-bit unsigned binary data to a single-precision real number.

Ladder	ST
	ENO:=UINT2FLT(EN,s,d); ENO:=UINT2FLTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UINT2FLT	
UINT2FLTP	

Setting data

■Description, range, data type

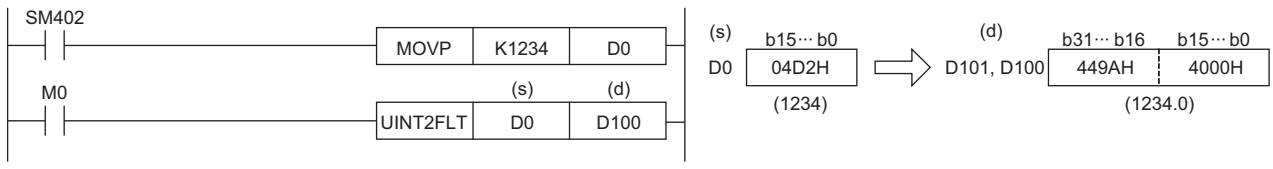
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the device containing integral data	0 to 65535	16-bit unsigned binary	ANY16_U
(d)	Start device for storing the converted single-precision real number	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 16-bit unsigned binary data in the device specified by (s) to a single-precision real number, and store the real number in the device specified by (d).



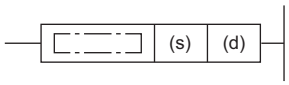
Operation error

There is no operation error.

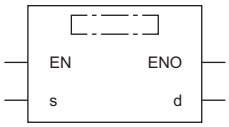
Converting 32-bit signed binary data to single-precision real number

DINT2FLT(P)

These instructions convert 32-bit signed binary data to a single-precision real number.

Ladder	ST
	ENO:=DINT2FLT(EN,s,d); ENO:=DINT2FLTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DINT2FLT	
DINT2FLTP	

Setting data

■Description, range, data type

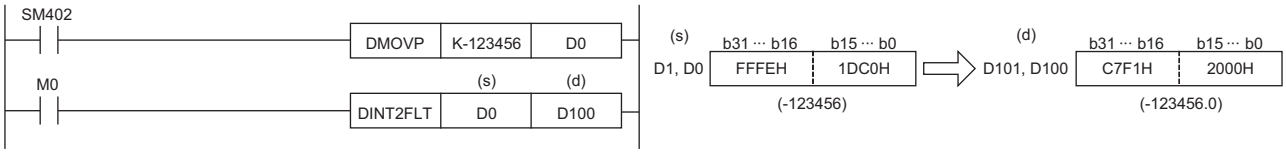
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the start device containing integral data	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
(d)	Start device for storing the converted single-precision real number	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

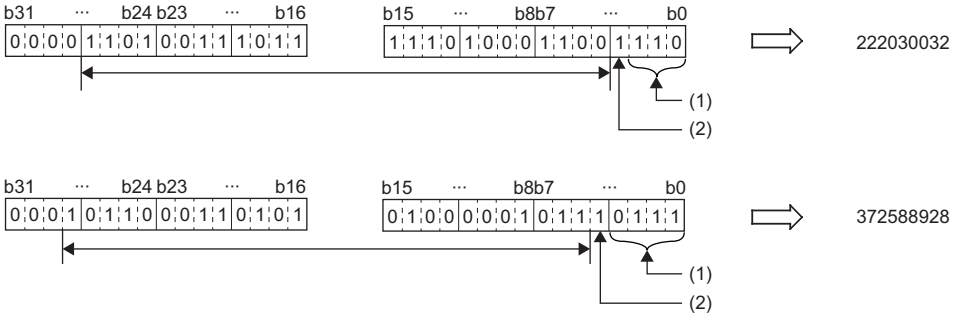
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 32-bit signed binary data in the device specified by (s) to a single-precision real number, and stores the real number in the device specified by (d).



- A single-precision real number is processed in 32-bit single precision, and therefore the effective number of digits is 24 bits when it is represented in binary and is about 7 digits when represented in decimal. For this reason, if the integer value exceeds the range from -16777216 to 16777215 (24-bit binary value), an error occurs in the converted value. The operation result is an integer value in which the 25th bit from upper bits is rounded off.



(1) Rounded down.
(2) Rounded off.

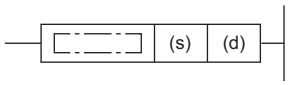
Operation error

There is no operation error.

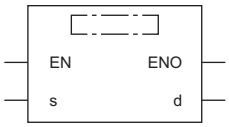
Converting 32-bit unsigned binary data to single-precision real number

UDINT2FLT(P)

These instructions convert 32-bit unsigned binary data to a single-precision real number.

Ladder	ST
	ENO:=UDINT2FLT(EN,s,d); ENO:=UDINT2FLTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UDINT2FLT	
UDINT2FLTP	

Setting data

■Description, range, data type

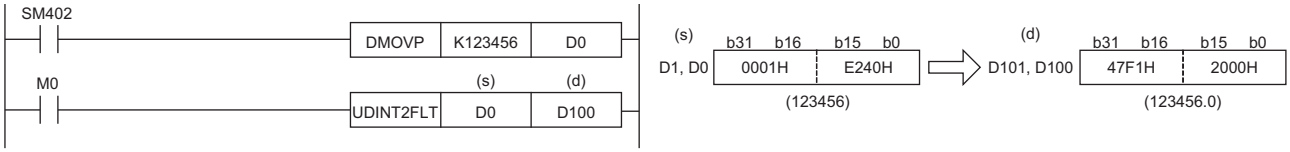
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the start device containing integral data	0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	Start device for storing the converted single-precision real number	—	Single-precision real number	ANYREAL_32

■Applicable devices

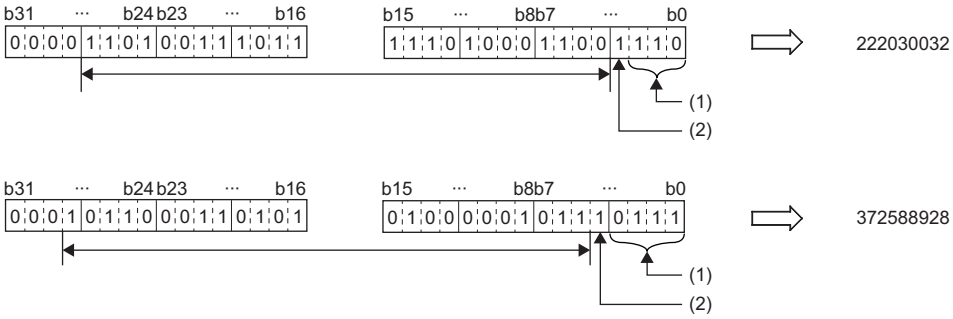
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the 32-bit unsigned binary data in the device specified by (s) to a single-precision real number, and stores the real number in the device specified by (d).



- A single-precision real number is processed in 32-bit single precision, and therefore the effective number of digits is 24 bits when it is represented in binary and is about 7 digits when represented in decimal. For this reason, if the integer value exceeds the range from 0 to 16777215 (24-bit binary value), an error occurs in the converted value. The operation result is an integer value in which the 25th bit from upper bits is rounded off.



(1) Rounded down.

(2) Rounded off.

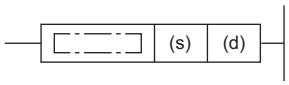
Operation error

There is no operation error.

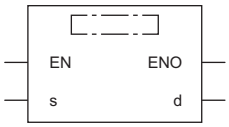
Converting double-precision real number to single-precision real number

DBL2FLT(P)

These instructions convert a double-precision real number to a single-precision real number.

Ladder	ST
	ENO:=DBL2FLT(EN,s,d); ENO:=DBL2FLTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBL2FLT	
DBL2FLTP	

Setting data

■Description, range, data type

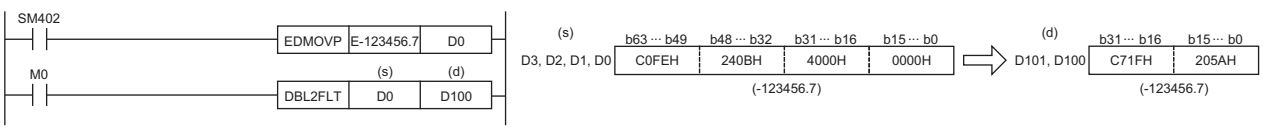
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the start device containing integral data	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the converted single-precision real number	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the double-precision real number in the device specified by (s) to a single-precision real number, and store the real number in the device specified by (d).



- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

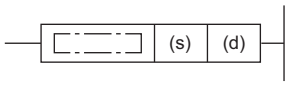
Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

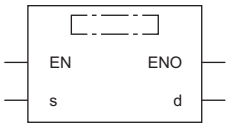
Converting 16-bit signed binary data to double-precision real number

INT2DBL(P)

These instructions convert 16-bit signed binary data to a double-precision real number.

Ladder	ST
	ENO:=INT2DBL(EN,s,d); ENO:=INT2DBLP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
INT2DBL	
INT2DBLP	

Setting data

■Description, range, data type

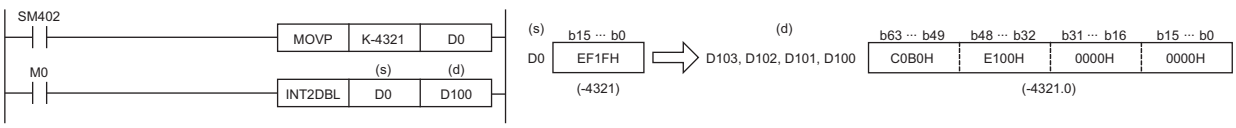
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a double-precision real number, or the device containing integral data	-32768 to 32767	16-bit signed binary	ANY16_S
(d)	Start device for storing the converted double-precision real number	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions convert the 16-bit signed binary data in the device specified by (s) to a double-precision real number, and store the real number in the device specified by (d).



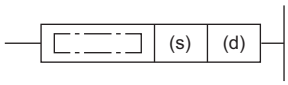
Operation error

There is no operation error.

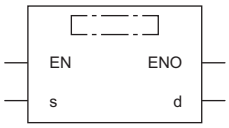
Converting 16-bit unsigned binary data to double-precision real number

UINT2DBL(P)

These instructions convert 16-bit unsigned binary data to a double-precision real number.

Ladder	ST
	ENO:=UINT2DBL(EN,s,d); ENO:=UINT2DBLP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UINT2DBL	
UINT2DBLP	

Setting data

■Description, range, data type

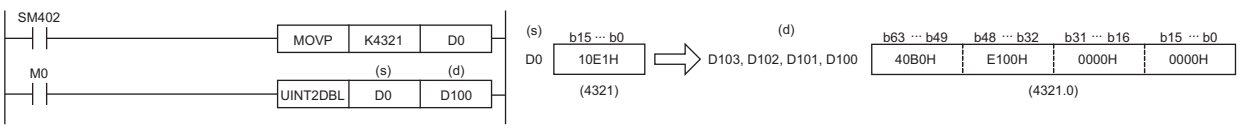
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a double-precision real number, or the device containing integral data	0 to 65535	16-bit unsigned binary	ANY16_U
(d)	Start device for storing the converted double-precision real number	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions convert the 16-bit unsigned binary data in the device specified by (s) to a double-precision real number, and store the real number in the device specified by (d).



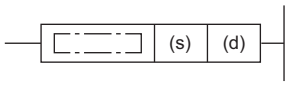
Operation error

There is no operation error.

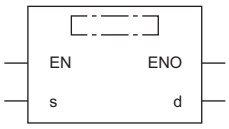
Converting 32-bit signed binary data to double-precision real number

DINT2DBL(P)

These instructions convert 32-bit signed binary data to a double-precision real number.

Ladder	ST
	ENO:=DINT2DBL(EN,s,d); ENO:=DINT2DBLP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DINT2DBL	
DINT2DBLP	

Setting data

■Description, range, data type

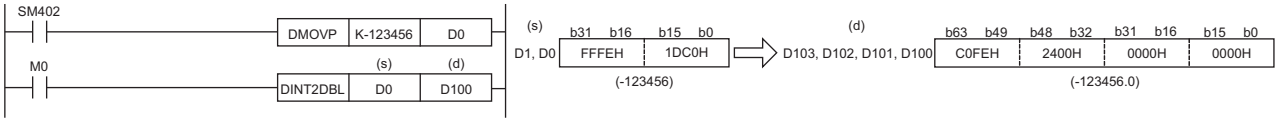
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a double-precision real number, or the start device containing integral data	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
(d)	Start device for storing the converted double-precision real number	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions convert the 32-bit signed binary data in the device specified by (s) to a double-precision real number, and store the real number in the device specified by (d).



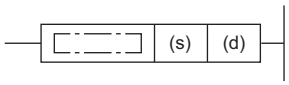
Operation error

There is no operation error.

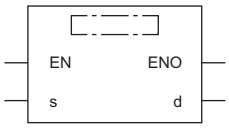
Converting 32-bit unsigned binary data to double-precision real number

UDINT2DBL(P)

These instructions convert 32-bit unsigned binary data to a double-precision real number.

Ladder	ST
	ENO:=UDINT2DBL(EN,s,d); ENO:=UDINT2DBLP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UDINT2DBL	
UDINT2DBLP	

Setting data

■Description, range, data type

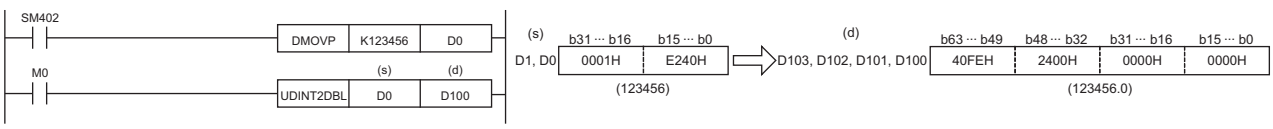
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a double-precision real number, or the start device containing integral data	0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	Start device for storing the converted double-precision real number	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions convert the 32-bit unsigned binary data in the device specified by (s) to a double-precision real number, and store the real number in the device specified by (d).



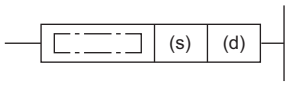
Operation error

There is no operation error.

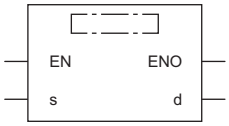
Converting single-precision real number to double-precision real number

FLT2DBL(P)

These instructions convert a single-precision real number to a double-precision real number.

Ladder	ST
	ENO:=FLT2DBL(EN,s,d); ENO:=FLT2DBLP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FLT2DBL	
FLT2DBLP	

Setting data

■Description, range, data type

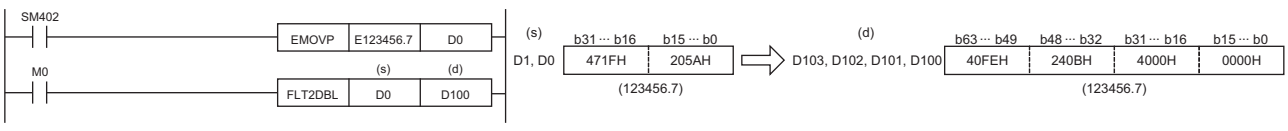
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the start device containing integral data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the converted double-precision real number	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	○	○	○	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

These instructions convert the single-precision real number in the device specified by (s) to a double-precision real number, and store the double-precision real number in the device specified by (d).



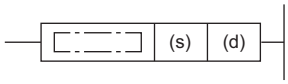
Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

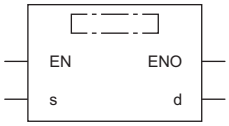
Converting string data to single-precision real number

EVAL(P)

These instructions convert a string to a single-precision real number.

Ladder	ST
	ENO:=EVAL(EN,s,d); ENO:=EVALP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EVAL	
EVALP	

Setting data

■Descriptions, ranges, and data types

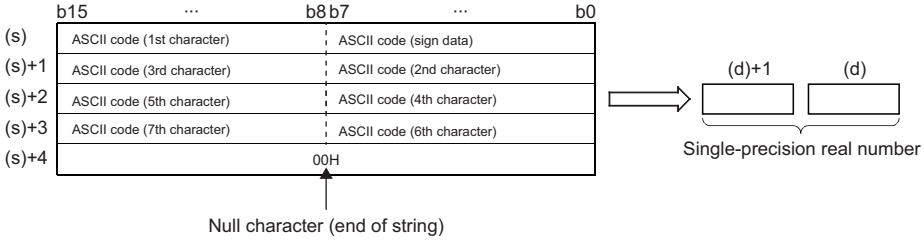
Operand	Description	Range	Data type	Data type (label)
(s)	String data to be converted into single-precision real number data, or the start device containing the string data	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the converted single-precision real number data	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

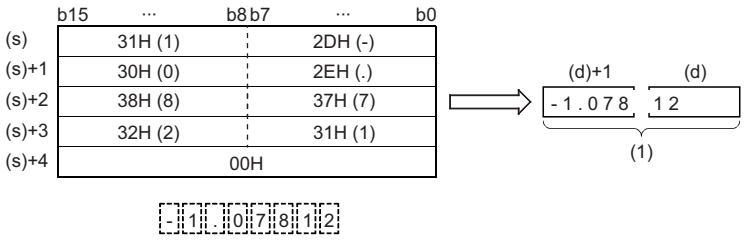
- These instructions convert the string stored in the device number specified by (s) and later to single-precision real number data, and store the converted real number in the device specified by (d).
- The specified string can be converted to a single-precision real number in either decimal point or exponential format.



- "20H" (space) that exists in the middle of the data is ignored.
- The string can consist of up to 24 characters. "20H" (space) and "30H" (0) are counted as one character as well.

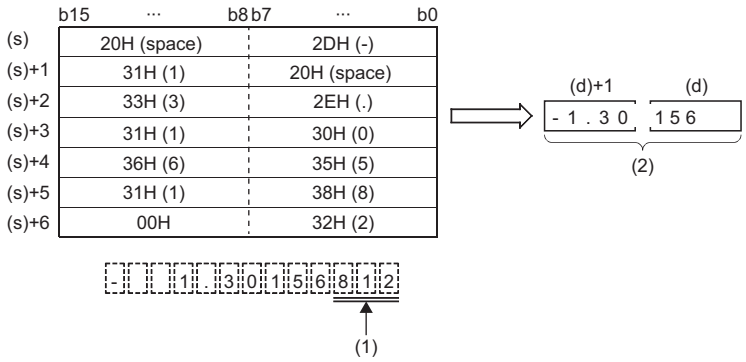
■Decimal point format

- When the string in the device specified by (s) is in decimal point format, conversion is performed as shown below.



(1) Single-precision real number

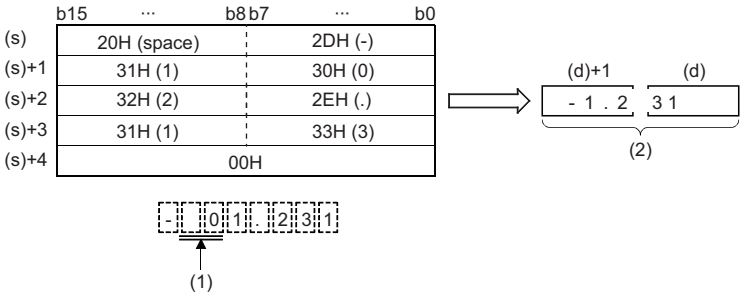
- The string in the device specified by (s) to be converted to a single-precision real number is converted by assuming that the six digits excluding the sign, decimal point, and exponent are effective and the seventh and subsequent digits are discarded.



(1) Rounded down.

(2) Single-precision real number

- If 2BH(+) is specified for the sign or the sign is omitted in decimal point format, the converted single-precision real number is treated as a positive value. If 2DH(-) is specified for the sign, it is treated as a negative value.
- If 20H (space) or 30H (0) exists between digits excluding the first 0 in the string in the device specified by (s), the instruction performs conversion by ignoring 20H and 30H.

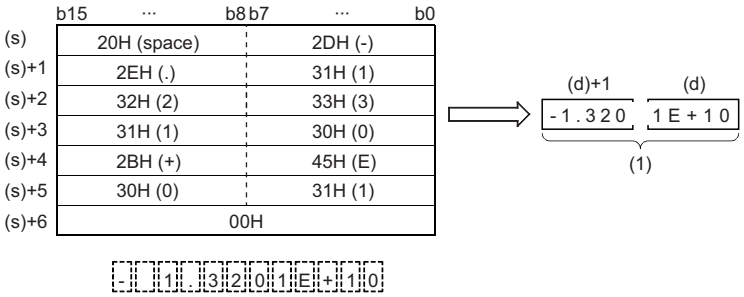


(1) Ignored.

(2) Single-precision real number

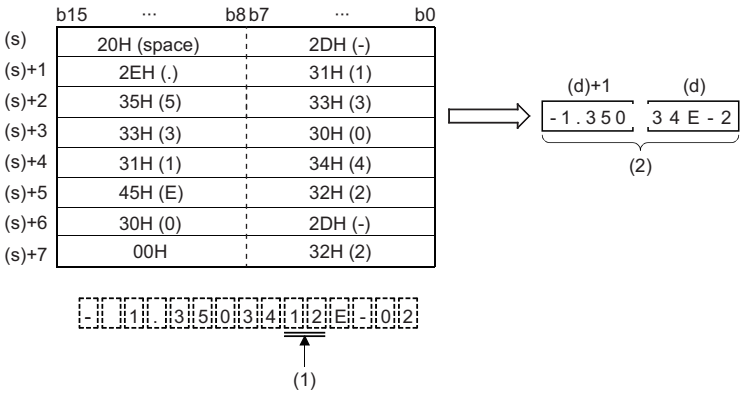
■Exponential format

- When the string in the device specified by (s) is in exponential format, conversion is performed as shown below.



(1) Single-precision real number

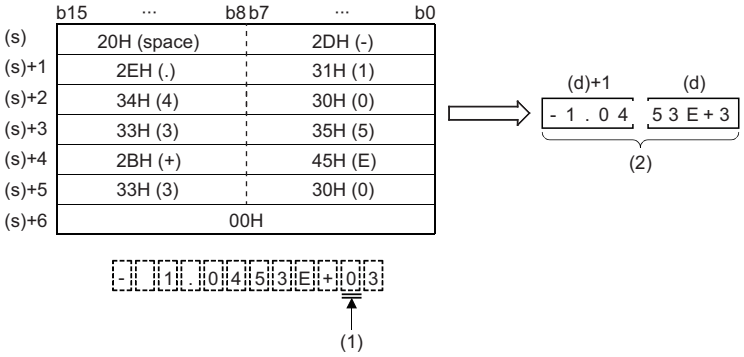
- The string in the device specified by (s) to be converted to a single-precision real number is converted by assuming that the six digits excluding the sign, decimal point, and exponent are effective and the seventh and subsequent digits are discarded.



(1) Rounded down.

(2) Single-precision real number

- If 2BH(+) is specified for the sign or the sign is omitted in exponential format, the converted single-precision real number is treated as a positive value. If 2DH(-) is specified for the sign in the exponent, it is treated as a negative value.
- If 20H (space) or 30H (0) exists between digits excluding the first 0 in the string in the device specified by (s), the instruction performs conversion by ignoring 20H and 30H.
- If 30H (0) exists between "E" and a numerical value in the string in exponential format, the instruction performs conversion by ignoring 30H.



(1) Ignored.

(2) Single-precision real number

- "20H" (space) that exists in the middle of the data is ignored.
- The string can consist of up to 24 characters. "20H" (space) and "30H" (0) in a string are counted as one character as well.

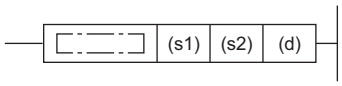
Operation error

Error code (SD0)	Description
2820H	00H does not exist within the range of the relevant device specified by (s).
3281H	Invalid data that cannot be converted are set to (s). • The integral part or decimal part contains a character other than 30H (0) to 39H (9). • More than one 2EH (.) exists in the specified string. • The exponent of the specified string contains a character other than 45H (E), 65H (e), 2BH (+), and 2DH (-). • The specified string contains more than one exponent 45H (E) or 65H (e). • The exponent in the specified string contains a numerical value consisting of three digits or more. • The exponent of the specified string contains more than one sign 2BH (+) or 2DH (-). • The specified string contains more than one sign 2BH (+) or 2DH (-) in the integral part of decimal point format or in the mantissa of exponential format.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$
3285H	The number of characters in the device specified by (s) and later is 0 or exceeds 24.

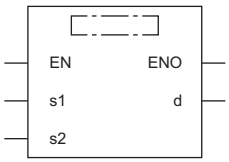
Converting BCD format data to single-precision real number

EREXP(P)

These instructions convert BCD floating point format data to single-precision real number data in accordance with the specified number of digits in the decimal part.

Ladder	ST
	ENO:=EREXP(EN,s1,s2,d); ENO:=EREXPP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EREXP	
EREXPP	

Setting data

■Descriptions, ranges, and data types

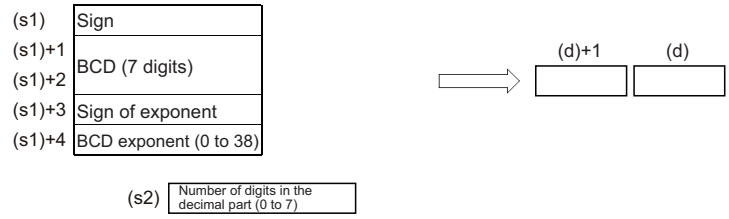
Operand	Description	Range	Data type	Data type (label)
(s1)	Start device containing BCD floating point format data	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
(s2)	Number of digits in the decimal part	0 to 7	16-bit signed binary	ANY16
(d)	Start device for storing a single-precision real number	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

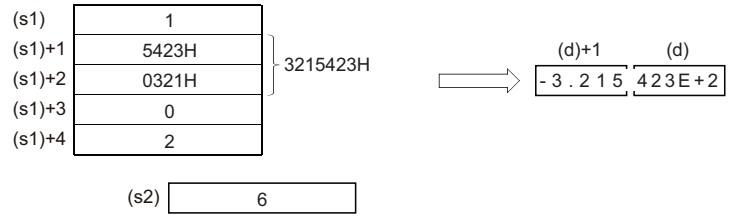
- These instructions convert the BCD floating-point format data stored in the device specified by (s1) and later to single-precision real number data in accordance with the number of decimal places stored in the device specified by (s2), and store the converted data in the device number specified by (d) and later.



(s1): Sign (Positive: 0, Negative: 1)
(s1)+1, (s1)+2: BCD (7 digits)
(s1)+3: Sign of exponent (Positive: 0, Negative: 1)
(s1)+4: BCD exponent (0 to 38)
(s2): Number of digits in the decimal part (0 to 7)
(d)+1, (d): Single-precision real number

- For the sign in (s1) and exponent sign in (s1)+3, 0 is set for positive and 1 is set for negative.
- A value of 0 to 38 can be set for the BCD exponent in (s1)+4.
- A value of 0 to 7 can be set for the number of decimal part digits in the device specified by (s2).

Ex.
When 6 is specified in (s2)



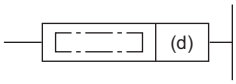
Operation error

Error code (SD0)	Description
3281H	The value set to (s1) as sign data is not 0 or 1.
	A value other than 0 to 9 exists at any digit of data set to (s1)+1 and (s1)+2.
	The value set to (s1)+3 as sign data of exponent is not 0 or 1.
	The exponent data set to (s1)+4 is out of the range, 0 to 38.
	The value set to (s2) as the number of digits in the decimal part is out of the range, 0 to 7.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

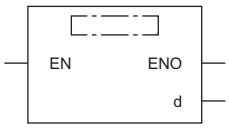
Inverting the sign of single-precision real number

ENEG(P)

These instructions invert the sign of single-precision real number data.

Ladder	ST
	ENO:=ENEG(EN,d); ENO:=ENEGP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ENEG	
ENEGP	

Setting data

■Description, range, data type

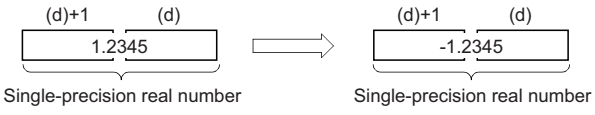
Operand	Description	Range	Data type	Data type (label)
(d)	Start device containing the single-precision real number data subject to sign inversion	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions invert the sign of the single-precision real number in the device specified by (d) and store the inverted data in the device specified by (d).



- The instructions are used to invert positive and negative signs.

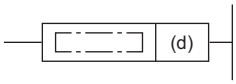
Operation error

Error code (SD0)	Description
3282H	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.

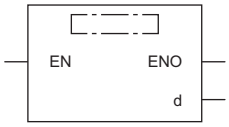
Inverting the sign of double-precision real number

EDNEG(P)

These instructions invert the sign of double-precision real number data.

Ladder	ST
	ENO:=EDNEG(EN,d); ENO:=EDNEGP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EDNEG	
EDNEGP	

Setting data

■Description, range, data type

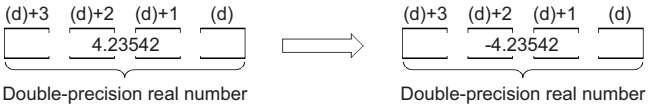
Operand	Description	Range	Data type	Data type (label)
(d)	Start device containing the double-precision real number subject to sign inversion	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions invert the sign of the double-precision real number data in the device specified by (d) and store the inverted data in the device specified by (d).



- The instructions are used to invert positive and negative signs.

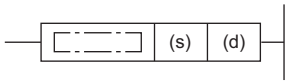
Operation error

Error code (SD0)	Description
3282H	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.

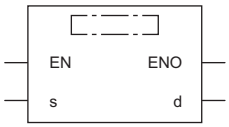
Transferring single-precision real number

EMOV(P)

These instructions transfer single-precision real number data to the specified device.

Ladder	ST
	ENO:=EMOV(EN,s,d); ENO:=EMOVP(EN,s,d)

FBD/LD



■Execution condition

Instruction	Execution condition
EMOV	
EMOVP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Data to be transferred or start device containing the data to be transferred	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing transferred data	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions transfer the single-precision real number data stored in the device specified by (s) to the device specified by (d).



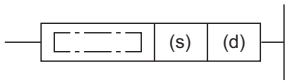
Operation error

There is no operation error.

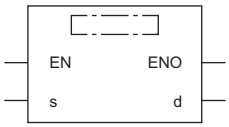
Transferring double-precision real number

EDMOV(P)

These instructions transfer double-precision real number data to the specified device.

Ladder	ST
	ENO:=EDMOV(EN,s,d); ENO:=EDMOVP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EDMOV	
EDMOVP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Data to be transferred or start device containing the data to be transferred	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing transferred data	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions transfer the double-precision real number data stored in the device specified by (s) to the device specified by (d).



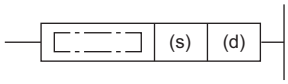
Operation error

There is no operation error.

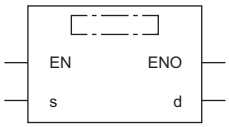
Calculating the sine of single-precision real number

SIN(P)

These instructions calculate the sine of the angle specified by a single-precision real number.

Ladder	ST*1
	ENO:=SINP(EN,s,d);

FBD/LD*1



*1 The SIN instruction does not support the ST and FBD/LD. Use the standard function, SIN.
☞ Page 1281 SIN(_E)

■Execution condition

Instruction	Execution condition
SIN	
SINP	

Setting data

■Description, range, data type

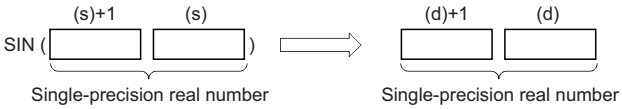
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for sine calculation, or the start device containing the angle data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the sine of the angle specified by (s), and store the operation result in the device specified by (d).



- Set the angle data in radians (angle×π÷180).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

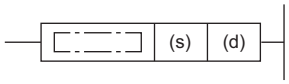
For the angle↔radian conversion, refer to the RAD(P) and DEG(P) instructions.

☞ Page 918 RAD(P)
☞ Page 920 DEG(P)

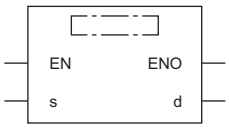
Calculating the cosine of single-precision real number

COS(P)

These instructions calculate the cosine of the angle specified by a single-precision real number.

Ladder	ST*1
	ENO:=COSP(EN,s,d);

FBD/LD*1



*1 The COS instruction does not support the ST and FBD/LD. Use the standard function, COS.
☞ Page 1282 COS(_E)

■Execution condition

Instruction	Execution condition
COS	
COSP	

Setting data

■Description, range, data type

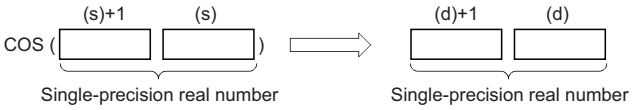
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for cosine calculation, or the start device containing the angle data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the cosine of the angle specified by (s), and store the operation result in the device specified by (d).



- Set the angle data in radians (angle×π÷180).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

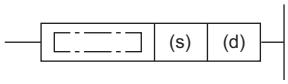
For the angle↔radian conversion, refer to the RAD(P) and DEG(P) instructions.

☞ Page 918 RAD(P)
☞ Page 920 DEG(P)

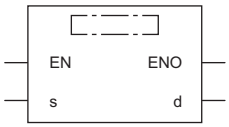
Calculating the tangent of single-precision real number

TAN(P)

These instructions calculate the tangent of the angle specified by a single-precision real number.

Ladder	ST*1
	ENO:=TANP(EN,s,d);

FBD/LD*1



*1 The TAN instruction does not support the ST and FBD/LD. Use the standard function, TAN.
☞ Page 1283 TAN(_E)

■Execution condition

Instruction	Execution condition
TAN	
TANP	

Setting data

■Description, range, data type

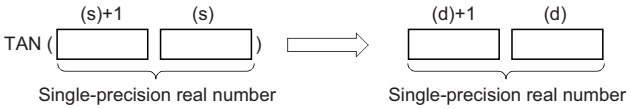
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for tangent calculation, or the start device containing the angle data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the tangent of the angle specified by (s), and store the operation result in the device specified by (d).



- Set the angle data in radians (angle×π÷180).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Precautions

If the angle specified by (s) is π/2 radian or (3/2)π radian, no operation error will be issued because of the truncation error in the radian value.

Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

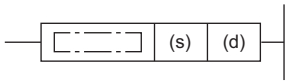
For the angle↔radian conversion, refer to the RAD(P) and DEG(P) instructions.

☞ Page 918 RAD(P)
☞ Page 920 DEG(P)

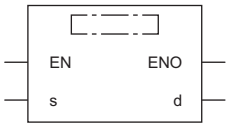
Calculating the arc sine of single-precision real number

ASIN(P)

These instructions calculate the angle from the sine specified by a single-precision real number.

Ladder	ST*1
	ENO:=ASINP(EN,s,d);

FBD/LD*1



*1 The ASIN instruction does not support the ST and FBD/LD. Use the standard function, ASIN.
☞ Page 1284 ASIN(_E)

■Execution condition

Instruction	Execution condition
ASIN	
ASINP	

Setting data

■Description, range, data type

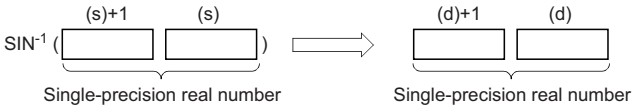
Operand	Description	Range	Data type	Data type (label)
(s)	Sine data used for arc sine calculation, or the start device containing the sine data	-1.0 to 1.0	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the angle based on the sine data in the device specified by (s), and store the operation result in the device number specified by (d).



- The sine data in the device specified by (s) can be set in the range from -1.0 to 1.0.
- The angle (operation result) is stored in radians in the device specified by (d).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3285H	The value in the device specified by (s) is out of the range, -1.0 to 1.0.

Point

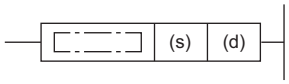
For the angle↔radian conversion, refer to the RAD(P) and DEG(P) instructions.

☞ Page 918 RAD(P)
☞ Page 920 DEG(P)

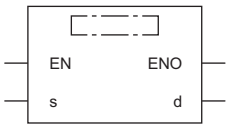
Calculating the arc cosine of single-precision real number

ACOS(P)

These instructions calculate the angle from the cosine specified by a single-precision real number.

Ladder	ST*1
	ENO:=ACOSP(EN,s,d)

FBD/LD*1



*1 The ACOS instruction does not support the ST and FBD/LD. Use the standard function, ACOS.
☞ Page 1285 ACOS(_E)

■Execution condition

Instruction	Execution condition
ACOS	
ACOSP	

Setting data

■Description, range, data type

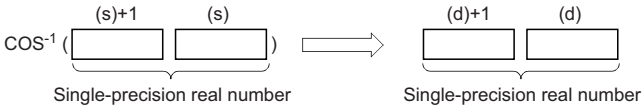
Operand	Description	Range	Data type	Data type (label)
(s)	Cosine data used for arc cosine calculation, or the start device containing the cosine data	-1.0 to 1.0	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the angle based on the cosine data in the device specified by (s), and store the operation result in the device number specified by (d).



- The cosine data in the device specified by (s) can be set in the range from -1.0 to 1.0.
- The angle (operation result) is stored in radians in the device specified by (d).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3285H	The value in the device specified by (s) is out of the range, -1.0 to 1.0.

Point

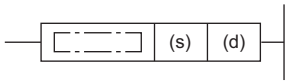
For the angle↔radian conversion, refer to the RAD(P) and DEG(P) instructions.

☞ Page 918 RAD(P)
☞ Page 920 DEG(P)

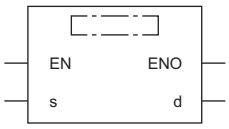
Calculating the arc tangent of single-precision real number

ATAN(P)

These instructions calculate the angle from the tangent specified by a single-precision real number.

Ladder	ST*1
	ENO:=ATANP(EN,s,d);

FBD/LD*1



*1 The ATAN instruction does not support the ST and FBD/LD. Use the standard function, ATAN.
☞ Page 1286 ATAN(_E)

■Execution condition

Instruction	Execution condition
ATAN	
ATANP	

Setting data

■Description, range, data type

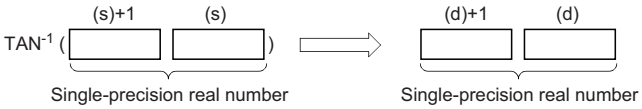
Operand	Description	Range	Data type	Data type (label)
(s)	Tangent data used for arc tangent calculation, or the start device containing the tangent data	$0, 2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the angle based on the tangent data in the device specified by (s), and store the operation result in the device number specified by (d).



- The angle (operation result) is stored in radians in the device specified by (d).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

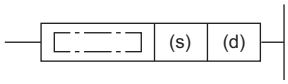
For the angle↔radian conversion, refer to the RAD(P) and DEG(P) instructions.

☞ Page 918 RAD(P)
☞ Page 920 DEG(P)

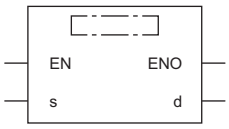
Calculating the sine of double-precision real number

SIND(P)

These instructions calculate the sine of the angle specified by a double-precision real number.

Ladder	ST*1
	ENO:=SINDP(EN,s,d);

FBD/LD*1



*1 The SIND instruction does not support the ST and FBD/LD. Use the standard function, SIN.
☞ Page 1281 SIN(_E)

■Execution condition

Instruction	Execution condition
SIND	
SINDP	

Setting data

■Description, range, data type

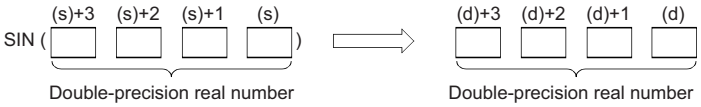
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for sine calculation, or the start device containing the angle data	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the sine of the angle specified by (s), and store the operation result in the device specified by (d).



- Set the angle data in radians (angle×π÷180).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

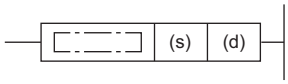
For the angle↔radian conversion, refer to the RADD(P) and DEGD(P) instructions.

☞ Page 922 RADD(P)
☞ Page 924 DEGD(P)

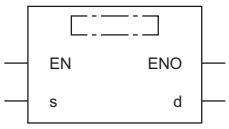
Calculating the cosine of double-precision real number

COSD(P)

These instructions calculate the cosine of the angle specified by a double-precision real number.

Ladder	ST*1
	ENO:=COSDP(EN,s,d);

FBD/LD*1



*1 The COSD instruction does not support the ST and FBD/LD. Use the standard function, COS.
☞ Page 1282 COS(_E)

■Execution condition

Instruction	Execution condition
COSD	
COSDP	

Setting data

■Descriptions, ranges, and data types

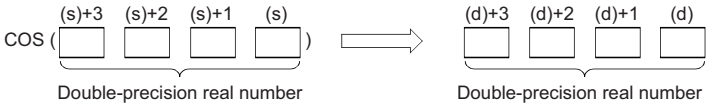
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for cosine calculation, or the start device containing the angle data	0, $2^{-1022} \leq |(s)| < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the cosine of the angle specified by (s), and store the operation result in the device specified by (d).



- Set the angle data in radians (angle×π÷180).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

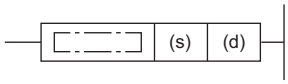
For the angle↔radian conversion, refer to the RADD(P) and DEGD(P) instructions.

☞ Page 922 RADD(P)
☞ Page 924 DEGD(P)

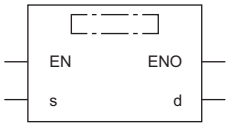
Calculating the tangent of double-precision real number

TAND(P)

These instructions calculate the tangent of the angle specified by a double-precision real number.

Ladder	ST*1
	ENO:=TANDP(EN,s,d);

FBD/LD*1



*1 The TAND instruction does not support the ST and FBD/LD. Use the standard function, TAN.
☞ Page 1283 TAN(_E)

■Execution condition

Instruction	Execution condition
TAND	
TANDP	

Setting data

■Description, range, data type

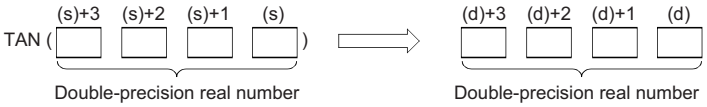
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for tangent calculation, or the start device containing the angle data	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the tangent of the angle specified by (s), and store the operation result in the device specified by (d).



- Set the angle data in radians (angle×π÷180).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Precautions

If the angle specified by (s) is π/2 radian or (3/2)π radian, no operation error will be issued because of the truncation error in the radian value.

Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

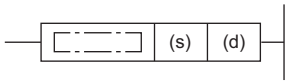
For the angle↔radian conversion, refer to the RADD(P) and DEGD(P) instructions.

☞ Page 922 RADD(P)
☞ Page 924 DEGD(P)

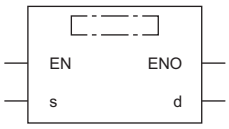
Calculating the arc sine of double-precision real number

ASIND(P)

These instructions calculate the angle from the sine specified by a double-precision real number.

Ladder	ST*1
	ENO:=ASINDP(EN,s,d)

FBD/LD*1



*1 The ASIND instruction does not support the ST and FBD/LD. Use the standard function, ASIN.
☞ Page 1284 ASIN(_E)

■Execution condition

Instruction	Execution condition
ASIND	
ASINDP	

Setting data

■Description, range, data type

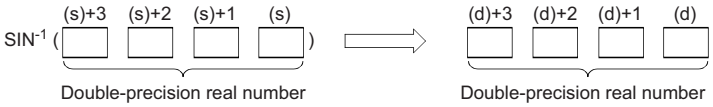
Operand	Description	Range	Data type	Data type (label)
(s)	Sine data used for arc sine calculation, or the start device containing the sine data	-1.0 to 1.0	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the angle based on the sine data in the device specified by (s), and store the operation result in the device number specified by (d).



- The sine data in the device specified by (s) can be set in the range from -1.0 to 1.0.
- The angle (operation result) is stored in radians in the device specified by (d).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3285H	The value in the device specified by (s) is out of the range, -1.0 to 1.0.

Point

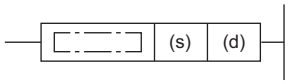
For the angle↔radian conversion, refer to the RADD(P) and DEGD(P) instructions.

☞ Page 922 RADD(P)
☞ Page 924 DEGD(P)

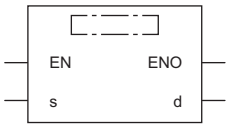
Calculating the arc cosine of double-precision real number

ACOSD(P)

These instructions calculate the angle from the cosine specified by a double-precision real number.

Ladder	ST*1
	ENO:=ACOSDP(EN,s,d);

FBD/LD*1



*1 The ACOSD instruction does not support the ST and FBD/LD. Use the standard function, ACOS.
☞ Page 1285 ACOS(_E)

■Execution condition

Instruction	Execution condition
ACOSD	
ACOSDP	

Setting data

■Description, range, data type

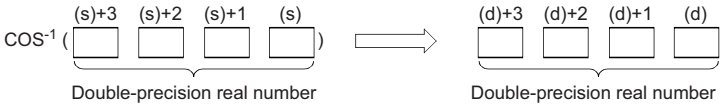
Operand	Description	Range	Data type	Data type (label)
(s)	Cosine data used for arc cosine calculation, or the start device containing the cosine data	-1.0 to 1.0	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the angle based on the cosine data in the device specified by (s), and store the operation result in the device number specified by (d).



- The cosine data in the device specified by (s) can be set in the range from -1.0 to 1.0.
- The angle (operation result) is stored in radians in the device specified by (d).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3285H	The value in the device specified by (s) is out of the range, -1.0 to 1.0.

Point

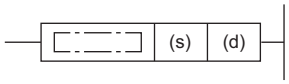
For the angle↔radian conversion, refer to the RADD(P) and DEGD(P) instructions.

☞ Page 922 RADD(P)
☞ Page 924 DEGD(P)

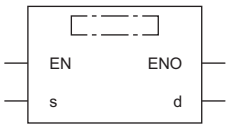
Calculating the arc tangent of double-precision real number

ATAND(P)

These instructions calculate the angle from the tangent specified by a double-precision real number.

Ladder	ST*1
	ENO:=ATANDP(EN,s,d);

FBD/LD*1



*1 The ATAND instruction does not support the ST and FBD/LD. Use the standard function, ATAN.
☞ Page 1286 ATAN(_E)

■Execution condition

Instruction	Execution condition
ATAND	
ATANDP	

Setting data

■Description, range, data type

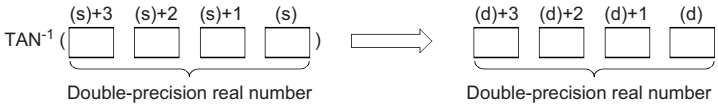
Operand	Description	Range	Data type	Data type (label)
(s)	Tangent data used for arc tangent calculation, or the start device containing the tangent data	0, $2^{-1022} \leq \|(s)\| < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the angle based on the tangent data in the device specified by (s), and store the operation result in the device number specified by (d).



- The angle (operation result) is stored in radians in the device specified by (d).
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Point

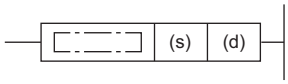
For the angle↔radian conversion, refer to the RADD(P) and DEGD(P) instructions.

☞ Page 922 RADD(P)
☞ Page 924 DEGD(P)

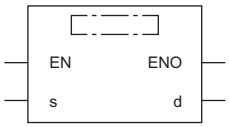
Calculating the sine of BCD data

BSIN(P)

These instructions calculate the sine of the angle specified by a BCD value.

Ladder	ST
	ENO:=BSIN(EN,s,d); ENO:=BSINP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BSIN	
BSINP	

Setting data

■Descriptions, ranges, and data types

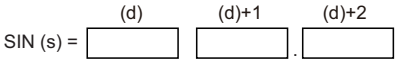
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for sine calculation, or the device containing the data	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the operation result	—	BCD 4-digit	ANY16_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions calculate the sine of the angle specified by (s), and store the sign of the operation result in the device specified by (d) and the operation result in the devices specified by (d)+1 and (d)+2.



(d): Sign
(d)+1: Integral part
(d)+2: Decimal part

- For the value to be specified in (s), set a value from 0 to 360° (in the DEG. unit) as a BCD value.
- For the sign of the operation result to be stored in the device specified by (d), 0 is stored when the operation result is positive and 1 is stored when the operation result is negative.
- The operation result to be stored in the devices specified by (d)+1 and (d)+2 is a BCD value in the range from -1.000 to 1.000.
- The operation result is a value whose 5th decimal place is rounded off.

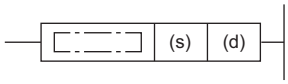
Operation error

Error code (SD0)	Description
3285H	An out-of-range value is set to (s). • The specified data is not a BCD value. • The specified data is out of the range, 0 to 360.

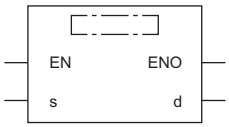
Calculating the cosine of BCD data

BCOS(P)

These instructions calculate the cosine of the angle specified by a BCD value.

Ladder	ST
	ENO:=BCOS(EN,s,d); ENO:=BCOSP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BCOS	
BCOSP	

Setting data

■Descriptions, ranges, and data types

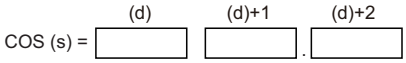
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for cosine calculation, or the device containing the data	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the operation result	—	BCD 4-digit	ANY16_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions calculate the cosine of the angle specified by (s), and store the sign of the operation result in the word device specified by (d) and the operation result in the word devices specified by (d)+1 and (d)+2.



(d): Sign
(d)+1: Integral part
(d)+2: Decimal part

- For the value to be specified in (s), set a value from 0 to 360° (in the DEG. unit) as a BCD value.
- For the sign of the operation result to be stored in the device specified by (d), 0 is stored when the operation result is positive and 1 is stored when the operation result is negative.
- The operation result to be stored in the devices specified by (d)+1 and (d)+2 is a BCD value in the range from -1.000 to 1.000.
- The operation result is a value whose 5th decimal place is rounded off.

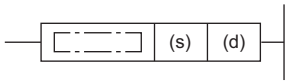
Operation error

Error code (SD0)	Description
3285H	An out-of-range value is set to (s). • The specified data is not a BCD value. • The specified data is out of the range, 0 to 360.

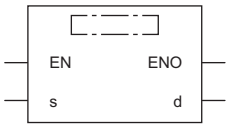
Calculating the tangent of BCD data

BTAN(P)

These instructions calculate the tangent of the angle specified by a BCD value.

Ladder	ST
	ENO:=BTAN(EN,s,d); ENO:=BTANP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BTAN	
BTANP	

Setting data

■Descriptions, ranges, and data types

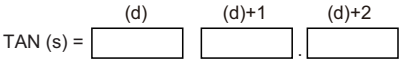
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for tangent calculation, or the device containing the data	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the operation result	—	BCD 4-digit	ANY16_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions calculate the tangent of the angle specified by (s), and store the sign of the operation result in the device specified by (d) and the operation result in the devices specified by (d)+1 and (d)+2.



(d): Sign
(d)+1: Integral part
(d)+2: Decimal part

- For the value to be specified in (s), set a value from 0 to 360° (in the DEG. unit) as a BCD value.
- For the sign of the operation result to be stored in the device specified by (d), 0 is stored when the operation result is positive and 1 is stored when the operation result is negative.
- The operation result to be stored in the devices specified by (d)+1 and (d)+2 is a BCD value in the range from -57.2901 to 57.2903.
- The operation result is a value whose 5th decimal place is rounded off.

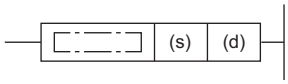
Operation error

Error code (SD0)	Description
3285H	An out-of-range value is set to (s). • The specified data is not a BCD value. • The specified data is out of the range, 0 to 360. • The specified data is 90° or 270°.

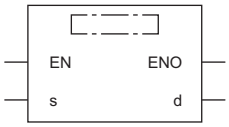
Calculating the arc sine of BCD data

BASIN(P)

These instructions calculate the arc sine of the angle specified by a BCD value.

Ladder	ST
	ENO:=BASIN(EN,s,d); ENO:=BASINP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BASIN	
BASINP	

Setting data

■Descriptions, ranges, and data types

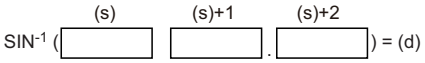
Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the data used for arc sine calculation	0 to 9999	BCD 4-digit	ANY16_ARRAY (Number of elements: 3)
(d)	Device for storing the operation result	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions calculate the arc sine of the value specified by (s), and store the operation result (angle) in the device specified by (d).



(s): Sign
(s)+1: Integral part
(s)+2: Decimal part

- Set the sign of the calculation data in the device specified by (s). Store 0 when the calculation data is positive, or store 1 when the calculation data is negative.
- Store a BCD value for the integral part of calculation data in (s)+1 and a BCD value for the decimal part in (s)+2. (A value from 0 to 1.0000 can be set.)
- The operation result to be stored in the device specified by (d) is a BCD value in the range from 0 to 90° or 270 to 360° (in DEG. unit).
- The operation result is a value whose decimal part is rounded off.

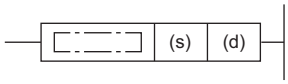
Operation error

Error code (SD0)	Description
3285H	An out-of-range value is set to (s). • The specified data is not a BCD value. • The specified data is out of the range, -1.0000 to 1.0000.

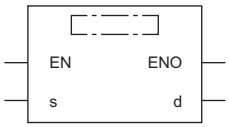
Calculating the arc cosine of BCD data

BACOS(P)

These instructions calculate the arc cosine of the angle specified by a BCD value.

Ladder	ST
	ENO:=BACOS(EN,s,d); ENO:=BACOSP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BACOS	
BACOSP	

Setting data

■Descriptions, ranges, and data types

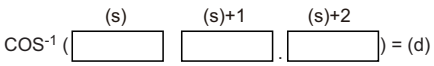
Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the data used for arc cosine calculation	0 to 9999	BCD 4-digit	ANY16_ARRAY (Number of elements: 3)
(d)	Device for storing the operation result	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions calculate the arc cosine of the value specified by (s), and store the operation result (angle) in the device specified by (d).



(s): Sign
(s)+1: Integral part
(s)+2: Decimal part

- Set the sign of the calculation data in the device specified by (s). Store 0 when the calculation data is positive, or store 1 when the calculation data is negative.
- Store a BCD value for the integral part of calculation data in (s)+1 and a BCD value for the decimal part in (s)+2. (A value from 0 to 1.0000 can be set.)
- The operation result to be stored in the device specified by (d) is a BCD value in the range from 0 to 180° (in DEG. unit).
- The operation result is a value whose decimal part is rounded off.

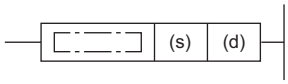
Operation error

Error code (SD0)	Description
3285H	An out-of-range value is set to (s). • The specified data is not a BCD value. • The specified data is out of the range, -1.0000 to 1.0000.

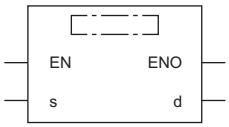
Calculating the arc tangent of BCD data

BATAN(P)

These instructions calculate the arc tangent of the angle specified by a BCD value.

Ladder	ST
	ENO:=BATAN(EN,s,d); ENO:=BATANP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BATAN	
BATANP	

Setting data

■Descriptions, ranges, and data types

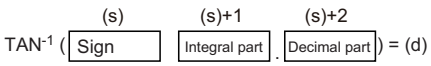
Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the data used for arc tangent calculation	0 to 9999	BCD 4-digit	ANY16_ARRAY (Number of elements: 3)
(d)	Device for storing the operation result	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions calculate the arc tangent of the value specified by (s), and store the operation result (angle) in the device specified by (d).



(s): Sign
(s)+1: Integral part
(s)+2: Decimal part

- Set the sign of the calculation data in the device specified by (s). Store 0 when the calculation data is positive, or store 1 when the calculation data is negative.
- Store a BCD value for the integral part of calculation data in (s)+1 and a BCD value for the decimal part in (s)+2. (A value from 0 to 9999.9999 can be set.)
- The operation result to be stored in the device specified by (d) is a BCD value in the range from 0 to 90° or 270 to 360° (in DEG. unit).
- The operation result is a value whose decimal part is rounded off.

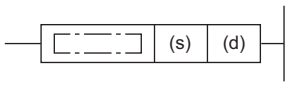
Operation error

Error code (SD0)	Description
3285H	The data in the device specified by (s) is not a BCD value.

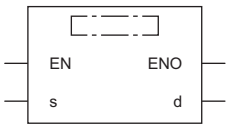
Converting single-precision real number angle to radian

RAD(P)

These instructions convert the unit of the measure of angle from the degree specified by a single-precision real number to radian.

Ladder	ST
	ENO:=RAD(EN,s,d); ENO:=RADP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
RAD	
RADP	

Setting data

■Description, range, data type

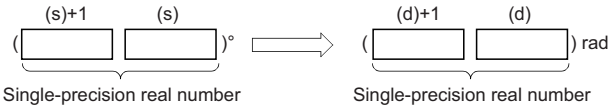
Operand	Description	Range	Data type	Data type (label)
(s)	Angle for which the unit is to be changed to radian, or the start device containing the angle	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the angle in radians	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the unit of the measure of angle from the degree specified by (s) to the radian, and store the angle in radians in the device number specified by (d).



- Unit conversion from the degree to the radian is performed as follows.

$$\text{Radian} = \text{Degree} \times \frac{\pi}{180}$$

- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

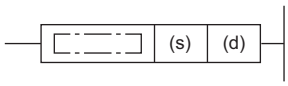
Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

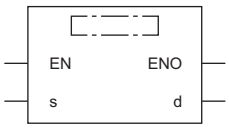
Converting single-precision real number radian to angle

DEG(P)

These instructions convert the unit of the measure of angle from the radian specified by a single-precision real number to the degree.

Ladder	ST
	ENO:=DEG(EN,s,d); ENO:=DEGP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DEG	
DEGP	

Setting data

■Description, range, data type

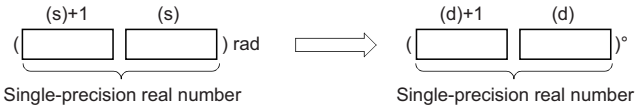
Operand	Description	Range	Data type	Data type (label)
(s)	Angle in radians for which the unit is to be changed to the degree, or the start device containing the angle in radians	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the value converted in degrees	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions convert the unit of the measure of angle from the radian specified by (s) to the degree, and store the angle in degrees in the device number specified by (d).



- Unit conversion from the radian to the degree is performed as follows.

$$\text{Degree} = \text{Radian} \times \frac{180}{\pi}$$

- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

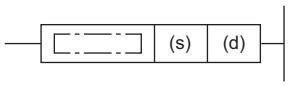
Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

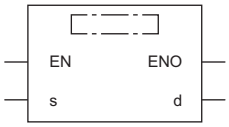
Converting double-precision real number angle to radian

RADD(P)

These instructions convert the unit of the measure of angle from the degree specified by a single-precision real number to radian.

Ladder	ST
	ENO:=RADD(EN,s,d); ENO:=RADDP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
RADD	
RADDP	

Setting data

■Description, range, data type

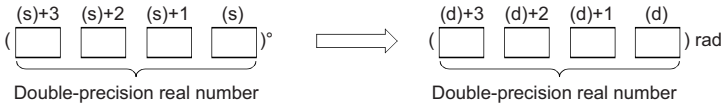
Operand	Description	Range	Data type	Data type (label)
(s)	Angle for which the unit is to be changed to radian, or the start device containing the angle	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the angle in radians	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions convert the unit of the measure of angle from the degree specified by (s) to the radian, and store the angle in radians in the device number specified by (d).



- Unit conversion from the degree to the radian is performed as follows.

$$\text{Radian} = \text{Degree} \times \frac{\pi}{180}$$

- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

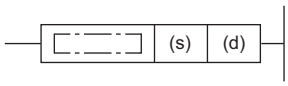
Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{1024}$

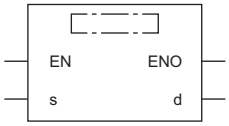
Converting double-precision real number radian to angle

DEGD(P)

These instructions convert the unit of the measure of angle from the radian specified by a double-precision real number to the degree.

Ladder	ST
	ENO:=DEGD(EN,s,d); ENO:=DEGDP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DEGD	
DEGDP	

Setting data

■Description, range, data type

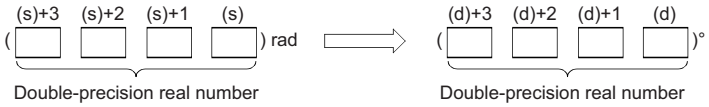
Operand	Description	Range	Data type	Data type (label)
(s)	Angle in radians for which the unit is to be changed to the degree, or the start device containing the angle in radians	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the value converted in degrees	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions convert the unit of the measure of angle from the radian specified by (s) to the degree, and store the angle in degrees in the device number specified by (d).



- Unit conversion from the radian to the degree is performed as follows.

$$\text{Degree} = \text{Radian} \times \frac{180}{\pi}$$

- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

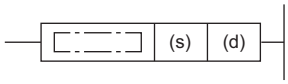
Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{1024}$

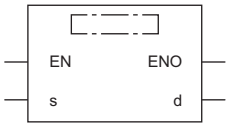
Calculating the square root of single-precision real number

ESQRT(P)

These instructions calculate the square root of the value specified by a single-precision real number.

Ladder	ST
	ENO:=ESQRT(EN,s,d); ENO:=ESQRTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ESQRT	
ESQRTP	

Setting data

■Description, range, data type

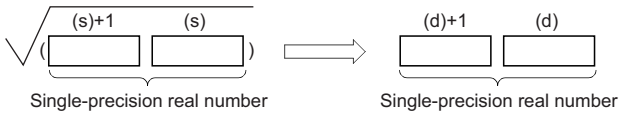
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for square root operation, or the start device containing the data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the square root of the value specified by (s), and store the operation result in the device specified by (d).



- The value specified by (s) must be positive. (No negative value can be calculated.)
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

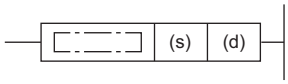
Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3285H	The value in the device specified by (s) is a negative number.

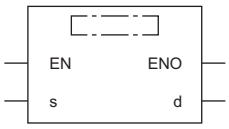
Calculating the square root of double-precision real number

EDSQRT(P)

These instructions calculate the square root of the value specified by a double-precision real number.

Ladder	ST
	ENO:=EDSQRT(EN,s,d); ENO:=EDSQRTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EDSQRT	
EDSQRTP	

Setting data

■Description, range, data type

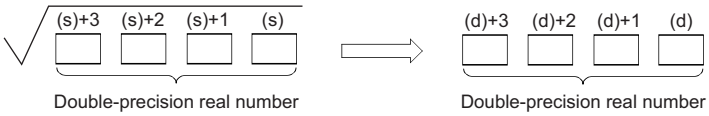
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for square root operation, or the start device containing the data	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the square root of the value specified by (s), and store the operation result in the device specified by (d).



- The value specified by (s) must be positive. (No negative value can be calculated.)
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

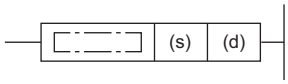
Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3285H	The value in the device specified by (s) is a negative number.

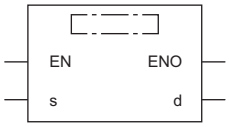
Calculating the exponent of single-precision real number

EXP(P)

These instructions calculate the exponent of the value specified by a single-precision real number.

Ladder	ST*1
	ENO:=EXPP(EN,s,d);

FBD/LD*1



*1 The EXP instruction does not support the ST and FBD/LD. Use the standard function, EXP.
☞ Page 1280 EXP(_E)

■Execution condition

Instruction	Execution condition
EXP	
EXPP	

Setting data

■Description, range, data type

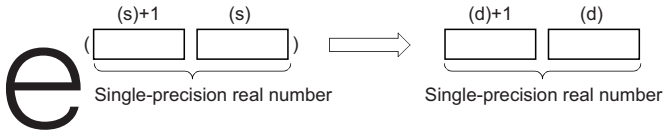
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for exponential operation, or the start device containing the data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the exponent of the value specified by (s), and store the operation result in the device specified by (d).



- Exponent operation is performed with the base (e) set to "2.71828".
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

Point

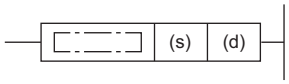
- The EXP(P) instruction performs operation using the natural logarithm. To determine a value using the common logarithm, determine a value by dividing the common logarithm value by 0.43429 and specify it in the device specified by (s).

$$10^{X} = e^{\frac{X}{0.43429}}$$

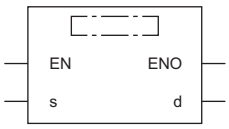
Calculating the exponent of double-precision real number

EXPD(P)

These instructions calculate the exponent of the value specified by a double-precision real number.

Ladder	ST*1
	ENO:=EXPDP(EN,s,d);

FBD/LD*1



*1 The EXPD instruction does not support the ST and FBD/LD. Use the standard function, EXP.
☞ Page 1280 EXP(_E)

■Execution condition

Instruction	Execution condition
EXPD	
EXPDP	

Setting data

■Description, range, data type

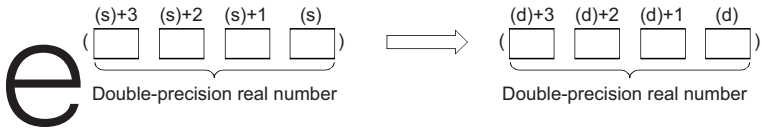
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for exponential operation, or the start device containing the data	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the exponent of the value specified by (s), and store the operation result in the device specified by (d).



- Exponent operation is performed with the base (e) set to "2.71828".
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{1024}$

Point

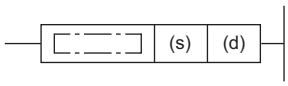
- The EXPD(P) instruction performs operation using the natural logarithm. To determine a value using the common logarithm, determine a value by dividing the common logarithm value by 0.43429 and specify it in the device specified by (s).

$$10^{X} = e^{\frac{X}{0.43429}}$$

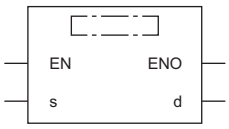
Calculating the natural logarithm of single-precision real number

LOG(P)

These instructions calculate the logarithm using the natural logarithm (e) of the value specified by a single-precision real number as the base.

Ladder	ST*1
	ENO:=LOGP(EN,s,d);

FBD/LD*1



*1 The LOG instruction does not support the ST and FBD/LD. Use the standard function, LOG.
☞ Page 1278 LOG(_E)

■Execution condition

Instruction	Execution condition
LOG	
LOGP	

Setting data

■Description, range, data type

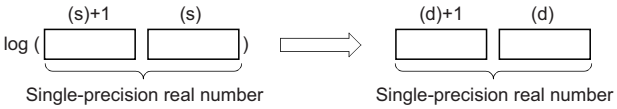
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for natural logarithm operation, or the start device containing the data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the logarithm using natural logarithm (e) of the value specified by (s), and store the operation result in the device specified by (d).



- Input a positive value only. (No negative value can be calculated.)
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

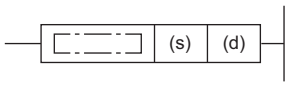
Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3285H	An out-of-range value is set to (s). • The specified value is a negative number. • The specified value is 0.

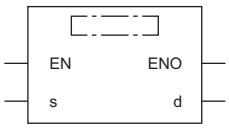
Calculating the natural logarithm of double-precision real number

LOGD(P)

These instructions calculate the logarithm using the natural logarithm (e) of the value specified by a double-precision real number as the base.

Ladder	ST*1
	ENO:=LOGDP(EN,s,d);

FBD/LD*1



*1 The LOGD instruction does not support the ST and FBD/LD. Use the standard function, LOG.
☞ Page 1278 LOG(_E)

■Execution condition

Instruction	Execution condition
LOGD	
LOGDP	

Setting data

■Description, range, data type

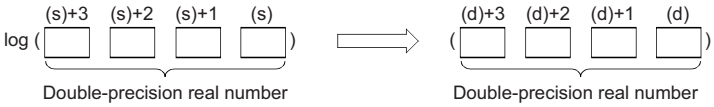
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for natural logarithm operation, or the start device containing the data	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the logarithm using natural logarithm (e) of the value specified by (s), and store the operation result in the device specified by (d).



- Input a positive value only. (No negative value can be calculated.)
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

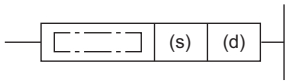
Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3285H	An out-of-range value is set to (s). • The specified value is a negative number. • The specified value is 0.

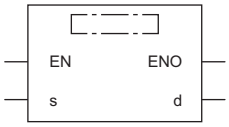
Calculating the square root of BCD 4-digit data

BSQRT(P)

These instructions calculate the square root of the value specified by a BCD 4-digit data.

Ladder	ST
	ENO:=BSQRT(EN,s,d); ENO:=BSQRTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BSQRT	
BSQRTP	

Setting data

■Descriptions, ranges, and data types

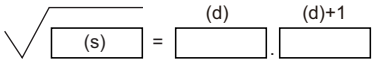
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for square root operation, or the device containing the data	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the operation result	—	BCD 4-digit	ANY16_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions calculate the square root of the BCD 4-digit data specified by (s), and store the operation result in the device specified by (d).



(d): Integral part

(d)+1: Decimal part

- The value to be specified in (s) is a BCD value with a maximum of 4 digits (0 to 9999).
- A BCD value from 0 to 9999.9999 is stored as the operation result in the device specified by (d).
- The operation result is a value whose 5th decimal place is rounded down.

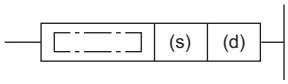
Operation error

Error code (SD0)	Description
3285H	The data in the device specified by (s) is not a BCD value.

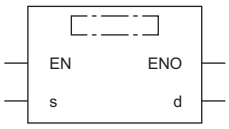
Calculating the square root of BCD 8-digit data

BDSQRT(P)

These instructions calculate the square root of the value specified by a BCD 8-digit data.

Ladder	ST
	ENO:=BDSQRT(EN,s,d); ENO:=BDSQRTP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BDSQRT	
BDSQRTP	

Setting data

■Description, range, data type

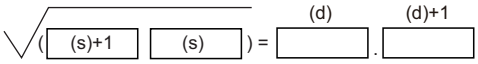
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for square root operation, or the start device containing the data	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the operation result	—	BCD 4-digit	ANY16_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions calculate the square root of the BCD 8-digit data specified by (s), and store the operation result in the device specified by (d).



(s)+1, (s): 2-word data
(d): Integral part
(d)+1: Decimal part

- The value to be specified in (s) is a BCD value with a maximum of 8 digits (0 to 99999999).
- A BCD value from 0 to 9999.9999 is stored as the operation result in the device specified by (d).
- The operation result is a value whose 5th decimal place is rounded down.

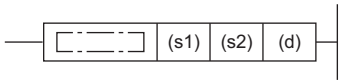
Operation error

Error code (SD0)	Description
3285H	The data in the device specified by (s) is not a BCD value.

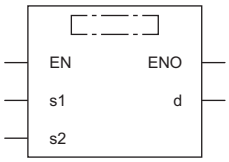
Calculating the exponentiation of single-precision real number

POW(P)

These instructions calculate the exponentiation of a single-precision real number.

Ladder	ST
	ENO:=POW(EN,s1,s2,d); ENO:=POWP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
POW	
POWP	

Setting data

■Description, range, data type

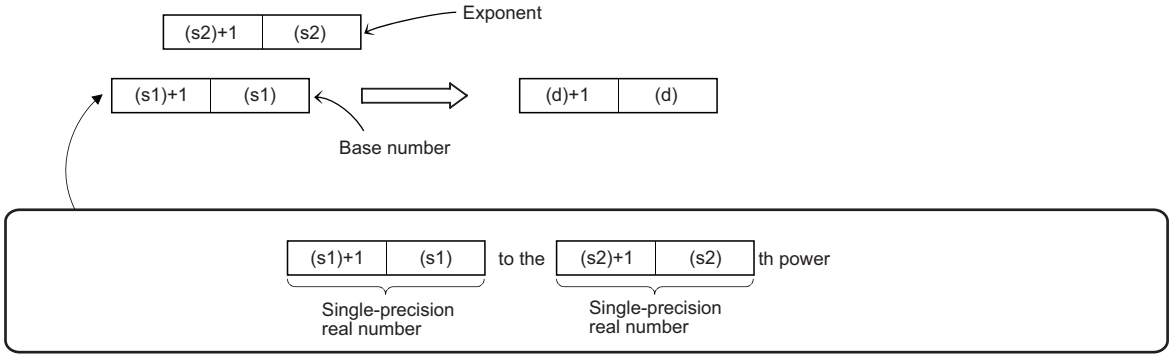
Operand	Description	Range	Data type	Data type (label)
(s1)	Exponentiation recipient data or the start device containing the exponentiation recipient data	0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Exponentiation data or the start device containing the data	0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions raises the single-precision real number specified by (s1) to the power of the single-precision real number specified by (s2), and store the operation result in the device specified by (d).



- The values that can be specified by (s1) and (s2) and the value that can be stored are 0, $2^{-126} \leq |$setting value (stored value)$| < 2^{128}$.
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

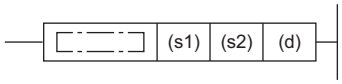
Operation error

Error code (SD0)	Description
3282H	The value input to (s1) or (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{128}$

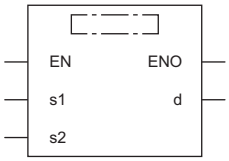
Calculating the exponentiation of double-precision real number

POWD(P)

These instructions calculate the exponentiation of a double-precision real number.

Ladder	ST
	ENO:=POWD(EN,s1,s2,d); ENO:=POWDP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
POWD	
POWDP	

Setting data

■Description, range, data type

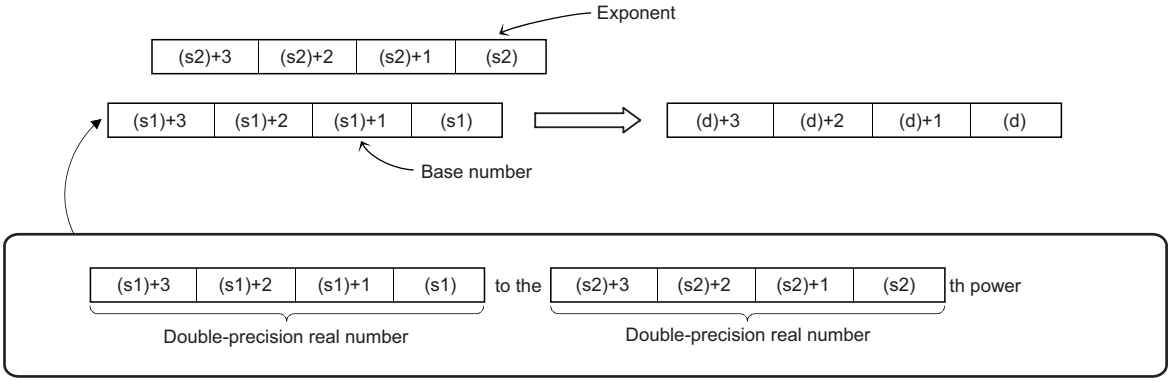
Operand	Description	Range	Data type	Data type (label)
(s1)	Exponentiation recipient data or the start device containing the exponentiation recipient data	0, $2^{-1022} \leq \lvert(s1)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Exponentiation data or the start device containing the data	0, $2^{-1022} \leq \lvert(s2)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions raises the double-precision real number specified by (s1) to the power of the double-precision real number specified by (s2), and store the operation result in the device specified by (d).



- The values that can be specified by (s1) and (s2) and the value that can be stored are 0, $2^{-1022} \leq$|setting value (stored value)|$< 2^{1024}$.
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

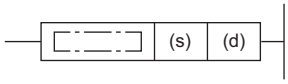
Operation error

Error code (SD0)	Description
3282H	The value input to (s1) or (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\|(d)\| < 2^{128}$

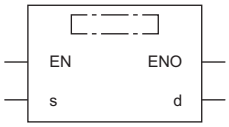
Calculating the common logarithm of single-precision real number

LOG10(P)

These instructions calculate the logarithm using the common logarithm (using 10 as the base) of the value specified by a single-precision real number.

Ladder	ST
	ENO:=LOG10(EN,s,d); ENO:=LOG10P(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
LOG10	
LOG10P	

Setting data

■Description, range, data type

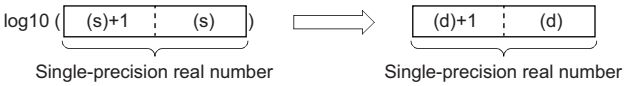
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for common logarithm operation, or the start device containing the data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

- These instructions calculate the common logarithm (using 10 as the base) of the value specified by (s), and store the operation result in the device number specified by (d).



- The value specified by (s) must be positive. (No negative value can be calculated.)
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

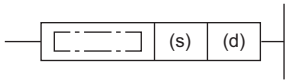
Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3285H	An out-of-range value is set to (s). • The specified value is a negative number. • The specified value is 0.

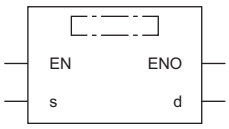
Calculating the common logarithm of double-precision real number

LOG10D(P)

These instructions calculate the logarithm using the common logarithm (using 10 as the base) of the value specified by a double-precision real number.

Ladder	ST
	ENO:=LOG10D(EN,s,d); ENO:=LOG10DP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
LOG10D	
LOG10DP	

Setting data

■Description, range, data type

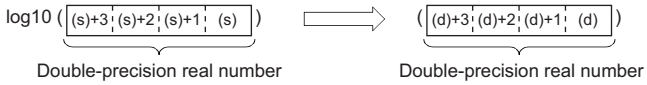
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for common logarithm operation, or the start device containing the data	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions calculate the common logarithm (using 10 as the base) of the value specified by (s), and store the operation result in the device number specified by (d).



Double-precision real number　　Double-precision real number

- The value specified by (s) must be positive. (No negative value can be calculated.)
- If the operation result is -0 or an underflow occurs, the operation result turns out to 0.
- When an input value is set using the engineering tool, a rounding error may occur. Refer to the following for the precautions on setting input values using the engineering tool.

☞ Page 40 Precautions

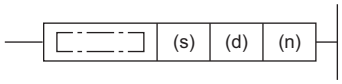
Operation error

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3285H	An out-of-range value is set to (s). • The specified value is a negative number. • The specified value is 0.

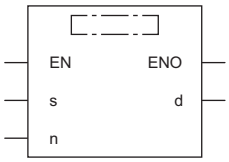
Searching the maximum value of single-precision real number

EMAX(P)

These instructions search the block data of single-precision real numbers for the maximum value.

Ladder	ST*1
	ENO:=EMAXP(EN,s,n,d);

FBD/LD*1



*1 The EMAX instruction does not support the ST and FBD/LD. Use the standard function, MAX.
☞ Page 1316 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
EMAX	
EMAXP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Single-precision real number used for maximum value search, or the start device containing single-precision real numbers	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32*1
(d)	Start device for storing the search result (d) to (d)+1: Maximum value (d)+2: Position (d)+3: The number of search target data points	—	Single-precision real number	—*2 (ANY_REAL_32_ARRAY)
(n)	Number of single-precision real number block data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

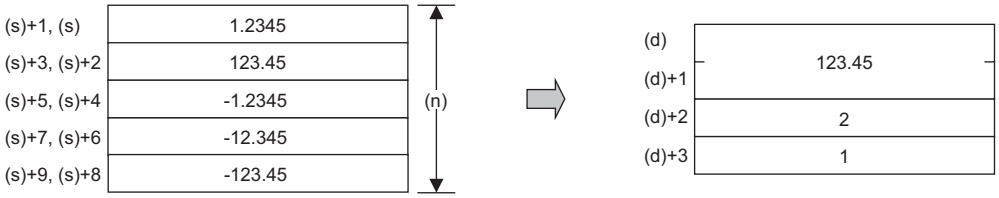
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.
*2 Specify a device regardless of the programming language used. Do not specify labels.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search for the maximum value in the (n) points of single-precision real number block data in the device starting from the one specified by (s), and store the maximum value in the device specified by (d). The instructions store the location of the first maximum value by the number of points from (s) in the device specified by (d)+2 and the number of maximum values in the device specified by (d)+3.
- The start of the block data in the device specified by (s) is counted as the 1st point when the search result (location) is counted.



(d), (d)+1: Maximum value
(d)+2: Position
(d)+3: The number of search target data points

- When (n) is 0, the processing is not performed.

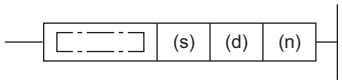
Operation error

Error code (SD0)	Description
3282H	The block data in the device specified by (s) includes a value other than single-precision real number.

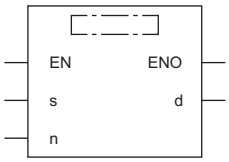
Searching the maximum value of double-precision real number

EDMAX(P)

These instructions search the block data of double-precision real numbers for the maximum value.

Ladder	ST[*1]
	ENO:=EDMAXP(EN,s,n,d);

FBD/LD[*1]



*1 The EDMAX instruction does not support the ST and FBD/LD. Use the standard function, MAX.
☞ Page 1316 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
EDMAX	
EDMAXP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Double-precision real number used for maximum value search, or the start device containing double-precision real numbers	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64[*1]
(d)	Start device for storing the search result (d) to (d)+3: Maximum value (d)+4: Position (d)+5: The number of search target data points	—	Double-precision real number	—[*2] (ANY_REAL_64_ARRAY)
(n)	Number of double-precision real number block data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

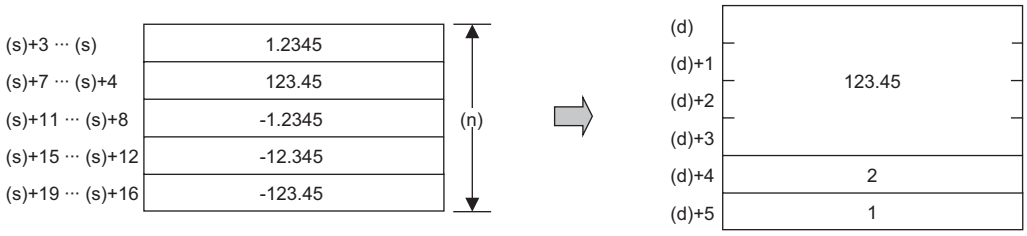
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.
*2 Specify a device regardless of the programming language used. Do not specify labels.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search for the maximum value in the (n) points of double-precision real number block data in the device starting from the one specified by (s), and store the maximum value in the device specified by (d). The instructions store the location of the first maximum value by the number of points from (s) in the device specified by (d)+4 and the number of maximum values in the device specified by (d)+5.
- The start of the block data in the device specified by (s) is counted as the 1st point when the search result (location) is counted.



(d), (d)+1, (d)+2, (d)+3: Maximum value
(d)+4: Location
(d)+5: Number of maximum values

- When (n) is 0, the processing is not performed.

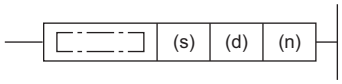
Operation error

Error code (SD0)	Description
3282H	The block data in the device specified by (s) includes a value other than double-precision real number.

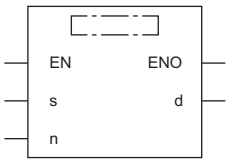
Searching the minimum value of single-precision real number

EMIN(P)

These instructions search the block data of single-precision real numbers for the minimum value.

Ladder	ST*1
	ENO:=EMINP(EN,s,n,d);

FBD/LD*1



*1 The EMIN instruction does not support the ST and FBD/LD. Use the standard function, MIN.
☞ Page 1316 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
EMIN	
EMINP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Single-precision real number used for minimum value search, or the start device containing single-precision real numbers	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32*1
(d)	Start device for storing the search result (d) to (d)+1: Minimum value (d)+2: Position (d)+3: The number of search target data points	—	Single-precision real number	—*2 (ANY_REAL_32_ARRAY)
(n)	Number of single-precision real number block data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

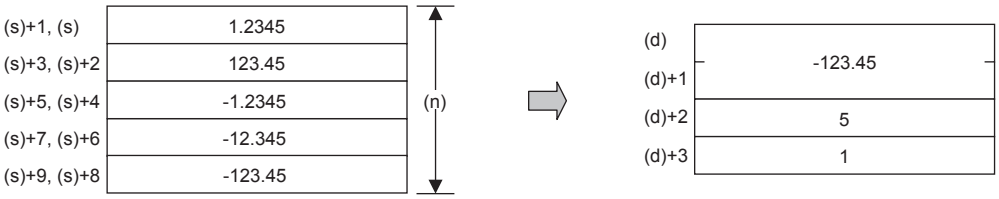
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.
*2 Specify a device regardless of the programming language used. Do not specify labels.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search for the minimum value in the (n) points of single-precision real number block data in the device starting from the one specified by (s), and store the maximum value in the device specified by (d). The instructions store the location of the first minimum value by the number of points from (s) in the device specified by (d)+2 and the number of minimum values in the device specified by (d)+3.
- The start of the block data in the device specified by (s) is counted as the 1st point when the search result (location) is counted.



(d), (d)+1: Minimum value
(d)+2: Position
(d)+3: The number of search target data points

- When (n) is 0, the processing is not performed.

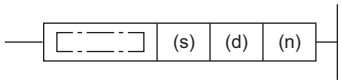
Operation error

Error code (SD0)	Description
3282H	The block data in the device specified by (s) includes a value other than single-precision real number.

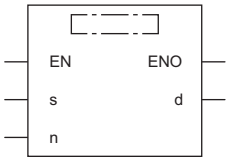
Searching the minimum value of double-precision real number

EDMIN(P)

These instructions search the block data of double-precision real numbers for the minimum value.

Ladder	ST*1
	ENO:=EDMINP(EN,s,n,d);

FBD/LD*1



*1 The EDMIN instruction does not support the ST and FBD/LD. Use the standard function, MIN.
☞ Page 1316 MAX(_E), MIN(_E)

■Execution condition

Instruction	Execution condition
EDMIN	
EDMINP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Double-precision real number used for minimum value search, or the start device containing double-precision real numbers	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64*1
(d)	Start device for storing the search result (d) to (d)+3: Minimum value (d)+4: Position (d)+5: The number of search target data points	—	Double-precision real number	—*2 (ANY_REAL_64_ARRAY)
(n)	Number of double-precision real number block data points	0 to 65535	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

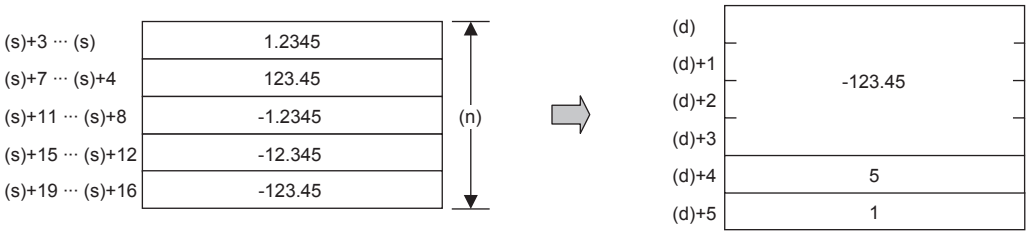
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.
*2 Specify a device regardless of the programming language used. Do not specify labels.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search for the minimum value in the (n) points of double-precision real number block data in the device starting from the one specified by (s), and store the maximum value in the device specified by (d). The instructions store the location of the first minimum value by the number of points from (s) in the device specified by (d)+4 and the number of minimum values in the device specified by (d)+5.
- The start of the block data in the device specified by (s) is counted as the 1st point when the search result (location) is counted.



(d), (d)+1, (d)+2, (d)+3: Minimum value
(d)+4: Location
(d)+5: Number of minimum values

- When (n) is 0, the processing is not performed.

Operation error

Error code (SD0)	Description
3282H	The block data in the device specified by (s) includes a value other than double-precision real number.

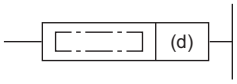
18 RANDOM NUMBER

18.1 Random Number Instructions

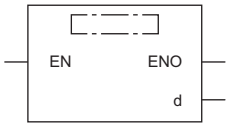
Generating random number

RND(P)

These instructions generate a random number between 0 and less than 32767, and store the random number in the specified device.

Ladder	ST
	ENO:=RND(EN,d); ENO:=RNDP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
RND	
RNDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Device for storing the random number	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

These instructions generate a random number between 0 and 32767, and store the random number in the device specified by (d). The RND(P) instruction generates random numbers according to a certain calculation expression. The calculation expression uses the previous calculation result as a coefficient.

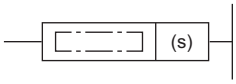
Operation error

There is no operation error.

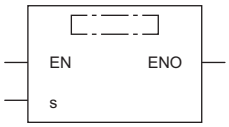
Changing random sequence

SRND(P)

These instructions change the random number sequence according to the content of the 16-bit binary data stored in the specified device.

Ladder	ST
	ENO:=SRND(EN,s); ENO:=SRNDP(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
SRND	
SRNDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Random number sequence data	-32768 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

These instructions change the random number sequence according to the content of the 16-bit binary data stored in the device specified by (s). The SRND(P) instruction can change the random number calculation pattern.

Operation error

There is no operation error.

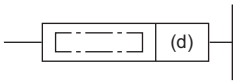
19 DEVICE OPERATION

19.1 Index Register Instructions

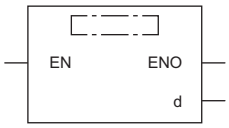
Saving all data of the index register

ZPUSH(P)

These instructions save the content of the index register to the specified area.

Ladder	ST
	ENO:=ZPUSH(EN,d); ENO:=ZPUSHP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ZPUSH	
ZPUSHP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device to which the index register will be saved	—	16-bit signed binary	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying data with a label, define the array so that an area required for operation can be secured, and specify the array label element.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions save the contents of the index register, long index register, local index register, and local long index register in the device specified by (d) or the array label element and later. The local index register and local long index register to be saved are the local index register and local long index register of the program that executed the ZPUSH instruction.
- When the content of the index register is saved, the number of saves (d) is incremented by 1.
- Regardless of the number of points assigned
- to the index register, long index register, local index register and local long index register, 48 words of data are saved.
- The ZPOP(P) instructions can be used to restore data. The ZPUSH(P) and ZPOP(P) instructions are used in pairs and can be nested to be used as a stack.

☞ Page 963 ZPOP(P)

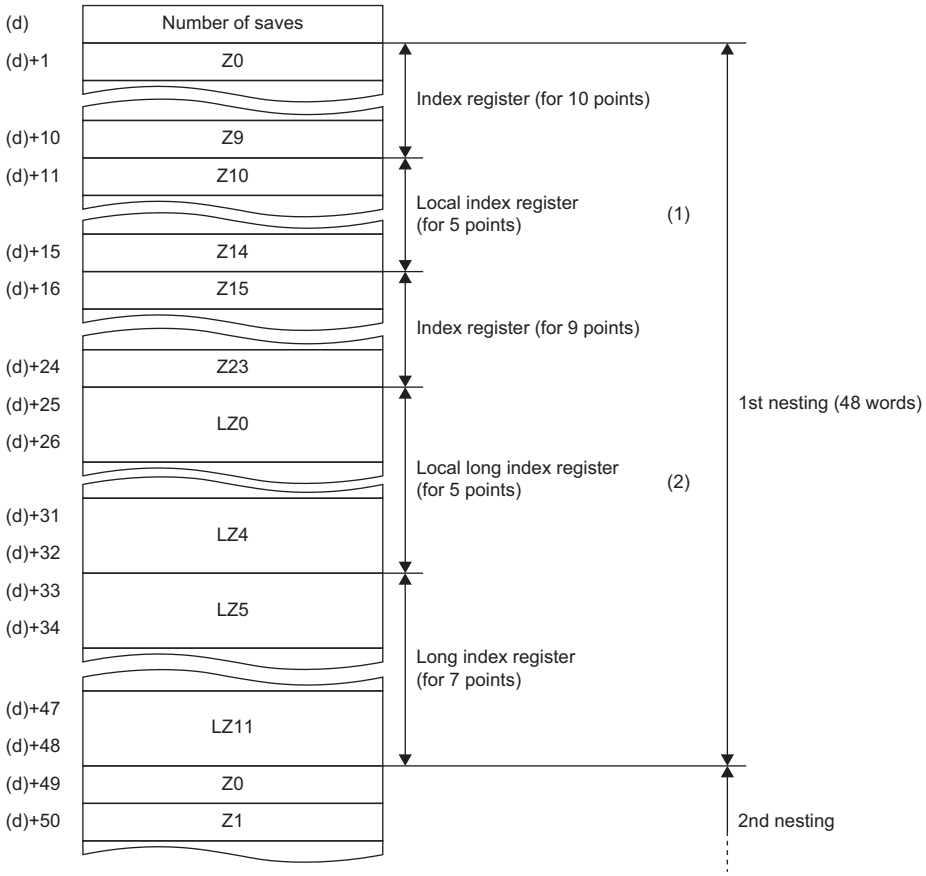
- A nesting structure can be created by specifying the area specified by (d) of the ZPUSH(P) instruction in (d) of the ZPUSH(P) instruction again. The number of saves (d) is incremented by 1 every time the ZPUSH(P) instruction is executed.
- When another area is specified in (d) of the ZPUSH(P) instruction in the nesting structure, the content of the index register is saved to the specified another area.
- In the nesting structure, every time the ZPUSH(P) instruction is executed, saved data is added. Therefore, secure in advance the areas necessary for the number of times the instruction is executed.

Ex.

When the local setting of the index register setting is set as follows in the engineering tool, the following figure shows the configuration of the areas used after (d).

[CPU Parameter] ⇨ [Memory/Device Setting] ⇨ [Index Register Setting]

Item			Set value
Local setting	Number of points setting	Local Index register (Z)	5 points
		Local long index register (LZ)	5 points
	Start	Index register (Z)	10
		Long index register (LZ)	0



(1) Save the local index register for 5 points from Z10.
(2) Save the local long index register for 5 points from LZ0.

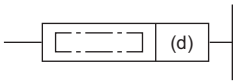
Operation error

Error code (SD0)	Description
3285H	The number of saves in (d)+0 is FFFFH.

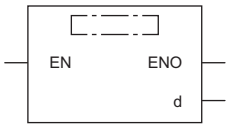
Returning all data of the index register

ZPOP(P)

These instructions read the data, which has been saved to the specified area, into the index register.

Ladder	ST
	ENO:=ZPOP(EN,d); ENO:=ZPOPP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ZPOP	
ZPOPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device to which the index register will be restored	—	16-bit signed binary	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying data with a label, define the array so that an area required for operation can be secured, and specify the array label element.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions read the data, which has been saved to the device specified by (d) or the array label element and later, into the index register, long index register, local index register, and local long index register.
- When the content of the index register is read, the number of saves (d) is decremented by 1.
- The local index register and local long index register to be restored are the local index register and local long index register of the program that executed the ZPOP instruction.
- Refer to the following for the configuration of the areas used after (d).

☞ Page 960 ZPUSH(P)

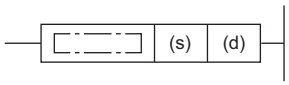
Operation error

Error code (SD0)	Description
3285H	The number of saves in (d)+0 is 0.

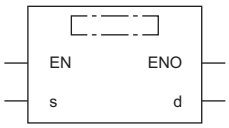
Saving the selected data of the index register and long index register

ZPUSH(P)

These instruction save the contents of the index register and long index register to the specified area.

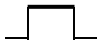
Ladder	ST
	ENO:=ZPUSH_2(EN,s,d); ENO:=ZPUSHP_2(EN,s,d);

FBD/LD



(☐ is replaced by ZPUSH_2 or ZPUSHP_2.)

■Execution condition

Instruction	Execution condition
ZPUSH	
ZPUSHP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Type of the index register and long index register to be saved	1 to 3	16-bit unsigned binary	ANY16
(d)	Start device to which the index register and long index register will be saved	—	16-bit signed binary	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying data with a label, define the array so that an area required for operation can be secured, and specify the array label element.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	○*1	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

*1 Only U3E☐\G☐ can be used.

Processing details

- These instructions save the index register, long index register, local index register, and local long index register to be saved in the device specified by (d) or the array label element and later.
- When the contents of the index register, long index register, local index register and local long index register are saved, the number of saves (d) is incremented by 1.
- The local index register and local long index register to be saved are the local index register and local long index register of the program that executed the ZPUSH instruction.
- The type of the index register, long index register, local index register and local long index register to be saved are stored to the end of the saved data.
- The following table lists the values specified by (s), and the index register, long index register, local index register, and local long index register to be saved.

Value of (s)	Z and LZ saved
1	Total range of Z and LZ
2	Total range of Z
3	Total range of LZ

- The ZPOP(P) instructions (restoring the selected data of the index register and long index register) are used to restore data. The ZPUSH(P) and ZPOP(P) instructions are used in pairs and can be nested to be used as a stack.

☞ Page 968 ZPOP(P)

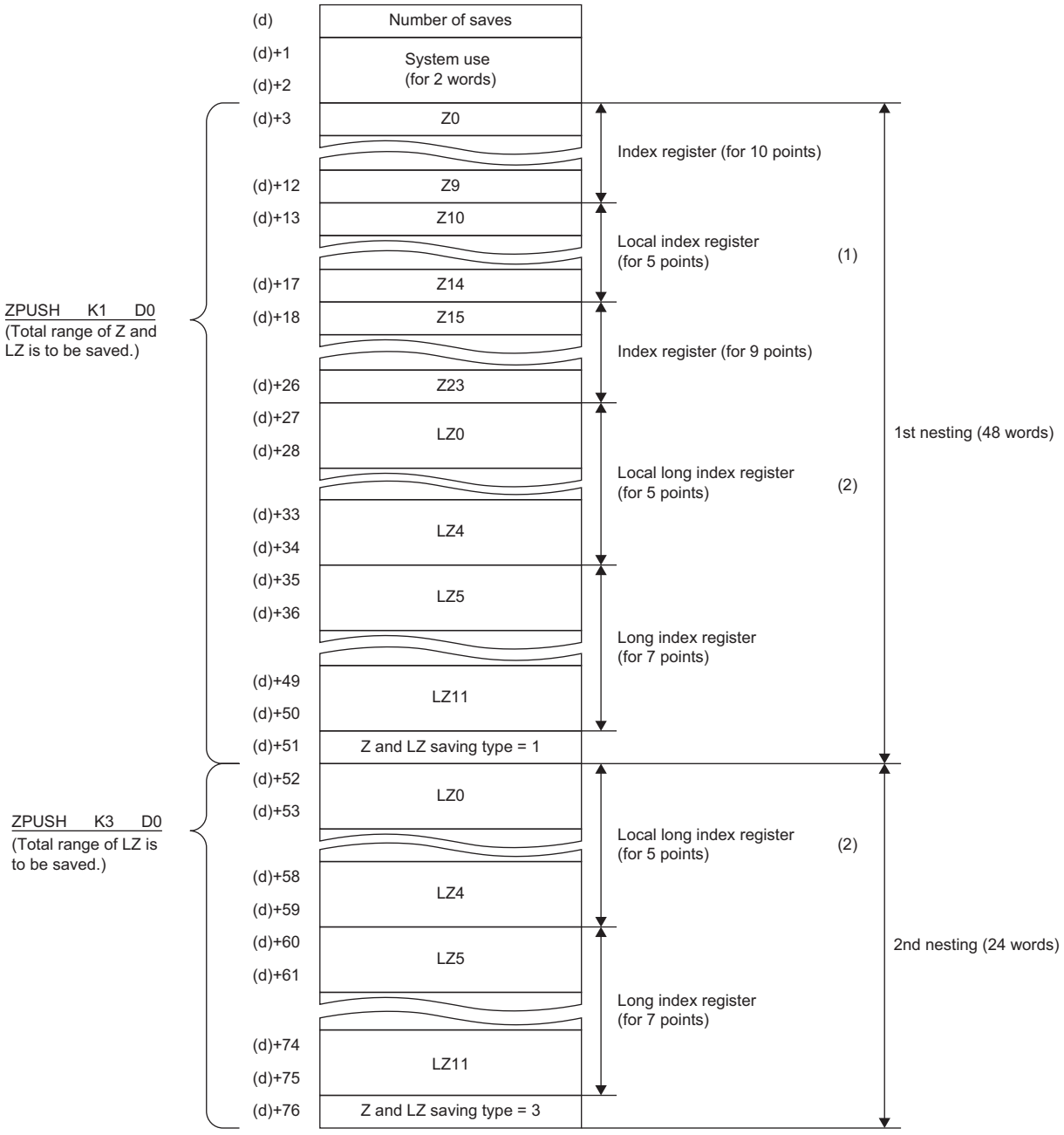
- A nesting structure can be created by specifying the area specified by (d) of the ZPUSH(P) instruction in (d) of the ZPUSH(P) instruction again. The number of saves (d) is incremented by 1 every time the ZPUSH(P) instruction is executed.
- When another area is specified in (d) of the ZPUSH(P) instruction in the nesting structure, the content of the index register or long index register is saved to the specified another area.
- In the nesting structure, every time the ZPUSH(P) instruction is executed, saved data is added. Therefore, check the numbers of points assigned to the index register and long index register according to SD300 and SD302, and secure in advance the areas necessary for the number of times the instruction is executed.

Ex.

When the local setting of the index register setting is set as follows in the engineering tool, the following figure shows the configuration of the areas used after (d).

[CPU Parameter] ⇨ [Memory/Device Setting] ⇨ [Index Register Setting]

Item			Set value
Local setting	Number of points setting	Local Index register (Z)	5 points
		Local long index register (LZ)	5 points
	Start	Index register (Z)	10
		Long index register (LZ)	0



(1) Save the local index register for 5 points from Z10.
(2) Save the local long index register for 5 points from LZ0.

Precautions

(d)+1 and (d)+2 for the ZPUSH(P) instructions are used for the system. Do not change the values.
The Z and LZ save types stored in the area specified by (d) and later are also used for the system. Do not change the values.
Changing the values may cause malfunction of the module.

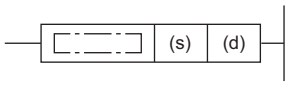
Operation error

Error code (SD0)	Description
3285H	An out-of-range value is set to (s). The specified value is other than 1 to 3.
	The value stored in the system-reserved area in the area specified by (d) has been changed.
	The number of saves in (d) is FFFFH.

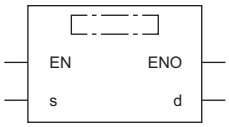
Returning the selected data of the index register and long index register

ZPOP(P)

These instructions read the data, which has been saved to the specified area, into the index register and long index register.

Ladder	ST
	ENO:=ZPOP_2(EN,s,d); ENO:=ZPOPP_2(EN,s,d);

FBD/LD



(☐ is replaced by ZPOP_2 or ZPOPP_2.)

■Execution condition

Instruction	Execution condition
ZPOP	
ZPOPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Dummy	—	16-bit unsigned binary	ANY16
(d)	Start device to which the index register will be restored	—	16-bit signed binary	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying data with a label, define the array so that an area required for operation can be secured, and specify the array label element.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	○*1	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

*1 Only U3E☐\G☐ can be used.

Processing details

- These instructions read the data, which has been saved to the device specified by (d) or the array label element and later, into the index register, long index register, local index register, and local long index register.
- When the data saved to the index register and long index register is read, the number of saves (d) is decremented by 1.
- The local index register and local long index register to be restored are the local index register and local long index register of the program that executed the ZPOP instruction.
- The date data in the device specified by (s) is regarded as dummy data and ignored.
- Refer to the following for the configuration of the areas used after (d).

☞ Page 964 ZPUSH(P)

Precautions

(d)+1 and (d)+2 are used for the system. Do not change the values.
The Z and LZ save types stored in the area specified by (d) and later are also used for the system. Do not change the values.
Changing the values may cause malfunction of the module.

Operation error

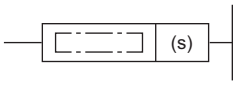
Error code (SD0)	Description
3285H	The number of saves in (d)+0 is 0.
	The value stored in the system-reserved area in the area specified by (d) has been changed.
	A value other than 1 to 3 is set to the Z, LZ save type.

19.2 File Register Operation Instructions

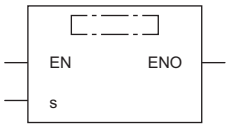
Switching the file register block number

RSET(P)

These instructions change the block number of the file register used in the program.

Ladder	ST
	ENO:=RSET(EN,s); ENO:=RSETP(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
RSET	
RSETP	

Setting data

■Description, range, data type

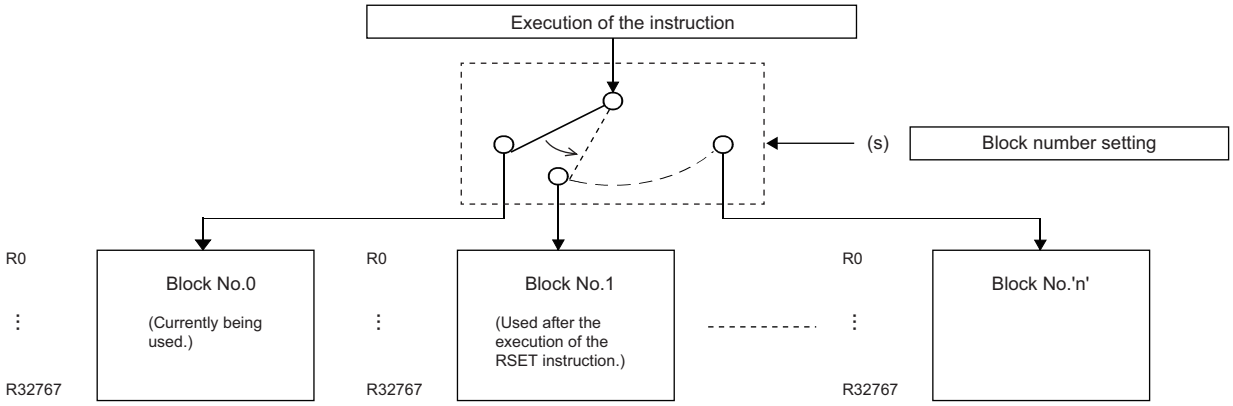
Operand	Description	Range	Data type	Data type (label)
(s)	Block number data to be changed or the device number where the block number data is stored	0 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions switch the block number of the file register used in the program to that stored in the device specified by (s). After the block number is changed, all file registers processed by the sequence program are those linked to the new block number.



Precautions

If "Block No. Save/Recovery Setting" of the file register from the engineering tool is set to "Not Saved/Recovered", do not use this instruction to switch the file register block number in the interrupt program. If the block number is switched, the change may not be reflected immediately after returning from the interrupt program, and the block number may not be switched until the start of the next program.

For the restrictions of the file register, refer to the following.

☞ Page 58 Restrictions on using file registers

Operation error

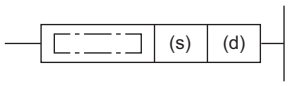
Error code (SD0)	Description
2820H	The specified file register does not exist.
3285H	The block number in the device specified by (s) does not exist.

19.3 File Register Read/Write Instructions

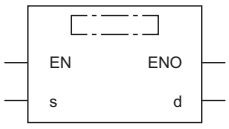
Reading 1-byte data from the file register

ZRRDB(P)

These instructions read the data from the file register with the specified serial byte number.

Ladder	ST
	ENO:=ZRRDB(EN,s,d); ENO:=ZRRDBP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ZRRDB	
ZRRDBP	

Setting data

■Descriptions, ranges, and data types

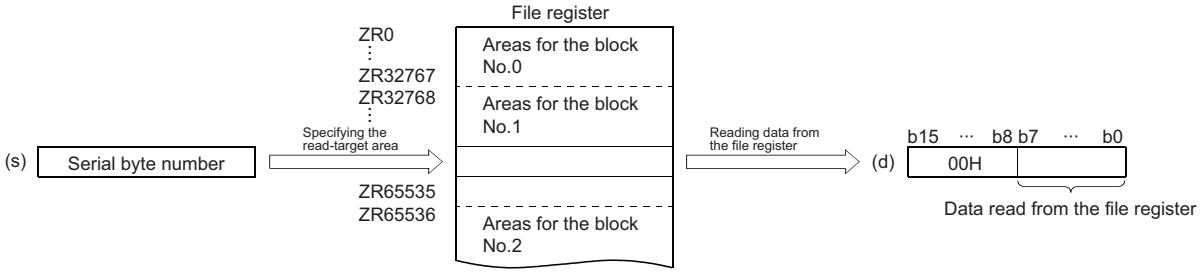
Operand	Description	Range	Data type	Data type (label)
(s)	Serial byte number of the file register to be read	0 to 4294967295	32-bit unsigned binary	ANY32
(d)	Start number of the device for storing the data that has been read	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

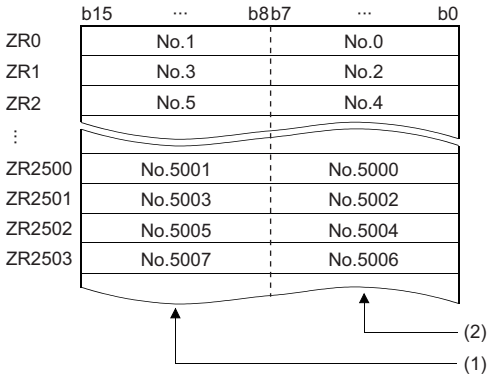
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- Without recognizing block numbers, these instructions read the data from the file register with the serial byte number specified by (s), and store the data in the lower 8 bits of the device specified by (d). 00H is stored in the upper 8 bits of the device specified by (d).



- The following figure shows the file register numbers corresponding to serial byte numbers.



(1) Data areas when an odd number is specified
(2) Data areas when an even number is specified

Ex.

When 23560 is specified in (s), the data in the lower 8 bits of ZR11780 is read.



(1) Specifying the read-target area

Ex.

When 43257 is specified in (s), the data in the upper 8 bits of ZR21628 is read.



(1) Specifying the read-target area

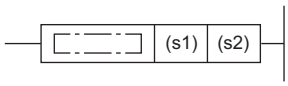
Operation error

Error code (SD0)	Description
2820H	The specified device number (serial byte number) is out of range.

Writing 1-byte data to the file register

ZRWRB(P)

These instructions write the data in the lower bits of the specified device to the file register with the specified serial byte number.

Ladder	ST
	ENO:=ZRWRB(EN,s1,s2); ENO:=ZRWRBP(EN,s1,s2);

FBD/LD



■Execution condition

Instruction	Execution condition
ZRWRB	
ZRWRBP	

Setting data

■Descriptions, ranges, and data types

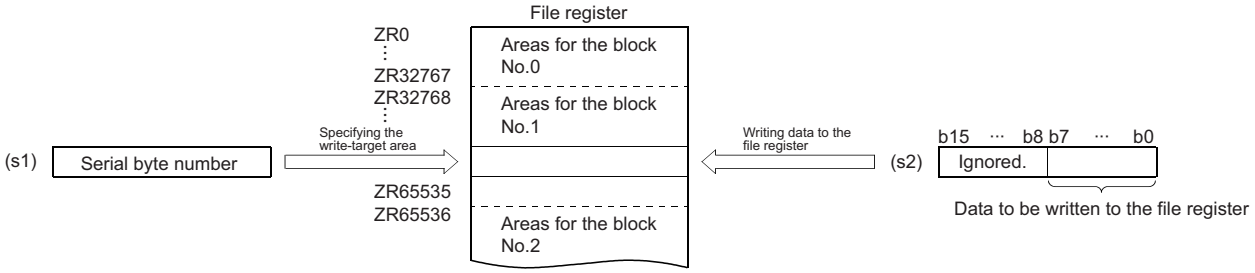
Operand	Description	Range	Data type	Data type (label)
(s1)	Serial byte number of the file register to be written	0 to 4294967295	32-bit unsigned binary	ANY32
(s2)	Device number where the write data is stored	-32768 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

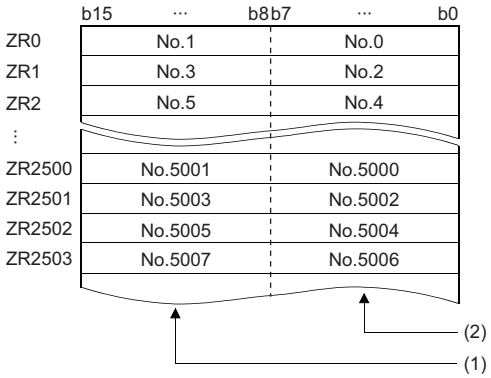
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- Without recognizing block numbers, these instructions write the lower 8-bit data stored in the device specified by (s2) to the file register with the serial byte number specified by (s1). The instructions ignore upper 8-bit data in the device specified by (s2).



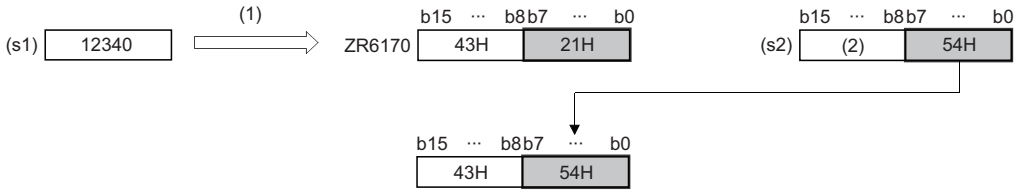
- The following figure shows the file register numbers corresponding to serial byte numbers.



(1) Data areas when an odd number is specified
(2) Data areas when an even number is specified

Ex.

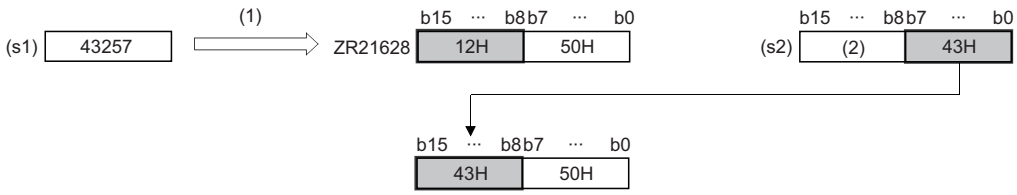
When 12340 is specified in (s1), data is written to the lower 8 bits of ZR11170.



(1) Specifying the write-target area
(2) Ignored

Ex.

When 43257 is specified in (s1), data is written to the upper 8 bits of ZR21628.



(1) Specifying the write-target area
(2) Ignored

Operation error

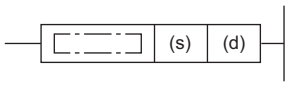
Error code (SD0)	Description
2820H	The specified device number (serial byte number) is out of range.

19.4 Indirect Address Read Instructions

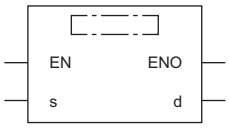
Reading the indirect address

ADRSET(P)

These instructions read the indirect address of the specified device.

Ladder	ST
	ENO:=ADRSET(EN,s,d); ENO:=ADRSETP(EN,s,d); ENO:=_ADRSET(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ADRSET	
ADRSETP	

Setting data

■Descriptions, ranges, and data types

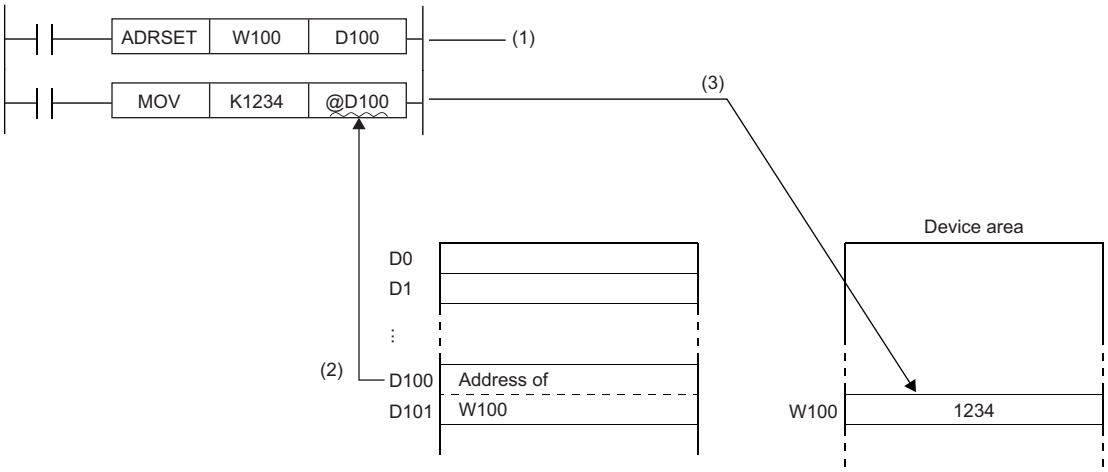
Operand	Description	Range	Data type	Data type (label)
(s)	Number of the device from which the indirect address is to be read	—	Device name	ANY_ELEMENTARY
(d)	Start number of the device for storing the indirect address of the device specified by (s)	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	—	○	—	○	—	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions store the indirect address of the device specified by (s), and stores it in the device specified by (d). The address stored in the devices specified by (d)+0 and (d)+1 is used for indirect addressing of the device in the program.



(1) The address of W100 is stored in D100 and D101.
(2) The data (address of W100) stored in D100 and D101 is specified.
(3) "1234" is written to W100.

- Digit specification of bit device or bit specification of word device in (s) is not permitted
- For the indirect specification of devices, refer to the following.

User's manual for the controller used

- Index specification of labels is not supported. When using a label by specifying it in (d), follow the method below.

Label	Description

Global label specifying a device	When using it as an indirect address, use the indirect specification of the device specified in the global label.
	■Example of ST
	ADRSET(TRUE , intVar , gvAddr); // Read the intVar indirect address to gvAddr.
	INC (TRUE , @D0);// Use the indirect specification of device D0 specified for gvAddr.

Automatic assignment global label/ local label	Transfer the indirect address to the device and use the indirect specification of the transfer destination device.
	■Example of ST
	ADRSET(TRUE , intVar , lvAddr); // Read the intVar indirect address to lvAddr.
	DMOV(TRUE , lvAddr , D0);// Transfer the indirect address, which has been read to lvAddr, to the device.
	INC (TRUE , @D0);// Use the indirect specification of the device to which the indirect address was transferred.

Operation error

There is no operation error.

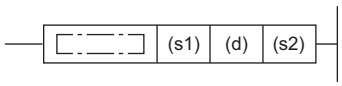
20 TIMER, COUNTER

20.1 Special Counter Instructions

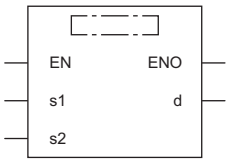
Counting up or down the current value (1-phase input)

UDCNT1

This instruction updates the current value of the specified counter.

Ladder	ST
	ENO:=UDCNT1(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UDCNT1	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	(s1)+0: Count input number (s1)+1: Count up/down flag Off indicates count-up (counting up the current value). On indicates count-down (counting down the current value).	—	Bit	ANYBIT_ARRAY*1 (Number of elements: 2)
(d)	Number of the counter (device name) to be counted by the UDCNT1 instruction	—	Device name	ANY16*2
(s2)	Set value	-32768 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to device (X) can be used.
*2 Only labels assigned to device (C) can be used.

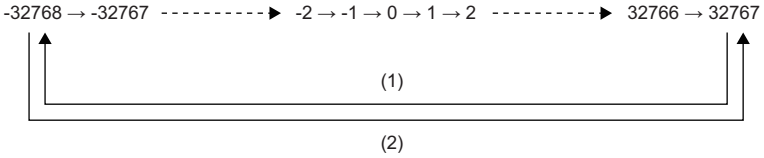
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○*1	—	—	—	—	—	—	—	—	—	—	—
(d)	—	—	○*2	—	—	—	—	—	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—

*1 Only X can be used. Note, however, that it can be used only within the range of the number of I/O points (the number of points that can access I/O modules).
*2 Only C can be used.

Processing details

- When the input specified by (s1) is turned on, this instruction updates the current value of the counter specified by (d).
- Counting up or down is determined by whether the input specified by (s1)+1 is on or off.
  - Off: Count-up (counting up the current value)
  - On: Count-down (counting down the current value)
- Count processing is performed as follows.
  - When the current value equals the value specified by (s2) during count-up, the contact of the counter specified by (d) is turned on. The current value is kept counting even when the contact of the counter is turned on.
  - When the current value equals "the set value - 1" during countdown, the contact of the counter specified by (d) is turned off.
  - The counter specified by (d) is a ring counter. Counting up the counter when the current value is 32767 proceeds to -32768. Similarly, counting down the counter when the current value is -32768 proceeds to 32767. The following figure shows the processing for counting the current value.



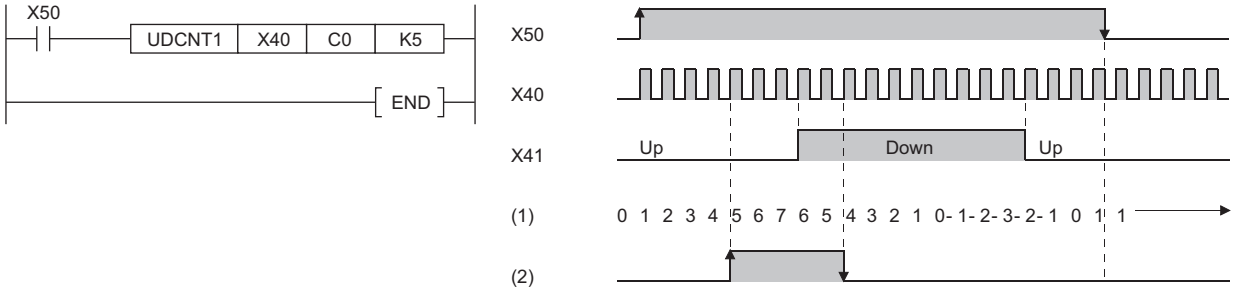
(1) When counting up

(2) When counting down

- The following figure shows the operation of count processing using the UDCNT1 instruction.

Ex.

Program which uses C0 (up/down counter) to count the number of times X40 turns off and on after X50 turns on



(1) Current value of C0

(2) Contact of C0

- When executed, the UDCNT1 instruction starts counting when the execution command turns on and stops counting when the command turns off. If the execution command is turned on again, the instruction resumes counting from the current value with which it stopped counting previously.
- The RST instruction is used to clear the current value of the counter specified by (d) and turn off the contact.

Point

- The UDCNT1 instruction stores the device data of the argument in the work area of the controller, and performs the actual count operation using system interrupts. (The device data stored in the work area of the controller is cleared by turning off the execution command or setting it to STOP then RUN.) For this reason, the pulses that can be counted must have longer on/off time than the interval of the controller. The interrupt interval of the controller is 1ms.
- The set value cannot be changed during counting by the UDCNT1 instruction (while the execution command is on). To change the set value, turn off the execution command in advance.
- The counter specified by the UDCNT1 instruction cannot be used by any other instruction. If another instruction uses it, normal counting is disabled.
- The UDCNT1 instruction can be used a maximum of six times in all running programs. The seventh or subsequent UDCNT1 instruction, if issued, causes no processing.

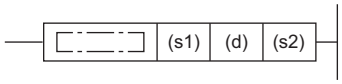
Operation error

There is no operation error.

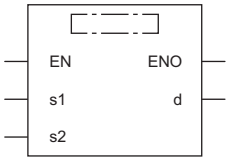
Counting up or down the current value (2-phase input)

UDCNT2

This instruction updates the current value of the counter depending on the status of phases A and B pulses.

Ladder	ST
	ENO:=UDCNT2(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UDCNT2	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	(s1)+0: Count input number (phase A pulse) (s1)+1: Count input number (phase B pulse)	—	Bit	ANYBIT_ARRAY*1 (Number of elements: 2)
(d)	Number of the counter (device name) to be counted by the UDCNT2 instruction	—	Device name	ANY16*2
(s2)	Set value	-32768 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to device (X) can be used.
*2 Only labels assigned to device (C) can be used.

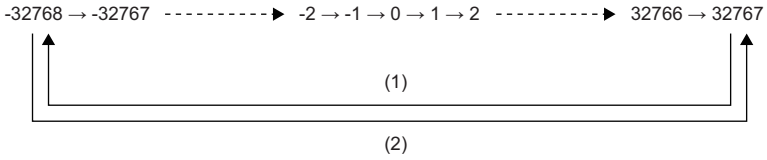
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○*1	—	—	—	—	—	—	—	—	—	—	—
(d)	—	—	○*2	—	—	—	—	—	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—

*1 Only X can be used. Note, however, that it can be used only within the range of the number of I/O points (the number of points that can access I/O modules).
*2 Only C can be used.

Processing details

- This instruction updates the current value of the counter specified by (d) depending on the status of the input (phase A pulse) specified by (s1) and the status of the input (phase B pulse) specified by (s1)+1.
- Counting up or down is determined as follows.
  - (s1)+1 is turned on while (s1) is on: Count-up (counting up the current value)
  - (s1)+1 is turned off while (s1) is on: Countdown (counting down the current value)
  - The instruction does not count while (s1) is off.
- Count processing is performed as follows.
  - When the current value equals the value specified by (s2) during count-up, the contact of the counter specified by (d) is turned on. The current value is kept counting even when the contact of the counter is turned on.
  - When the current value equals "the set value - 1" during countdown, the contact of the counter specified by (d) is turned off.
  - The counter specified by (d) is a ring counter. Counting up the counter when the current value is 32767 proceeds to -32768. Similarly, counting down the counter when the current value is -32768 proceeds to 32767. The following figure shows the processing for counting the current value.



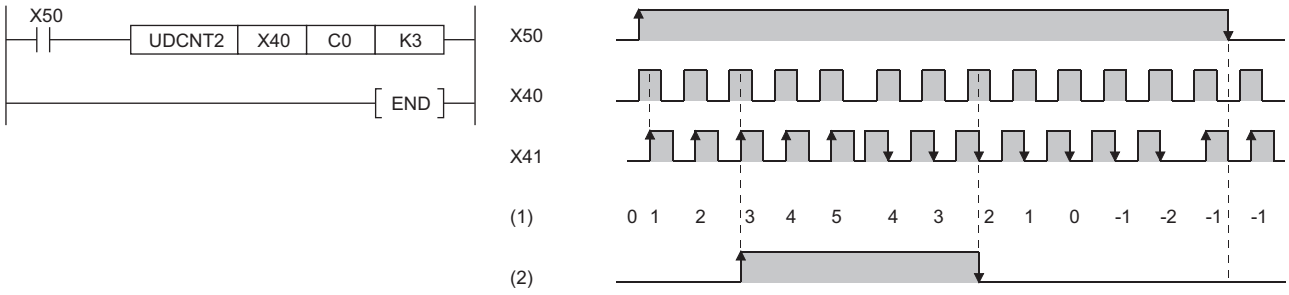
(1) When counting up

(2) When counting down

- The following figure shows the operation of count processing using the UDCNT2 instruction.

Ex.

Program which uses C0 (up/down counter) to count the states of X40 and X41 after X50 turns on



(1) Current value of C0

(2) Contact of C0

- When executed, the UDCNT2 instruction starts counting when the execution command turns on and stops counting when the command turns off. If the execution command is turned on again, the instruction resumes counting from the current value with which it stopped counting previously.
- The RST instruction is used to clear the current value of the counter specified by (d) and turn off the contact.

Point

- The UDCNT2 instruction stores the device data of the argument in the work area of the controller, and performs the actual count operation using system interrupts. (The device data stored in the work area of the controller is cleared by turning off the execution command or setting it to STOP then RUN.) For this reason, the pulses that can be counted must have longer on/off time than the interval of the controller. The interrupt interval of the controller is 1ms.
- The set value cannot be changed during counting by the UDCNT2 instruction (while the execution command is on). To change the set value, turn off the execution command in advance.
- The counter specified by the UDCNT2 instruction cannot be used by any other instruction. If another instruction uses it, normal counting is disabled.
- The UDCNT2 instruction can be used a maximum of five times in all running programs. The sixth or subsequent UDCNT1 instruction, if issued, causes no processing.

Operation error

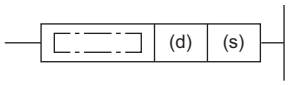
There is no operation error.

20.2 Special Timer Instructions

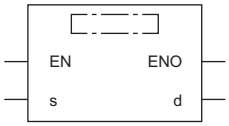
Teaching timer

TTMR

This instruction measures the on time of the measurement command in seconds, multiplies it by a multiplier, and stores the operation result.

Ladder	ST
	ENO:=TTMR(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
TTMR	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(d)	(d)+0: Device for storing the measurement value	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
	(d)+1: For controller system			
(s)	Multiplier of measurement value	0 to 2	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s)	—	○	○	○	○	—	—	○	○	—	—	—

■Control data

Operand: (d)			
Device	Description	Setting range	Set by
+0	Device for storing the measurement value	—	System
+1	For controller system	—	System

Processing details

- This instruction measures the on time of the execution command in seconds, multiplies it by the multiplier specified by (s), and stores the resultant value in the device specified by (d).
- When the execution command is turned on, the instruction clears the device specified by (d)+0, (d)+1.
- The table below lists the multipliers that can be specified by (s).

(s)	Multiplier
0	1
1	10
2	100

- When the value in the device specified by (s) is not in the range from 0 to 2, no processing is performed.

Point

- When executed, the TTMR instruction implements time measurement. Do not use the JMP instruction to skip the TTMR instruction. Otherwise, accurate measurement is disabled.
- Do not change the multiplier specified by (s) during execution of the TTMR instruction. Otherwise, accurate values cannot be determined.
- The device specified by (d)+1 is used by the system of the controller. Do not change the value. If the value is changed, an accurate resultant value is not stored in the device specified by (d).

Operation error

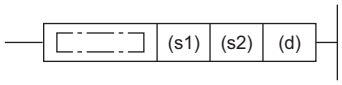
There is no operation error.

Special function timer

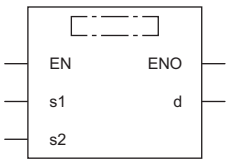
STMR

This instruction implements the following four types of timer output.

- Off delay timer output
- After-off one-shot timer output
- After-on one-shot timer output
- On delay + off delay timer output

Ladder	ST
	ENO:=STMR(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
STMR	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Timer device or timer type label	—	Device name	ANY16
(s2)	Set value	0 to 32767	16-bit signed binary	ANY16
(d)	(d)+0: Off delay timer output	—	Bit	ANYBIT_ARRAY (Number of elements: 4)
	(d)+1: After-off one-shot timer output			
	(d)+2: After-on one-shot timer output			
	(d)+3: On delay + off delay timer output			
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○*1	—	—	—	—	—	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	—	—	—	—	—	—	—	—	—	—

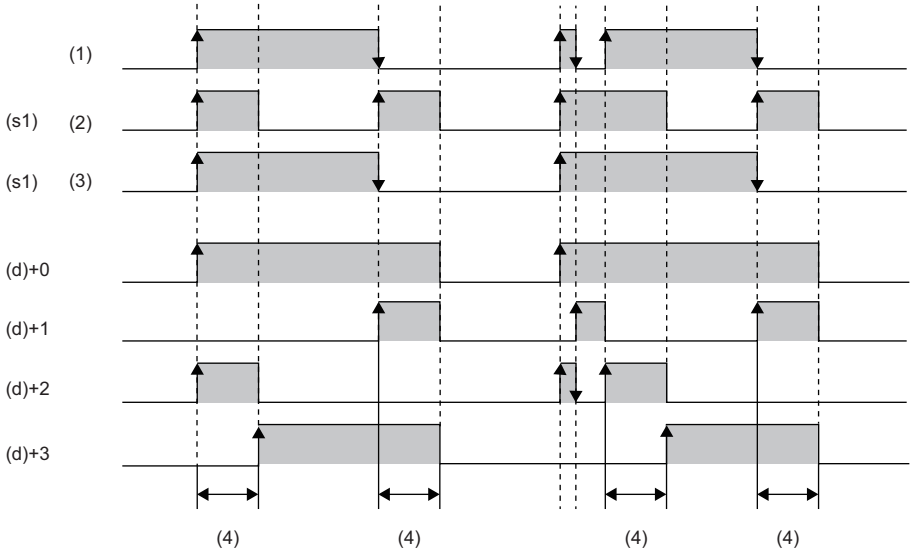
*1 Only T can be used.

Processing details

- This instruction uses four points from the device specified by (d) to implement four types of timer output.

Setting data		Description	
(d)	+0	Off delay timer output:	Turns on on the rising edge of the command of the STMR instruction, and turns off after a lapse of the time specified by (s2) after the falling edge of the command.
	+1	After-off one-shot timer output	Turns on on the falling edge of the command of the STMR instruction, and turns off after a lapse of the time specified by (s2).
	+2	After-on one-shot timer output	Turns on on the falling edge of the command of the STMR instruction, and turns off after a lapse of the time specified by (s2) or when the command of the instruction turns off.
	+3	On delay + off delay timer output	Turns on on the falling edge of the timer coil, and turns off after a lapse of the time specified by (s2) after the falling edge of the command of the STMR instruction.

- The coil of the timer specified by (s1) turns on, on the rising and falling edges of the command of the STMR instruction, to start measurement of the current value.
- The coil of the timer keeps measurement during the time specified by (s2) and turns off when the time is up.
- The coil of the timer is kept on even if the STMR instruction is turned off before time-up. Timer measurement is continued. When the STMR instruction is turned on again, the coil resets the current value to 0 and restarts measurement.
- The contact of the timer turns on, on the rising edge of the command of the STMR instruction, and turns off on the falling edge of the command after the coil of the timer falls. Users cannot use the contact of the timer because it is reserved for the system.



(1) Command for the STMR instruction
(2) Coil
(3) Contact
(4) Value specified by (s2)

- Measurement of the current value of the timer specified by the STMR instruction is executed regardless of whether the command of the STMR instruction is on or off. If the STMR instruction is skipped such as by the JMP instruction, normal measurement is not performed.
- The measurement unit of the timer specified by (d) is the same as that of the low-speed timer.
- A value from 0 to 32767 can be specified in (s2). If a value out of the range is specified, no processing is performed.
- Do not use the OUT instruction for the timer specified by (s1). If the same timer device or timer type label is used for the STMR and OUT instructions, normal operation is not performed.

Precautions

If there is an STMR instruction within the range for changing the ladder block online or writing data to the running programmable controller, the STMR instruction is executed.
For details, refer to the following.
User's manual for the controller used

Operation error

There is no operation error.

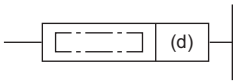
21 CLOCK

21.1 Clock Instructions

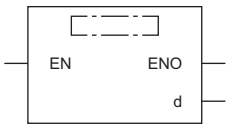
Reading clock data

DATERD(P)

Reads "year, month, day, hour, minute, second, and day of week" from the clock element of the controller.

Ladder	ST
	ENO:=DATERD(EN,d); ENO:=DATERDP(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DATERD	
DATERDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device for storing the clock data that has been read	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 7)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions read "year, month, day, hour, minute, second, and day of week" from the clock element of the controller, and store the read data in binary in the device specified by (d) and later.

(Data)	(d)	(d)+1	(d)+2	(d)+3	(d)+4	(d)+5	(d)+6
(Clock element)	**Year**	**Month**	**Day**	**Hour**	**Minute**	**Second**	**Day of week**
(Setting range)	1980 to 2079	1 to 12	1 to 31	0 to 23	0 to 59	0 to 59	0 to 6

- "Year" stored in the device specified by (d) is a 4-digit year.
- "Day of week" stored in the device specified by (d)+6 is a number from 0 to 6 corresponding to Sunday to Saturday.

Day of week	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Stored data	0	1	2	3	4	5	6

- Data is automatically corrected in leap years.

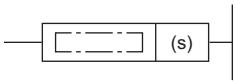
Operation error

There is no operation error.

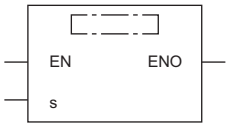
Writing clock data

DATEWR(P)

Writes the clock data stored in the specified device and later to the clock element of the controller.

Ladder	ST
	ENO:=DATEWR(EN,s); ENO:=DATEWRP(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
DATEWR	
DATEWRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the clock data to be written to the clock element is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 7)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions write the clock data stored in the specified device and later to the clock element of the controller.

(Data)	(d)	(d)+1	(d)+2	(d)+3	(d)+4	(d)+5	(d)+6
(Clock element)	Year	Month	Day	Hour	Minute	Second	Day of week
(Setting range)	1980 to 2079	1 to 12	1 to 31	0 to 23	0 to 59	0 to 59	0 to 6

- Set each data in binary.
- Set the year data in the range from 1980 to 2079 in the device specified by (s).
- Set the month data in the range from 1 to 12 in the device specified by (s)+1.
- Set the day data in the range from 1 to 31 in the device specified by (s)+2.
- Set the hour data in the range from 0 to 23 in the device specified by (s)+3. (Set in 24-hour format.)
- Set the minute data in the range from 0 to 59 in the device specified by (s)+4.
- Set the second data in the range from 0 to 59 in the device specified by (s)+5.
- Set the day of week in the range from 0 to 6 corresponding to Sunday to Saturday in the device specified by (s)+6.

Day of week	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Stored data	0	1	2	3	4	5	6

Operation error

Error code (SD0)	Description
3285H	Out-of-range data is set in (s) to (s)+6.
3295H	A time value less than an hour from the start time of the daylight-saving time is set in (s) to (s)+6.

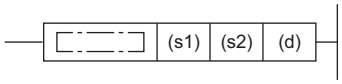
Point

When clock data is changed, "clock setting" (event code: 24000) is saved to the event history. Therefore, "clock setting" is saved to the event history when this instruction is executed.

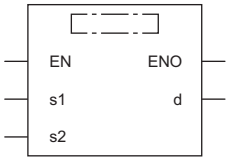
Adding clock data

DATE+(P)

These instructions add time data.

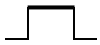
Ladder	ST
	ENO:=DATEPLUS(EN,s1,s2,d); ENO:=DATEPLUSP(EN,s1,s2,d);

FBD/LD



(□ is replaced by either of the following: DATEPLUS, DATEPLUSP.)

■Execution condition

Instruction	Execution condition
DATE+	
DATE+P	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the augend clock data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(s2)	Start device where the addend time (clock) data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(d)	Start device for storing the addition result time (clock) data	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions add the time data in the device specified by (s2) to the time data in the device specified by (s1), and store the addition result in the device number specified by (d) and later.

(s1)	hour	(0 to 23)	+	(s2)	hour	(0 to 23)	⇨	(d)	hour	(0 to 23)
(s1)+1	minute	(0 to 59)		(s2)+1	minute	(0 to 59)		(d)+1	minute	(0 to 59)
(s1)+2	second	(0 to 59)		(s2)+2	second	(0 to 59)		(d)+2	second	(0 to 59)

Ex.

7:48:10 is added to 6:32:40.

(s1)	6	+	(s2)	7	⇨	(d)	14
(s1)+1	32		(s2)+1	48		(d)+1	20
(s1)+2	40		(s2)+2	10		(d)+2	50

- If the time obtained as the result of addition exceeds 24 hours, 24 hours are subtracted from the resultant time to produce the operation result. For example, when 20:20:20 is added to 14:20:30, the operation result is 10:40:50 rather than 34:40:50.

(s1)	14	+	(s2)	20	⇨	(d)	10
(s1)+1	20		(s2)+1	20		(d)+1	40
(s1)+2	30		(s2)+2	20		(d)+2	50

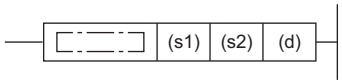
Operation error

Error code (SD0)	Description
3285H	The data in the device specified by (s1) or (s2) is out of range.

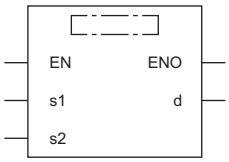
Subtracting clock data

DATE-(P)

These instructions subtract time data.

Ladder	ST
	ENO:=DATEMINUS(EN,s1,s2,d); ENO:=DATEMINUSP(EN,s1,s2,d);

FBD/LD



(□ is replaced by either of the following: DATEMINUS, DATEMINUSP.)

■Execution condition

Instruction	Execution condition
DATE-	
DATE-P	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where minuend clock data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(s2)	Start device where the subtrahend time (clock) data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(d)	Start device for storing the subtraction result time (clock) data	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions subtract the time data in the device specified by (s2) from the time data in the device specified by (s1), and store the subtraction result in the device number specified by (d) and later.

(s1)	hour	(0 to 23)	-	(s2)	hour	(0 to 23)	⇨	(d)	hour	(0 to 23)
(s1)+1	minute	(0 to 59)		(s2)+1	minute	(0 to 59)		(d)+1	minute	(0 to 59)
(s1)+2	second	(0 to 59)		(s2)+2	second	(0 to 59)		(d)+2	second	(0 to 59)

Ex.

3:50:10 is subtracted from 10:40:20.

(s1)	10	-	(s2)	3	⇨	(d)	6
(s1)+1	40		(s2)+1	50		(d)+1	50
(s1)+2	20		(s2)+2	10		(d)+2	10

- If the time obtained as the result of subtraction becomes a negative value, 24 hours are added to the resultant time to produce the operation result. For example, when 10:42:12 is subtracted from 4:50:32, the operation result is 18:8:20 rather than -6:8:20.

(s1)	4	-	(s2)	10	⇨	(d)	18
(s1)+1	50		(s2)+1	42		(d)+1	8
(s1)+2	32		(s2)+2	12		(d)+2	20

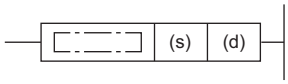
Operation error

Error code (SD0)	Description
3285H	The data in the device specified by (s1) or (s2) is out of range.

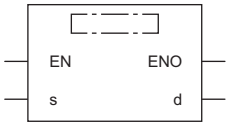
Converting time data from hour/minute/second to second

TIME2SEC(P)

These instructions convert time data (hour, minute, second) to second data.

Ladder	ST
	ENO:=TIME2SEC(EN,s,d); ENO:=TIME2SECP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
TIME2SEC	
TIME2SECP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the time data to be converted is stored	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(d)	Start device for storing the clock data after conversion	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

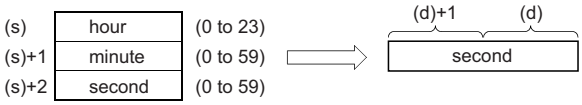
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○*1	○	○	○	○	○	○	○	—	—	—	—

*1 FX and FY cannot be used.

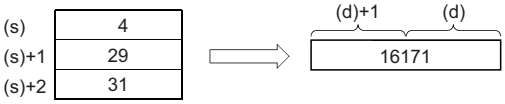
Processing details

- These instructions convert the time data stored in the device number specified by (s) and later to second data, and store the operation result in the device specified by (d).



Ex.

4:29:31 is specified in (s).



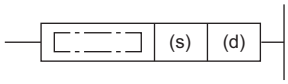
Operation error

Error code (SD0)	Description
3285H	The data in the device specified (s) is out of range.

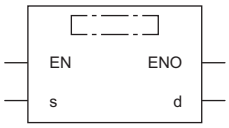
Converting time data from second to hour/minute/second

SEC2TIME(P)

These instructions convert second data to time data (hour, minute, second).

Ladder	ST
	ENO:=SEC2TIME(EN,s,d); ENO:=SEC2TIMEP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SEC2TIME	
SEC2TIMEP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the time data to be converted is stored	0 to 86399	32-bit signed binary	ANY32
(d)	Start device for storing the clock data after conversion	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

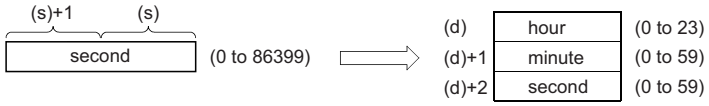
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

*1 FX and FY cannot be used.

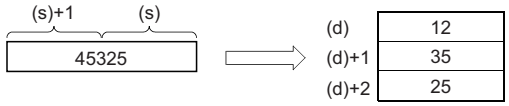
Processing details

- These instructions convert the seconds data stored in the device number specified by (s) and later to time data (hour, minute, second), and store the operation result in the device specified by (d) and later.



Ex.

45325 seconds are specified in (s).



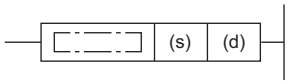
Operation error

Error code (SD0)	Description
3285H	The data in the device specified (s) is out of range.

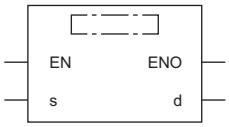
Converting date and time data from date and time to second

DATE2SEC(P)(_U)

These instructions convert date and time data into second.

Ladder	ST
	ENO:=DATE2SEC(EN,s,d); ENO:=DATE2SECP(EN,s,d); ENO:=DATE2SEC_U(EN,s,d); ENO:=DATE2SECP_U(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DATE2SEC DATE2SEC_U	
DATE2SECP DATE2SECP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)		Start device where the date and time data to be converted is stored	Refer to the function details.	16-bit signed binary	ANY16_ARRAY (Number of elements: 6)
(d)	DATE2SEC(P)	Start device for storing the converted second data	0 to 2145916799	32-bit signed binary	ANY32
	DATE2SEC(P)_U		0 to 3155759999	32-bit unsigned binary	ANY32_U
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

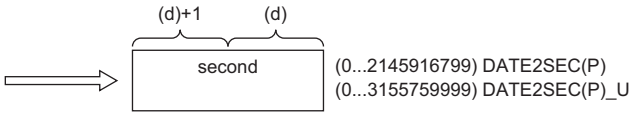
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○*1	○	○	○	○	○	○	○	—	—	—	—

*1 FX and FY cannot be used.

Processing details

- These instructions convert the date and time data stored in the device number specified by (s) and later into second data, and store the operation result in the device specified by (d). The starting point (0 second) is January 1, 2000 at 00:00:00.

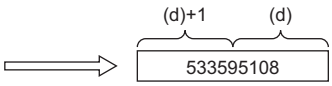
(s)	year	(2000...2067) DATE2SEC(P) (2000...2099) DATE2SEC(P)_U
(s)+1	month	(1...12)
(s)+2	day	(1...31)
(s)+3	hour	(0...23)
(s)+4	minute	(0...59)
(s)+5	second	(0...59)



- The year data range is 2000 to 2067 for DATE2SEC(P) and 2000 to 2099 for DATE2SEC(P)_U.

For example, the following shows when November 27, 2016 at 20:51:48 is specified.

(s)	2016
(s)+1	11
(s)+2	27
(s)+3	20
(s)+4	51
(s)+5	48



- Changing clock data (including the time zone and time adjustment by daylight-saving time) does not affect the operation of these instructions. For example, the operation result during the season when the daylight-saving time is applied is same as the result during the season when the daylight-saving time is not applied.

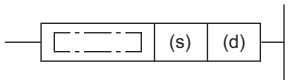
Operation error

Error code (SD0)	Description
3285H	The data in the device specified (s) is out of range.
	A non-existent date and time is specified. (Example: February 30, 2016)

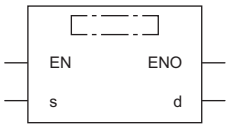
Converting date and time data from second to date and time

SEC2DATE(P)(_U)

These instructions convert second data into date and time data.

Ladder	ST
	ENO:=SEC2DATE(EN,s,d); ENO:=SEC2DATEP(EN,s,d); ENO:=SEC2DATE_U(EN,s,d); ENO:=SEC2DATEP_U(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SEC2DATE SEC2DATE_U	
SEC2DATEP SEC2DATEP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s)	SEC2DATE(P)	Start device where the second data to be converted is stored	0 to 2145916799	32-bit signed binary	ANY32
	SEC2DATE(P)_U		0 to 3155759999	32-bit unsigned binary	ANY32_U
(d)		Start device for storing the converted date and time data	Refer to the function details.	16-bit signed binary	ANY16_ARRAY (Number of elements: 7)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

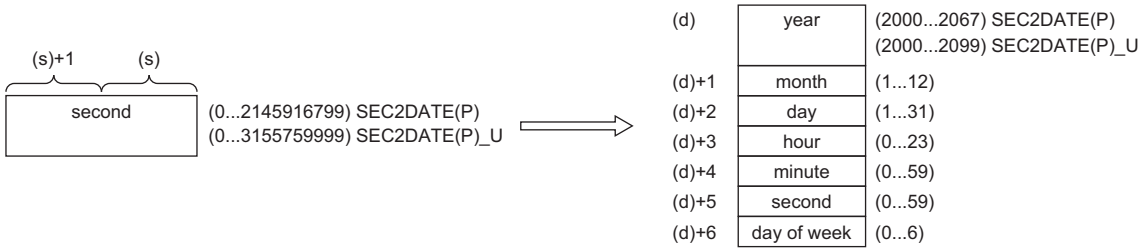
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

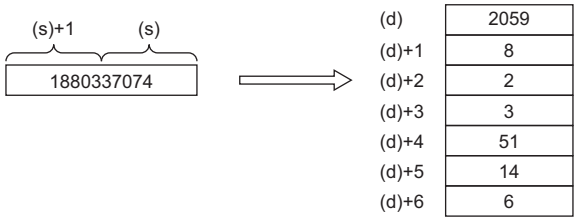
*1 FX and FY cannot be used.

Processing details

- These instructions convert the second data stored in the device number specified by (s) and later into date and time data, and store the operation result in the device specified by (d) and later. The starting point (0 second) is January 1, 2000 at 00:00:00.



For example, the following shows 1880337074 seconds are specified.



- Day of week stored in the device specified by (d)+6 is a number from "0 to 6" corresponding to "Sunday to Saturday".

Day of week	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Stored data	0	1	2	3	4	5	6

- Changing clock data (including the time zone and time adjustment by daylight-saving time) does not affect the operation of these instructions. For example, the operation result during the season when the daylight-saving time is applied is same as the result during the season when the daylight-saving time is not applied.

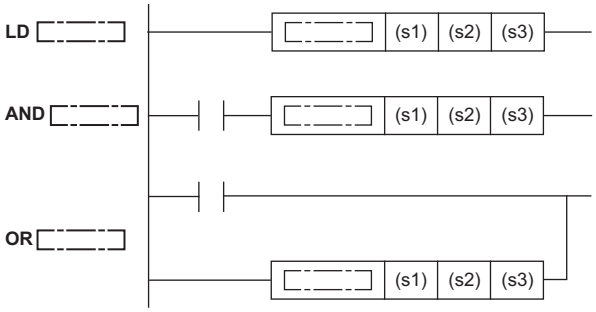
Operation error

Error code (SD0)	Description
3285H	The data in the device specified (s) is out of range.

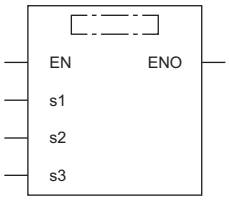
Comparing date data

LDDT□, ANDDT□, ORDT□

These instructions compare the specified date data, or compare the date data with the current date.

Ladder	ST
 (□ indicates DT=, DT<>, DT>, DT<=, DT<, or DT>=.)	ENO:=LDDT_□(EN,s1,s2,s3); ENO:=ANDDT_□(EN,s1,s2,s3); ENO:=ORDT_□(EN,s1,s2,s3); (□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)*1

FBD/LD



(□ is replaced by a combination of LDDT_, ANDDT_, or ORDT_ and EQ, NE, GT, LE, LT, or GE.)*1

*1 EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
LDDT□, ANDDT□, ORDT□	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the data to be compared is stored	—	16-bit signed binary	ANY_DT*1
(s2)	Start device where the data to be compared is stored	—	16-bit signed binary	ANY_DT*1
(s3)	Comparison target setting value or the number of comparison target data	0001H to 0007H, 8001H to 8007H	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

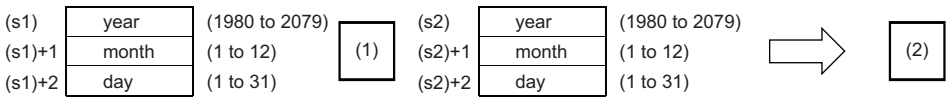
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions compare the date data in the devices specified by (s1) and (s2), or compare the date data in the device specified by (s1) with the current date. Set the comparison target by (s3).

• Comparing two specified date data

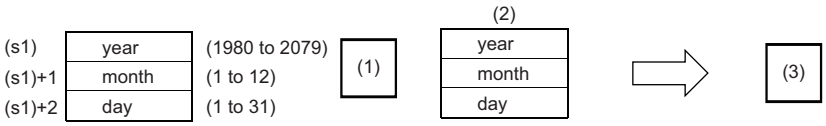
These instructions compare the date data in the device specified by (s1) with the date data in the device specified by (s2) in accordance with the conditions set by (s3). (Devices are used as a normally open contact.)



(1)Relational operator
(2)Comparison result

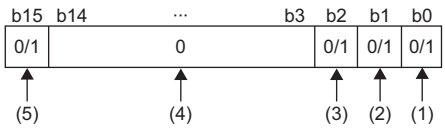
• Comparing the specified date data with the current date

These instructions compare the date data in the device specified by (s1) with the current date data in accordance with the conditions set by (s3). (Devices are used as a normally open contact.) The date data in the device specified by (s2) is regarded as dummy data and ignored.



(1)Relational operator
(2)Current date data
(3)Comparison result

- Set each data in binary.
- Set the year data in the range from 1980 to 2079 in the devices specified by (s1) and (s2).
- Set the month data in the range from 1 to 12 in the devices specified by (s1)+1 and (s2)+1.
- Set the date data in the range from 1 to 31 in the devices specified by (s1)+2 and (s2)+2.
- Set the following in (s3) as comparison target setting values. The following shows the bit configuration of (s3).



(1) Set "day" as comparison target.
(2) Set "month" as comparison target.
(3) Set "year" as comparison target.
(4) Set 0. If a value other than 0 is set, the operation result will be non-continuity.
(5) When bit 15 turns on (1), the data in the device specified by (s1) is compared with the current date in accordance with the conditions set in bits 0 to 2.

- When 0 is set to bits 0 to 2, the date data are not compared. When 1 is set, the entire date data (year, month, and day) are compared.
- When 0 is set to bit 15, the data in the device specified by (s1) and the date data in the device specified by (s2) are compared. When 1 is set, the data in the device specified by (s1) is compared with the current date. The date data in the device specified by (s2) is ignored.
- The following table lists processing details of each bit.

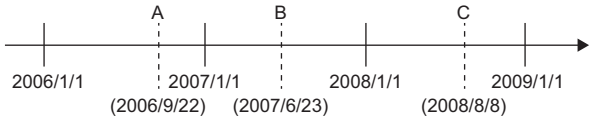
(s3) value when comparing two specified date data	(s3) value when comparing the specified date data with the current date	Comparison target	Description
0001H	8001H	Day	Only data in the device specified by (s1)+2 is compared.
0002H	8002H	Month	Only data in the device specified by (s1)+1 is compared.
0003H	8003H	Month, day	Data in the device areas specified by (s1)+1 and (s1)+2 are compared.
0004H	8004H	Year	Only data in the device specified by (s1) is compared.
0005H	8005H	Year, day	Data in the device areas specified by (s1) and (s1)+2 are compared.
0006H	8006H	Year, month	Data in the device areas specified by (s1) and (s1)+1 are compared.
0007H	8007H	Year, month, day	The entire date data in the device areas specified by (s1), (s1)+1, and (s1)+2 are compared.
Other than 0001H to 0007H, 8001H to 8007H		None	The entire date data in the device areas specified by (s1), (s1)+1, and (s1)+2 are not compared. (The operation result will be non-continuity.)

- If the comparison target data in the device are not recognized as date data, SM709 (DT/TM instruction improper data detection flag) turns on after the instruction is executed and the operation result will be non-continuity. Even if the data are not recognized as date data, SM709 does not turn on if the data are within the setting range. If the device areas specified by (s1) to (s1)+2 or (s2) to (s2)+2 exceed the assignment range of the relevant device/label, SM709 turns on after the instruction is executed and the operation result will be non-continuity as well. Once SM709 turns on, the on state is held until the controller is powered off or reset. Turn off SM709 as needed.
- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
DT=, EQ	(s1)=(s2)	Continuity state (ENO is on.)
DT<>, NE	(s1)≠(s2)	
DT>, GT	(s1)>(s2)	
DT<=, LE	(s1)≤(s2)	
DT<, LT	(s1)<(s2)	
DT>=, GE	(s1)≥(s2)	
DT=, EQ	(s1)≠(s2)	Non-continuity state (ENO is off.)
DT<>, NE	(s1)=(s2)	
DT>, GT	(s1)≤(s2)	
DT<=, LE	(s1)>(s2)	
DT<, LT	(s1)>≥(s2)	
DT>=, GE	(s1)<(s2)	

Ex.

The date data A, B, and C are compared.



- The following table lists the comparison operation results between A, B, and C. Even when the data are compared under the same conditions, the results differ depending on the comparison target data.

Comparison target data	Condition*1		
	A<B	B<C	A<C
Day	Continuity	Non-continuity	Non-continuity
Month	Non-continuity	Continuity	Non-continuity
Month, day	Non-continuity	Continuity	Non-continuity
Year	Continuity	Continuity	Continuity
Year, day	Continuity	Continuity	Continuity
Year, month	Continuity	Continuity	Continuity
Year, month, day	Continuity	Continuity	Continuity
None	Non-continuity	Non-continuity	Non-continuity

*1 In FBD/LD, ENO ON indicates continuity and ENO OFF indicates non-continuity.

- Even though the specified date does not exist, the comparison operation is performed in accordance with the conditions in the following table as long as the date data are within the valid range.
  - Date A: 2006/02/30 (Even though the date does not exist, this date can be set.)
  - Date B: 2007/03/29
  - Date C: 2008/02/31 (Even though the date does not exist, this date can be set.)

Comparison target data	Condition*2		
	A<B	B<C	A<C
Day	Non-continuity	Continuity	Continuity
Month	Continuity	Non-continuity	Non-continuity
Month, day	Continuity	Non-continuity	Continuity
Year	Continuity	Continuity	Continuity
Year, day	Continuity	Continuity	Continuity
Year, month	Continuity	Continuity	Continuity
Year, month, day	Continuity	Continuity	Continuity
None	Non-continuity	Non-continuity	Non-continuity

*2 In FBD/LD, ENO ON indicates continuity and ENO OFF indicates non-continuity.

- If the LDDT_☐ instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.
- If the ORDT_☐ instruction is used in the program written in FBD/LD and EN is set to TRUE, ENO turns on. EN will not be an execution condition.

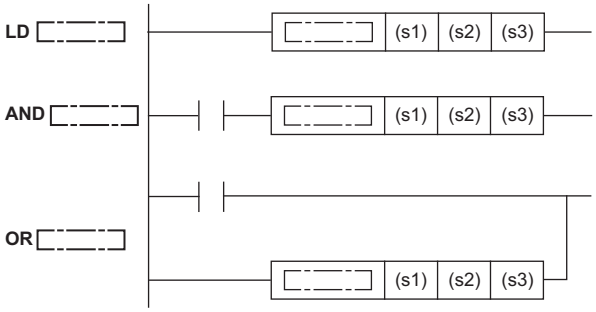
Operation error

There is no operation error.

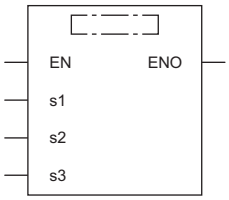
Comparing time data

LDTM□, ANDTM□, ORTM□

These instructions compare the specified time data, or compare the specified time data with the current time.

Ladder	ST
 (□ is replaced by TM=, TM<>, TM>, TM<=, TM<, or TM>=.)	ENO:=LDTM_□(EN,s1,s2,s3); ENO:=ANDTM_□(EN,s1,s2,s3); ENO:=ORTM_□(EN,s1,s2,s3); (□ is replaced by any of the following: EQ, NE, GT, LE, LT, GE.)*1

FBD/LD



(□ is replaced by a combination of LDTM_, ANDTM_, or ORTM_ and EQ, NE, GT, LE, LT, or GE.)*1

*1 EQ indicates =, NE indicates <>, GT indicates >, LE indicates <=, LT indicates <, and GE indicates >=.

■Execution condition

Instruction	Execution condition
LDTM□, ANDTM□, ORTM□	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the data to be compared is stored	—	16-bit signed binary	ANY_TM*1
(s2)	Start device where the data to be compared is stored	—	16-bit signed binary	ANY_TM*1
(s3)	Comparison target setting value or the number of comparison target data	0001H to 0007H, 8001H to 8007H	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

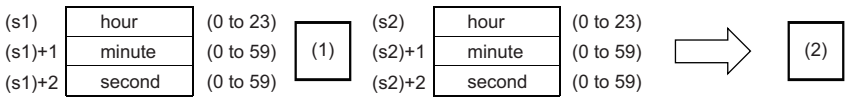
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions compare the time data in the device specified by (s1) and (s2), or compare the time data in the device specified by (s1) with the current time. Set the comparison target by (s3).

• Comparing two specified time data

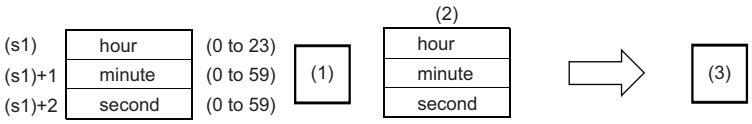
These instructions compare the time data in the device specified by (s1) with the time data (hour, minute, second) in the device specified by (s2) in accordance with the conditions set by (s3). (Devices are used as a normally open contact.)



(1)Relational operator
(2)Comparison result

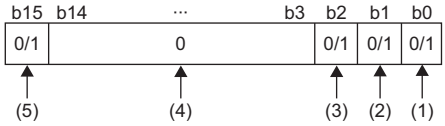
• Comparing the specified time data with the current time data

These instructions compare the time data in the device specified by (s1) with the current time data in accordance with the conditions set by (s3). (Devices are used as a normally open contact.) The time data in the device specified by (s2) is regarded as dummy data and ignored.



(1)Relational operator
(2)Current time data
(3)Comparison result

- Set each data in binary.
- Set the hour data in the range from 0 to 23 in 24-hour format in the devices specified by (s1) and (s2).
- Set the minute data in the range from 0 to 59 in the devices specified by (s1)+1 and (s2)+1.
- Set the second data in the range from 0 to 59 in the devices specified by (s1)+2 and (s2)+2.
- Set the following in (s3) as comparison target setting values. The following shows the bit configuration of (s3).



(1) Set "second" as comparison target.
(2) Set "minute" as comparison target.
(3) Set "hour" as comparison target.
(4) Set 0. If a value other than 0 is set, the operation result will be non-continuity.
(5) When bit 15 turns on (1), the data in the device specified by (s1) is compared with the current time in accordance with the conditions set in bits 0 to 2.

- When 0 is set to bits 0 to 2, the time data are not compared. When 1 is set, the comparison target time data (hour, minute, second) are compared.
- When 0 is set to bit 15, the data in the device specified by (s1) and the time data in the device specified by (s2) are compared. When 1 is set, the time data in the device specified by (s1) is compared with the current time. The time data in the device specified by (s2) is ignored.
- The following table lists processing details of each bit.

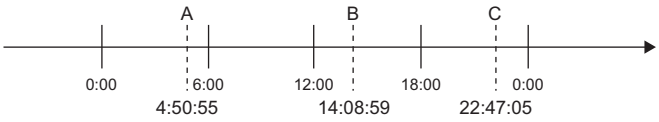
(s3) value when comparing two specified time data	(s3) value when comparing with current time data	Comparison target time	Description
0001H	8001H	Second	Only data in the device specified by (s1)+2 is compared.
0002H	8002H	Minute	Only data in the device specified by (s1)+1 is compared.
0003H	8003H	Minute, second	Data in the device areas specified by (s1)+1 and (s1)+2 are compared.
0004H	8004H	Hour	Only data in the device specified by (s1) is compared.
0005H	8005H	Hour, second	Data in the device areas specified by (s1) and (s1)+2 are compared.
0006H	8006H	Hour, minute	Data in the device areas specified by (s1) and (s1)+1 are compared.
0007H	8007H	Hour, minute, second	The entire date data in the device areas specified by (s1), (s1)+1, and (s1)+2 are compared.
Other than 0001H to 0007H, 8001H to 8007H		None	The entire date data in the device areas specified by (s1), (s1)+1, and (s1)+2 are not compared. (The operation result will be non-continuity.)

- If the comparison target data in the device are not recognized as time data, SM709 (DT/TM instruction improper data detection flag) turns on after the instruction is executed and the operation result will be non-continuity (ENO OFF). If the device areas specified by (s1) to (s1)+2 or (s2) to (s2)+2 exceed the assignment range of the relevant device/label, SM709 turns on after the instruction is executed and the operation result will be non-continuity (ENO OFF) as well. Once SM709 turns on, the on state is held until the controller is powered off or reset. Turn off SM709 as needed.
- The following table lists the comparison operation results of each instruction.

Instruction symbol (ladder, FBD/LD)	Condition	Result
TM=, EQ	(s1)=(s2)	Continuity state (ENO is on.)
TM<>, NE	(s1)≠(s2)	
TM>, GT	(s1)>(s2)	
TM<=, LE	(s1)≤(s2)	
TM<, LT	(s1)<(s2)	
TM>=, GE	(s1)≥(s2)	
TM=, EQ	(s1)≠(s2)	Non-continuity state (ENO is off.)
TM<>, NE	(s1)=(s2)	
TM>, GT	(s1)≤(s2)	
TM<=, LE	(s1)>(s2)	
TM<, LT	(s1)≥(s2)	
TM>=, GE	(s1)<(s2)	

Ex.

The time data A, B, and C are compared.



- The following table lists the comparison operation results between A, B, and C. Even when the data are compared under the same conditions, the results differ depending on the comparison target data.

Comparison target data	Condition*1		
	A<B	B<C	A<C
Second	Continuity	Non-continuity	Non-continuity
Minute	Non-continuity	Continuity	Non-continuity
Minute, second	Non-continuity	Continuity	Non-continuity
Hour	Continuity	Continuity	Continuity
Hour, second	Continuity	Continuity	Continuity
Hour, minute	Continuity	Continuity	Continuity
Hour, minute, second	Continuity	Continuity	Continuity
None	Non-continuity	Non-continuity	Non-continuity

*1 In FBD/LD, ENO ON indicates continuity and ENO OFF indicates non-continuity.

- If the LDTM_☐ instruction is used in the program written in FBD/LD, use a left rail or a variable/constant which is always on for EN.
- If the ORTM_☐ instruction is used in the program written in FBD/LD and EN is set to TRUE, ENO turns on. EN will not be an execution condition.

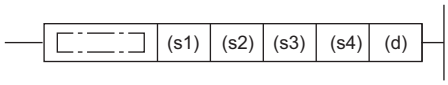
Operation error

There is no operation error.

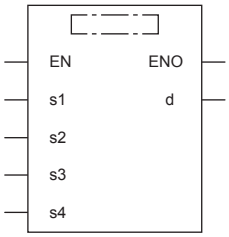
Outputting a comparison result of time data

TCMP(P)

These instructions compare the time data to be compared that is specified by (s1), (s2), and (s3) with the time data specified by (s4), and according to the result (small, match, or large), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST
	ENO:=TCMP(EN,s1,s2,s3,s4,d); ENO:=TCMPP(EN,s1,s2,s3,s4,d);

FBD/LD



■Execution condition

Instruction	Execution condition
TCMP	
TCMPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device for storing the time data (hour) to be compared	0 to 23	16-bit signed binary	ANY16
(s2)	Start device for storing the time data (minute) to be compared	0 to 59	16-bit signed binary	ANY16
(s3)	Start device for storing the time data (second) to be compared	0 to 59	16-bit signed binary	ANY16
(s4)	Start device for storing the time data (hour, minute, second) to be compared	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(d)	The start device where the comparison result is stored	—	Bit	ANYBIT_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

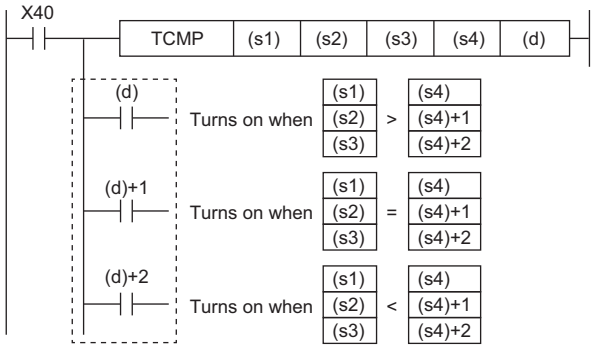
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(s2)	○*1	○	○	○	○	—	—	○	○	—	—	—
(s3)	○*1	○	○	○	○	—	—	○	○	—	—	—
(s4)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○*2	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.
*2 T, ST, and C cannot be used.

Processing details

- These instructions compare the time data to be compared that is specified by (s1), (s2), and (s3) with the time data specified by (s4), and according to the result (small, match, or large), (d), (d)+1, or (d)+2 is turned on.



- (s1): hour, (s2): minute, (s3): second
- (s4): hour, (s4)+1: minute, (s4)+2: second

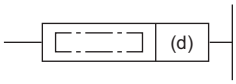
Operation error

Error code (SD0)	Description
3285H	The value specified by (s1) and (s4) is outside the following range. 0 to 23
	The value specified by (s2), (s3), (s4)+1, and (s4)+2 is outside the following range. 0 to 59

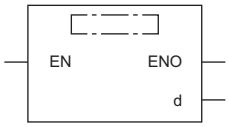
Reading expansion clock data

S(P).DATERD

These instructions read clock data including millisecond from the clock elements in the controller.

Ladder	ST
	ENO:=S_DATERD(EN,d); ENO:=SP_DATERD(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
S.DATERD	
SP.DATERD	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Start device for storing the clock data that has been read	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 8)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions read "year, month, day, hour, minute, second, day of week, and 1/1000 second" from the clock element of the controller, and store the read data in binary in the device specified by (d) and later.

(Data)	(d)	(d)+1	(d)+2	(d)+3	(d)+4	(d)+5	(d)+6	(d)+7
(Clock element)	Year	Month	Day	Hour	Minute	Second	Day of week	Millisecond
(Setting range)	1980 to 2079	1 to 12	1 to 31	0 to 23	0 to 59	0 to 59	0 to 6	0 to 999

- "Year" stored in the device specified by (d) is a 4-digit year.
- "Day of week" stored in the device specified by (d)+6 is a number from 0 to 6 corresponding to Sunday to Saturday.

Day of week	Day	Month	Tuesday	Wednesday	Thursday	Friday	Saturday
Stored data	0	1	2	3	4	5	6

- Data is automatically corrected in leap years.

Precautions

- S(P).DATERD instruction reads clock data and store it in the device even when incorrect click data is set in the controller. (Example: February 30) When setting clock data with the DATEWR(P) instruction or engineering tool, be careful not to set incorrect clock data.
- When millisecond clock data is read, the maximum error is 2ms. (This error means the difference between the data stored in clock elements in the controller and the data read by the S(P).DATERD instruction.)

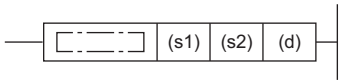
Operation error

There is no operation error.

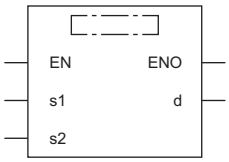
Adding expansion clock data

S(P).DATE+

These instructions add time data.

Ladder	ST
	ENO:=S_DATEPLUS(EN,s1,s2,d); ENO:=SP_DATEPLUS(EN,s1,s2,d);

FBD/LD



(□ is replaced by either of the following: S_DATEPLUS, SP_DATEPLUS.)

■Execution condition

Instruction	Execution condition
S.DATE+	
SP.DATE+	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where the augend clock data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
(s2)	Start device where the addend time (clock) data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
(d)	Start device for storing the addition result time (clock) data	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions add the time data in the device specified by (s2) to the time data in the device specified by (s1), and store the addition result in the device number specified by (d) and later.

(s1)	hour	(0 to 23)		(s2)	hour	(0 to 23)		(d)	hour	(0 to 23)
(s1)+1	minute	(0 to 59)		(s2)+1	minute	(0 to 59)		(d)+1	minute	(0 to 59)
(s1)+2	second	(0 to 59)	+	(s2)+2	second	(0 to 59)	⇨	(d)+2	second	(0 to 59)
(s1)+3	–			(s2)+3	–			(d)+3	–	
(s1)+4	1/1000second	(0 to 999)		(s2)+4	1/1000second	(0 to 999)		(d)+4	1/1000second	(0 to 999)

Ex.

7:48:10:500 is added to 6:32:40:875.

(s1)	6		(s2)	7		(d)	14
(s1)+1	32		(s2)+1	48		(d)+1	20
(s1)+2	40	+	(s2)+2	10	⇨	(d)+2	51
(s1)+3	–		(s2)+3	–		(d)+3	–
(s1)+4	875		(s2)+4	500		(d)+4	375

- If the time obtained as the result of addition exceeds 24 hours, 24 hours are subtracted from the resultant time to produce the operation result. For example, when 20:20:20:500 is added to 14:20:30:875, the operation result is 10:40:51:375 rather than 34:40:51:375.

(s1)	14		(s2)	20		(d)	10
(s1)+1	20		(s2)+1	20		(d)+1	40
(s1)+2	30	+	(s2)+2	20	⇨	(d)+2	51
(s1)+3	–		(s2)+3	–		(d)+3	–
(s1)+4	875		(s2)+4	500		(d)+4	375

Point

- Devices (s1)+3, (s2)+3, and (d)+3 are not used for operation.
- The clock data that has been read by the S(P).DATERD instruction can be added without conversion.

(d)	Hour
(d)+1	Minute
(d)+2	Second
(d)+3	Day of week
(d)+4	Millisecond

When clock data is read by the S(P).DATERD instruction, "day of week" is inserted between "second" and "millisecond".
If the S(P).DATE+ instruction is used to read clock data, the data can be directly used for addition since it does not perform calculation for the day of week.

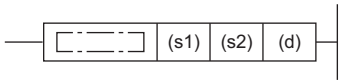
Operation error

Error code (SD0)	Description
3285H	The data in the device specified by (s1) or (s2) is out of range.

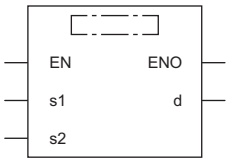
Subtracting expansion clock data

S(P).DATE-

These instructions subtract time data.

Ladder	ST
	ENO:=S_DATEMINUS(EN,s1,s2,d); ENO:=SP_DATEMINUS(EN,s1,s2,d);

FBD/LD



(☐ is replaced by either of the following: S_DATEMINUS, SP_DATEMINUS.)

■Execution condition

Instruction	Execution condition
S.DATE-	
SP.DATE-	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where minuend clock data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
(s2)	Start device where the subtrahend time (clock) data is stored	Refer to "Processing details".	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
(d)	Start device for storing the subtraction result time (clock) data	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions subtract the time data in the device specified by (s2) from the time data in the device specified by (s1), and store the subtraction result in the device number specified by (d) and later.

(s1)	hour	(0 to 23)	(s2)	hour	(0 to 23)	(d)	hour	(0 to 23)
(s1)+1	minute	(0 to 59)	(s2)+1	minute	(0 to 59)	(d)+1	minute	(0 to 59)
(s1)+2	second	(0 to 59)	(s2)+2	second	(0 to 59)	(d)+2	second	(0 to 59)
(s1)+3	–		(s2)+3	–		(d)+3	–	
(s1)+4	1/1000second	(0 to 999)	(s2)+4	1/1000second	(0 to 999)	(d)+4	1/1000second	(0 to 999)

(s1) − (s2) ⇨ (d)

Ex.

3:50:10:500 is subtracted from 10:40:20:875.

(s1)	10		(s2)	3		(d)	6	
(s1)+1	40		(s2)+1	50		(d)+1	50	
(s1)+2	20	-	(s2)+2	10	⇨	(d)+2	10	
(s1)+3	–		(s2)+3	–		(d)+3	–	
(s1)+4	875		(s2)+4	500		(d)+4	375	

- If the time obtained as the result of subtraction becomes a negative value, 24 hours are added to the resultant time to produce the operation result. For example, when 10:42:12:500 is subtracted from 4:50:32:875, the operation result is 18:8:20:375 rather than -6:8:20:375.

(s1)	4		(s2)	10		(d)	18	
(s1)+1	50		(s2)+1	42		(d)+1	8	
(s1)+2	32	-	(s2)+2	12	⇨	(d)+2	20	
(s1)+3	–		(s2)+3	–		(d)+3	–	
(s1)+4	875		(s2)+4	500		(d)+4	375	

Point

- Devices (s1)+3, (s2)+3, and (d)+3 are not used for operation.
- The clock data that has been read by the S(P).DATERD instruction can be subtracted without conversion.

(d)	Hour
(d)+1	Minute
(d)+2	Second
(d)+3	Day of week
(d)+4	Millisecond

When clock data is read by the S(P).DATERD instruction, "day of week" is inserted between "second" and "millisecond".
If the S(P).DATE- instruction is used to read clock data, the data can be directly used for subtraction since it does not perform calculation for the day of week.

Operation error

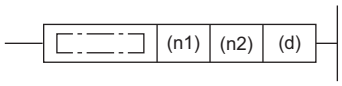
Error code (SD0)	Description
3285H	The data in the device specified by (s1) or (s2) is out of range.

21.2 Timing Check Instructions

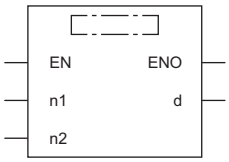
Generating timing pulses

DUTY

This instruction turns on the user timing clock for the specified number of scans and off for the specified number of scans.

Ladder	ST
	ENO:=DUTY(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DUTY	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(n1)	Number of scans during which the clock is turned on	0 to 65535	16-bit unsigned binary	ANY16
(n2)	Number of scans during which the clock is turned off	0 to 65535	16-bit unsigned binary	ANY16
(d)	Special relay device number of user timing clock to be operated	SM420 to SM424	Bit	ANY_BOOL*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to SM420 to SM424 can be used.

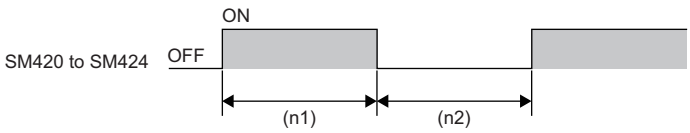
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(n1)	○	○	○	○	○	—	—	○	○	—	—	—
(n2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○*1	—	—	—	—	—	—	○	—	—	—	—

*1 Only SM420 to SM424 can be used.

Processing details

- This instruction turns on SM420 to SM424 in the device specified by (d) for the number of scans specified by (n1) and turns it off for the number of scans specified by (n2).



(n1): (n1) scans
(n2): (n2) scans

- The scan execution type program uses SM420 to SM424.
- When 0 is specified in (n1) and (n2) is equal to or greater than 0, SM420 to SM424 stay off. When (n1) is greater than 0 and (n2) is 0, SM420 to SM424 stay on.
- When the DUTY instruction is executed, the data specified by (n1), (n2), and (d) is stored in the system, and the timing pulses are turned on or off by the END processing.

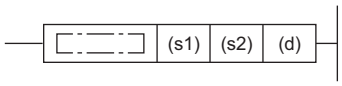
Operation error

Error code (SD0)	Description
2820H	The device other than SM420 to SM424 is specified by (d).

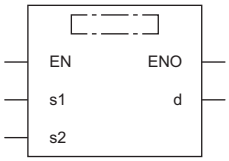
Measuring time of the specified data

TIMCHK

This instruction measures the on time of the device and, if the on time has continued as specified or longer, turns on the specified device.

Ladder	ST
	ENO:=TIMCHK(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
TIMCHK	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Device for storing the current value measured (unit: 100ms)	—	16-bit signed binary	ANY16
(s2)	Set value for measurement or the device where the set value for measurement is stored (unit: 100ms)	0 to 32767	16-bit signed binary	ANY16
(d)	Device to be turned on at time-up	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—

Processing details

- This instruction measures the on time of the device and, if the on time has continued as specified in the device specified by (s2) or longer, turns on the device specified by (d).
- The current value in the device specified by (s1) is cleared to 0 and the device specified by (d) is turned off on the rising edge of the execution command. The current value in the device specified by (s1) and the on state of the device specified by (d) are retained even after the execution command turns off.
- The current value measured is stored in units of 100ms. Set the measurement time in increments of 100ms.
- If 0 is specified in (s2), the current value in the device specified by (s1) is cleared to 0 and the device specified by (d) is turned off on the rising edge of the execution command.
- If a number other than 0 to 32767 is specified in (s2), (d) is turned on at the next scan after the execution command turns on.

Operation error

There is no operation error.

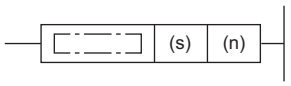
22 MODULE ACCESS

22.1 Module Access Instructions

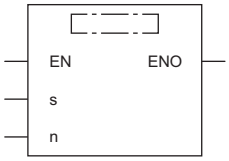
Performing I/O refresh

RFS(P)

These instructions refresh the n points of data from the specified device, and import external inputs or output data to the output module.

Ladder	ST
	ENO:=RFS(EN,s,n); ENO:=RFSP(EN,s,n);

FBD/LD



■Execution condition

Instruction	Execution condition
RFS	
RFSP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Start device to be refreshed	—	Bit	ANY_BOOL*1
(n)	Number of refreshed points	1 to 4096	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to devices (X, Y) can be used.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	—	—	—	—	—	—	—	—	—	—	—
(n)	—	○	○	○	○	—	—	○	○	—	—	—

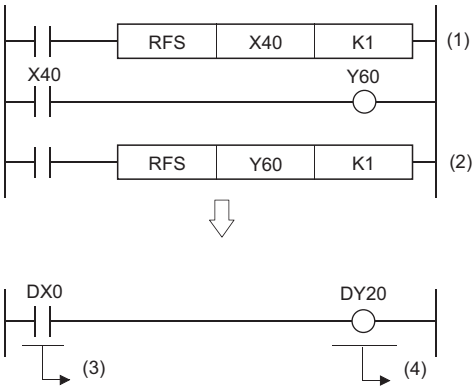
*1 Only X and Y can be used.

Processing details

- This instruction refreshes only the relevant device during one scan, and imports external inputs or outputs data to the output module.
- The instruction imports external inputs and outputs data to the outside altogether only after execution of the END instruction of the program, and therefore cannot output pulse signals to the outside during one scan. When executed, the I/O refresh instruction forcibly refreshes the relevant input (X) and output (Y) during program execution and therefore can output pulse signals to the outside during one scan.
- To refresh the input (X) or output (Y) in units of points, use the direct access input (DX) or direct access output (DY).

Ex.

When a program using the RFS instruction is changed to a program using direct access input/output



(1)Refresh X40.
(2)Refresh Y60.
(3)Direct access input
(4)Direct access output

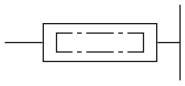
Operation error

Error code (SD0)	Description
2820H	The range of (n) points from the device specified by (s) exceeds the range of the proximal I/O.

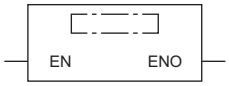
Selecting refresh to be performed

COM(P)

These instructions perform I/O refresh, link refresh of the network module, and device/label access service processing.

Ladder	ST
	ENO:=COM(EN); ENO:=COMP(EN);

FBD/LD



■Execution condition

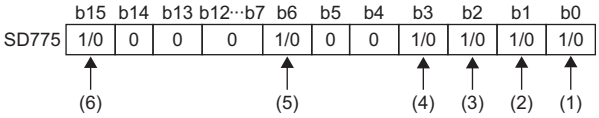
Instruction	Execution condition
COM	
COMP	

Processing details

- The COM(P) instructions are used to perform processing such as I/O refresh at any time during execution of the sequence program.
- The processing performed by the COM(P) instruction includes the following.
  - I/O refresh
  - Link refresh of the CC-Link module
  - Link refresh of the CC-Link IE Controller Network module
  - Link refresh of the CC-Link IE Field Network module
  - Intelligent function module refresh
  - Device/label access service processing (communication with the engineering tool, GOT, or other external devices)
- When SM775 is turned off, every processing except I/O refresh is performed.

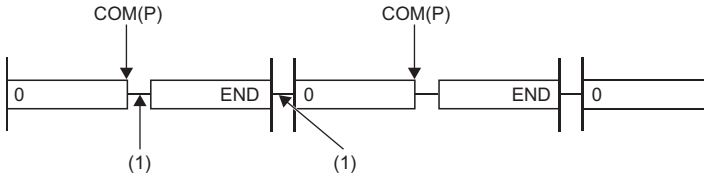
Description	SM775 is off	SM775 is on
I/O refresh	Non-execution	Execution or non-execution can be selected.
Link refresh of the CC-Link module	Execution	
Link refresh of the CC-Link IE Controller Network module		
Link refresh of the CC-Link IE Field Network module		
Intelligent function module refresh		
Device/label access service processing (communication with the engineering tool, GOT, or other external devices)		

- Select execution or non-execution for b0 to b3 and b15 of SD775 (Selection of refresh processing during the COM instruction execution), and then turn on SM775.



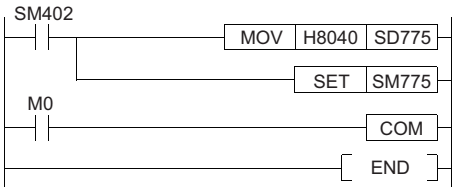
(1) I/O refresh
(2) Link refresh of the CC-Link module
(3) Link refresh of the CC-Link IE Controller Network module
(4) Intelligent function module refresh
(5) Link refresh of the CC-Link IE Field Network module
(6) Device/label access service processing (communications with the engineering tool, GOT, or other external devices)

- When executed, the COM(P) instruction performs the specified refresh processing.



(1) Specified processing

- In the following program example, link refresh of the CC-Link IE Field Network module is executed when M0 turns on.



Precautions

- The COM(P) instruction can be used as many times as needed in the program. Note, however, that the scan time of the program is extended by the time of the processing selected by SD775.
- Interrupts are enabled during execution of the COM(P) instruction. If refresh data is used by an interrupt program, data inconsistency may occur.
- If device/label access service processing is performed by the COM(P) instruction while an Ethernet device is connected to the Ethernet port, the processing time of the instruction may be extended.

Point

The COM(P) instruction cannot be used in the interrupt program.

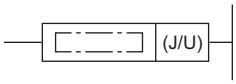
Operation error

There is no operation error.

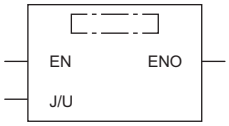
Performing module refresh

S(P).ZCOM

These instructions perform refresh processing for the specified module.

Ladder	ST
	ENO:=S_ZCOM(EN,J); ENO:=S_ZCOM(EN,U); ENO:=SP_ZCOM(EN,J); ENO:=SP_ZCOM(EN,U);

FBD/LD



■Execution condition

Instruction	Execution condition
S.ZCOM	
SP.ZCOM	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(J)*1	Own station network number	1 to 239	Device name	ANY16
(U)*1	Start I/O number (first three digits in four-digit hexadecimal representation) of a module	0H to FFH		
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 For the CC-Link IE TSN Plus master/local module, link refresh of the RJ71GN11-EIP (CC-Link IE TSN part) and refresh of the RJ71GN11-EIP (EtherNet/IP part) are performed when (U) is specified, while link refresh of the RJ71GN11-EIP (CC-Link IE TSN part) alone is performed when (J) is specified. (In both cases, these refreshes are performed when refresh settings are configured for the module.)

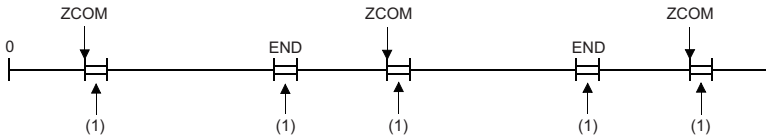
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	(J/U)
(J/U)	—	—	—	—	—	—	—	—	—	—	—	○

- The S(P).ZCOM instructions are used to perform refresh at any time during execution of the sequence program. The following lists the targets of refresh by the S(P).ZCOM instructions.
  - Link refresh of the CC-Link IE TSN master/local module (when refresh settings are configured)
  - Link refresh of the CC-Link IE Controller Network module (when refresh settings are configured)
  - Link refresh of the CC-Link IE Field Network master/local module (in refresh setting)
  - Link refresh of the CC-Link module (when refresh settings are configured)
  - Intelligent function module refresh (when a refresh parameter is specified)

Processing details

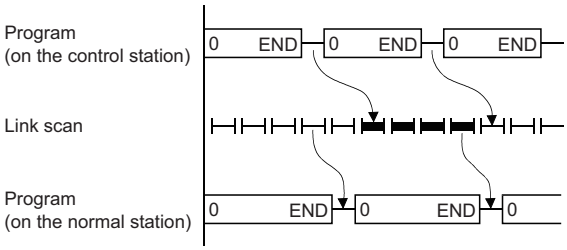
- When executed, the S(P).ZCOM instruction temporarily stops sequence program processing by the controller and perform refresh processing for the module specified by (J/U).



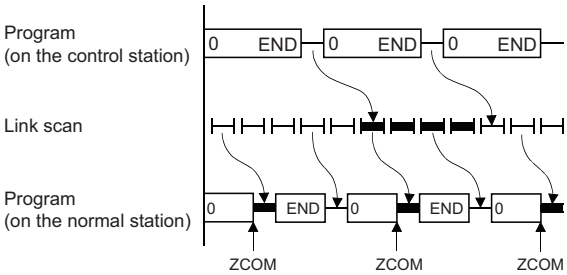
(1) Refresh processing

- The following is applicable when refresh processing of the CC-Link IE Controller Network module is performed.
  - When the scan time of the sequence program of the host station is longer than that of another station, the S(P).ZCOM instruction is used to ensure the import of data from the other station.

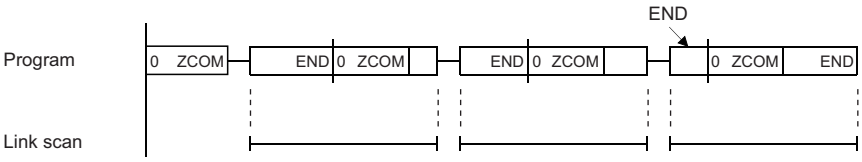
[When the S(P).ZCOM instruction is not used]



[When the S(P).ZCOM instruction is used]



- When the link scan time is longer than the sequence program scan time, using the S(P).ZCOM instruction will not make data communication faster.



Precautions

- The S(P).ZCOM instruction can be used as many times as needed in the program. Note, however, that the scan time of the program is extended by the refresh time.
- Interrupts are enabled during execution of the S(P).ZCOM instruction. If refresh data is used by an interrupt program, data separation may occur.

Operation error

Error code (SD0)	Description
2800H	The specified start I/O number is out of the range, 0 to FFH.
2801H	No module exists at the position specified by the start I/O number.
2804H	The network number set to (J) is out of the range, 1 to 239.
2805H	The specified network number is not connected to the host station.

Point

- The S(P).ZCOM instruction cannot be used in the interrupt program.
- To communicate only with external devices, use the COM(P) instruction.

Reading 1-word/2-word data from another module (16-bit specification)

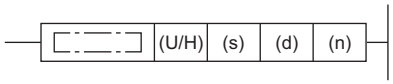
FROM(P), DFROM(P)

• FROM(P):

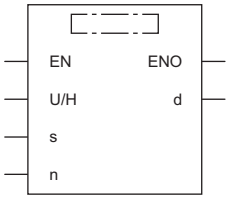
This instruction reads n words of data from the buffer memory address in the specified module.

• DFROM(P):

This instruction reads n×2 words of data from the buffer memory address in the specified module.

Ladder	ST
	ENO:=FROM(EN,U/H,s,n,d); ENO:=FROMP(EN,U/H,s,n,d); ENO:=DFROM(EN,U/H,s,n,d); ENO:=DFROMP(EN,U/H,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FROM DFROM	
FROMP DFROMP	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U/H)		Start I/O number (first three digits in four-digit hexadecimal representation) of a module and controller	0H to FFH, 3E0H	16-bit unsigned binary	ANY16
(s)		Start address of buffer memory or CPU built-in memory containing the data to be read	0 to 65535	16-bit unsigned binary	ANY16
(d)	FROM(P)	Start device for storing the data that has been read	—	16-bit signed binary	ANY16*1
	DFROM(P)			32-bit signed binary	ANY32*1
(n)		Number of read data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

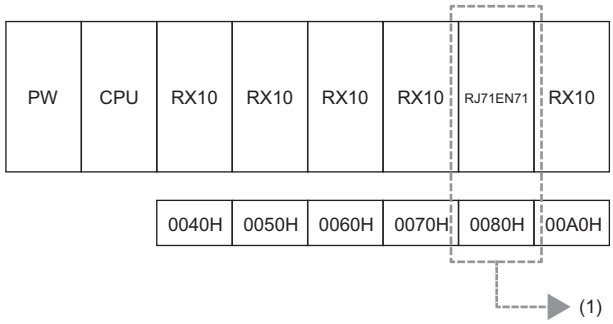
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U/H)	○	○	○	○	○	—	—	○	○	—	—	○
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	○	—	—	○*1	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

*1 Only the DFROM(P) instruction can be used.

Processing details

- For (U/H), specify the start I/O number of a module or controller with upper 3 digits when it is represented by 4 hexadecimal digits.



(1) Specify K8 or H8 as the start I/O number of the read-target module.

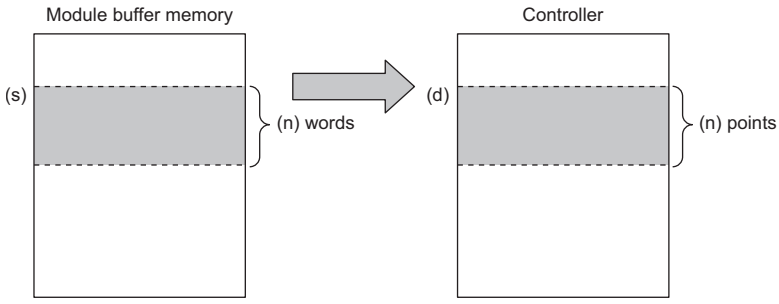
Specify the start I/O numbers of the controller as shown in the table below.

Controller	Start I/O number
Controller No.1	3E0H

■FROM(P)

- This instruction reads (n) words of data from the buffer memory address specified by (s) in the module specified by (U/H), and stores in the device specified by (d) and later.

• Reading word data from module

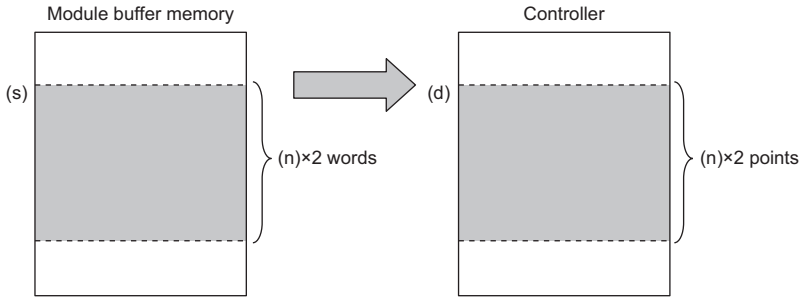


- If the read data (n) is 0, no processing is performed.
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

■DFROM(P)

- This instruction reads (n)×2 words of data from the buffer memory address specified by (s) in the module specified by (U/H), and stores in the device specified by (d) and later.

• Reading double word data from module



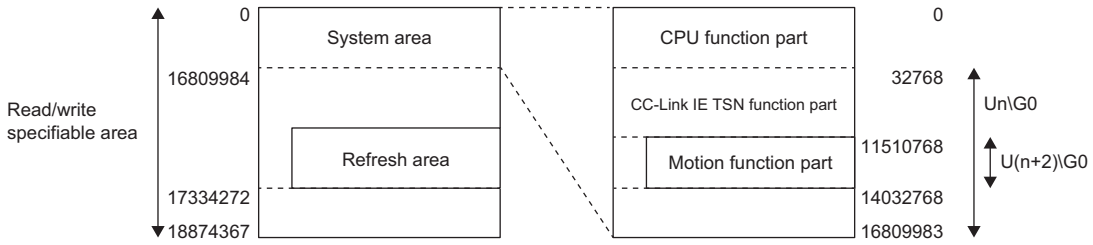
- If the read data (n) is 0, no processing is performed.
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

Operation error

Error code (SD0)	Description
2820H	The module with the I/O number specified by (U) does not have buffer memory.
2823H	The module with the I/O number specified by (H) does not have buffer memory.
	The address specified by (s) is outside the range of buffer memory.
	The (n) points of data starting from the address specified by (s) are outside the range of buffer memory. (FROM(P) instruction)
	The 2×(n) points of data starting from the address specified by (s) are outside the range of buffer memory. (DFROM(P) instruction)

Point

- Module data can also be read using the module access device. (User's manual for the controller used)
- If refresh settings are not made for the refresh area of the read/write enabled area in the CPU buffer memory, the area can be used as a read/write specifiable area. Even when refresh settings are made, the area can be used as a read/write specifiable area in the reference send range and later. If the start I/O number Un\G0 or U(n+2)\G0 of the CC-Link IE TSN function part or motion function part is specified in the operand (U/H), the following range can be accessed.

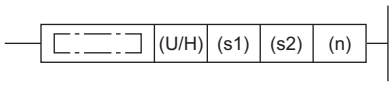


- A CPU buffer memory access device can be used to read data from the CPU buffer memory. (User's manual for the controller used)
- The FROM(P) and DFROM(P) instructions can read data from the buffer memory address with a capacity of 64K bytes or less. To read data from the buffer memory address with a capacity exceeding 64K bytes, use the FROMD(P) or DFROMD(P) instruction. (Page 1037 FROMD(P), DFROMD(P))

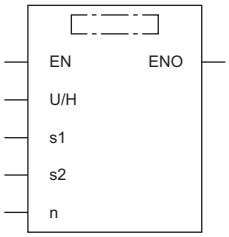
Writing 1-word/2-word data to a module (16-bit specification)

TO(P), DTO(P)

- TO(P): This instruction writes the n points of data from the specified device to the buffer memory in the module/host controller.
- DTO(P): This instruction writes the n×2 points of data from the specified device to the buffer memory in the module/host controller.

Ladder	ST
	ENO:=TO(EN,U/H,s1,s2,n); ENO:=TOP(EN,U/H,s1,s2,n); ENO:=DTO(EN,U/H,s1,s2,n); ENO:=DTOP(EN,U/H,s1,s2,n);

FBD/LD



■Execution condition

Instruction	Execution condition
TO DTO	
TOP DTOP	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U/H)		Start I/O number (first three digits in four-digit hexadecimal representation) of a module and controller	0H to FFH, 3E0H	16-bit unsigned binary	ANY16
(s1)		Start address of buffer memory or CPU built-in memory to which data is to be written	0 to 65535	16-bit unsigned binary	ANY16
(s2)	TO(P)	Write data or the start device where the write data is stored	-32768 to 32767	16-bit signed binary	ANY16*1
	DTO(P)		-2147483648 to 2147483647	32-bit signed binary	ANY32*1
(n)		Number of write data	0 to 65535	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

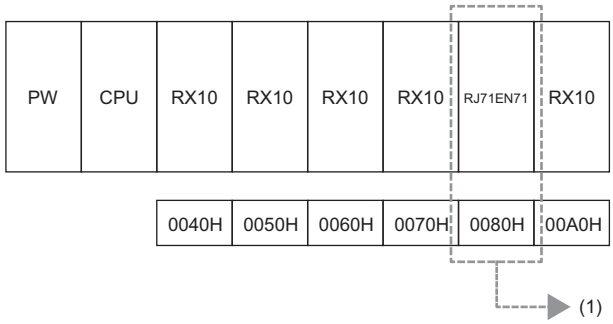
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U/H)	○	○	○	○	○	—	—	○	○	—	—	○
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	—	○	—	—	○*1	—	○	○	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

*1 Only the DTO(P) instruction can be used.

Processing details

- For (U/H), specify the start I/O number of a module or controller with upper 3 digits when it is represented by 4 hexadecimal digits.



(1) Specify K8 or H8 as the start I/O number of the write-target module.

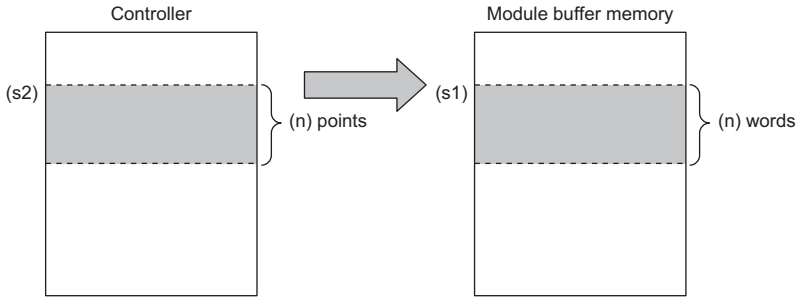
Specify the start I/O numbers of the controller as shown in the table below.

Controller	Start I/O number
Controller No.1	3E0H

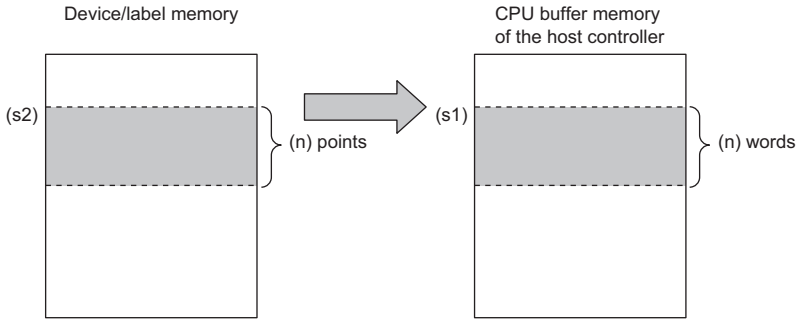
■TO(P)

- This instruction writes the n points of data from the device specified by (s2) to the buffer memory address specified by (s1) and later in the buffer memory in the module or host controller specified by (U/H).

• Writing word data to a module

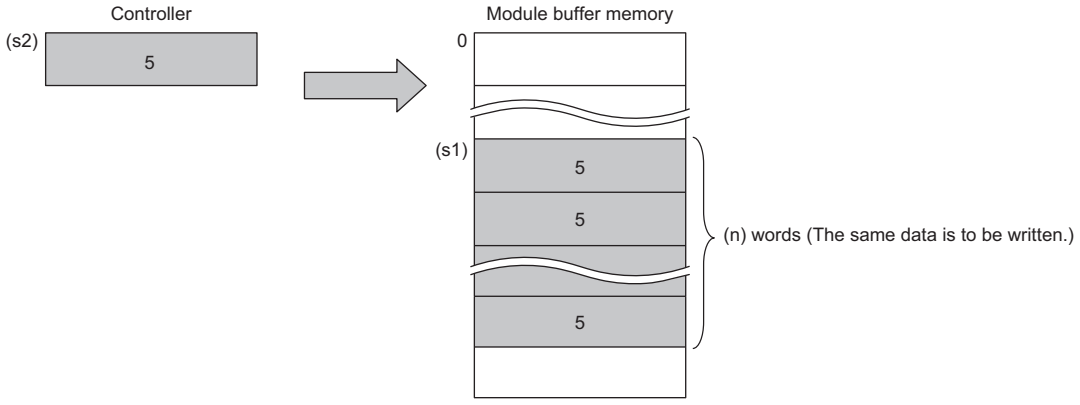


• Writing word data to a host controller

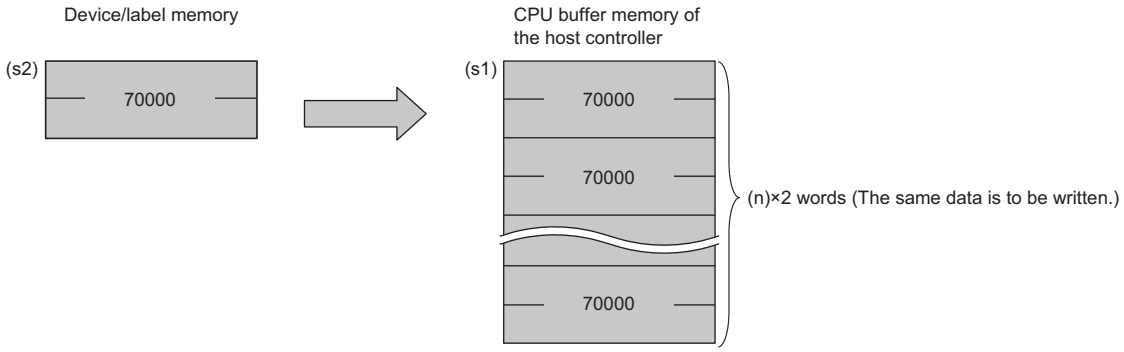


- If a constant is specified in (s2), the instructions write the same data (the value in the device specified by (s2)) to the (n) words from the specified buffer memory address.

• Writing word data to a module



• Writing word data to a host controller

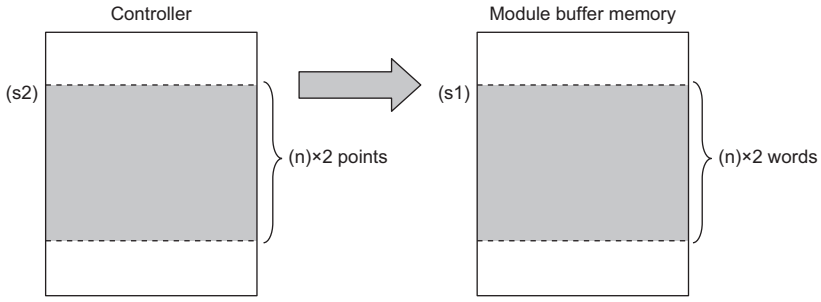


- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

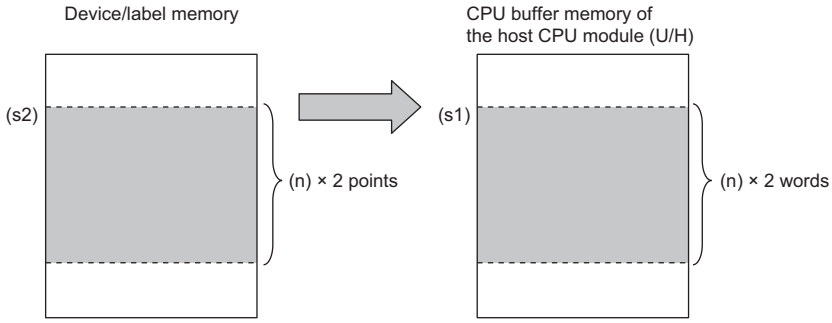
■DTO(P)

- This instruction writes the (n)×2 points of data from the device specified by (s2) to the buffer memory address specified by (s1) and later in the buffer memory in the module or host controller specified by (U/H).

• Writing double word data to a module

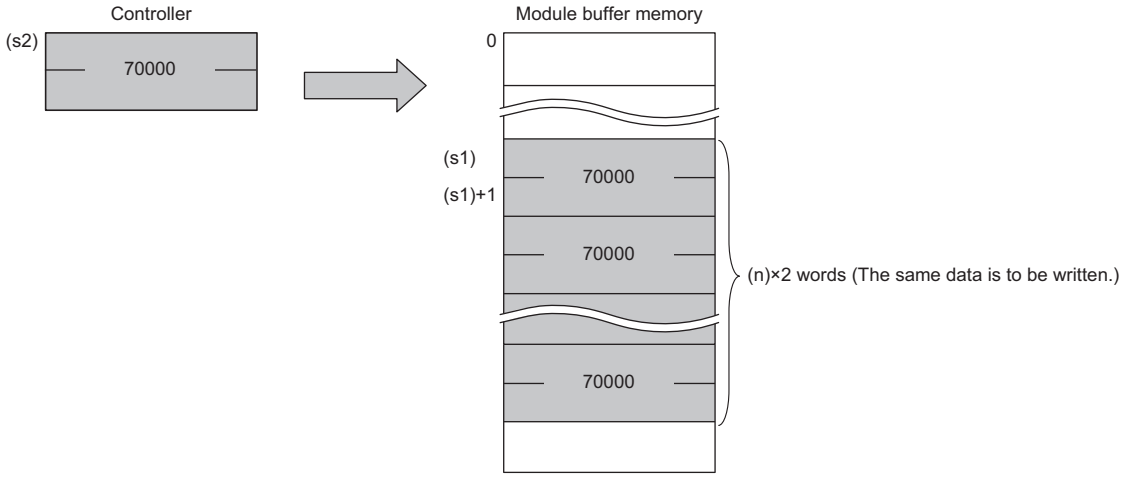


• Writing double word data to a host controller

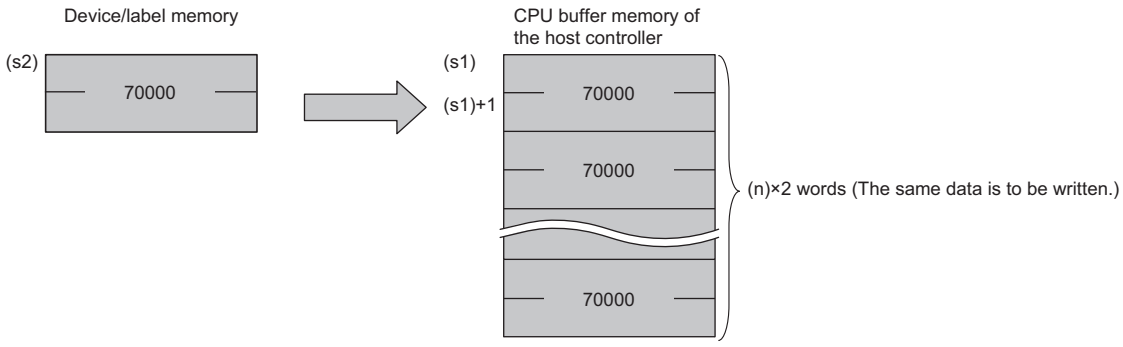


- If a constant is specified in (s2), the instructions write the same data (the value in the device specified by (s2)) to the (n)×2 words from the specified buffer memory address.

• Writing double word data to a module



• Writing double word data to a host controller



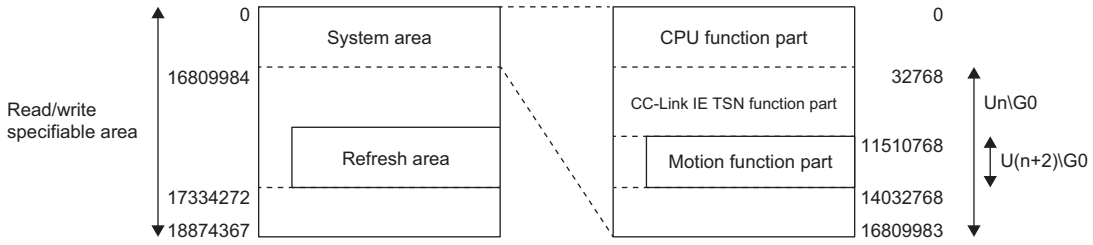
• An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

Operation error

Error code (SD0)	Description
2820H	The module with the I/O number specified by (U) does not have buffer memory.
2823H	The module with the I/O number specified by (H) does not have buffer memory.
	The address specified by (s1) is outside the range of buffer memory.
	The (n) points of data starting from the address specified by (s1) are outside the range of buffer memory. (TO(P) instruction)
	The 2×(n) points of data starting from the address specified by (s1) are outside the range of buffer memory. (DTO(P) instruction)

Point

• If refresh settings are not made for the refresh area of the read/write enabled area in the CPU buffer memory, the area can be used as a read/write specifiable area. Even when refresh settings are made, the area can be used as a read/write specifiable area in the reference send range and later. If the start I/O number Un\G0 or U(n+2)\G0 of the CC-Link IE TSN function part or motion function part is specified in the operand (U/H), the following range can be accessed.



• A CPU buffer memory access device can be used to write data to the CPU buffer memory. (User's manual for the controller used)

• The TO(P) and DTO(P) instructions can write data to the buffer memory address with a capacity of 64K bytes or less. To write data to the buffer memory address with a capacity exceeding 64K bytes, use the TOD(P) or DTOD(P) instruction. (☞ Page 1041 TOD(P), DTOD(P))

Reading 1-word/2-word data from another module (32-bit specification)

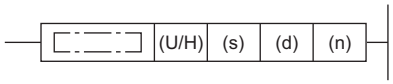
FROMD(P), DFROMD(P)

• FROMD(P):

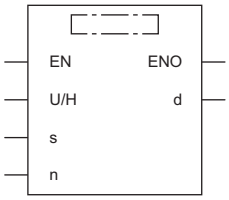
This instruction reads n words of data from the buffer memory address in the specified module.

• DFROMD(P):

This instruction reads n×2 words of data from the buffer memory address in the specified module.

Ladder	ST
	ENO:=FROMD(EN,U/H,s,n,d); ENO:=FROMDP(EN,U/H,s,n,d); ENO:=DFROMD(EN,U/H,s,n,d); ENO:=DFROMDP(EN,U/H,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
FROMD DFROMD	
FROMDP DFROMDP	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U/H)		Start I/O number (first three digits in four-digit hexadecimal representation) of a module and controller	0H to FFH, 3E0H	16-bit unsigned binary	ANY16
(s)		Buffer memory from which the data is read or the start device where the start address of the CPU memory is stored	0 to 4294967295	32-bit unsigned binary	ANY32
(d)	FROMD(P)	Start device for storing the data that has been read	—	16-bit signed binary	ANY16*1
	DFROMD(P)			32-bit signed binary	ANY32*1
(n)		Number of read data	0 to 4294967295	32-bit unsigned binary	ANY32
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

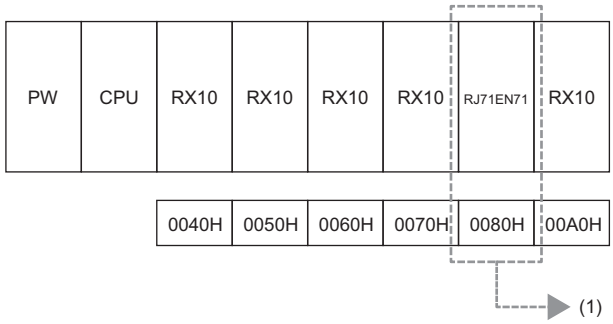
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U/H)	○	○	○	○	○	—	—	○	○	—	—	○
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	—	○	—	—	○*1	—	○	○	—	—	—
(n)	○	○	○	○	○	○	○	○	○	—	—	—

*1 Only the DFROMD(P) instruction can be used.

Processing details

- For (U/H), specify the start I/O number of a module or controller with upper 3 digits when it is represented by 4 hexadecimal digits.



(1) Specify K8 or H8 as the start I/O number of the read-target module.

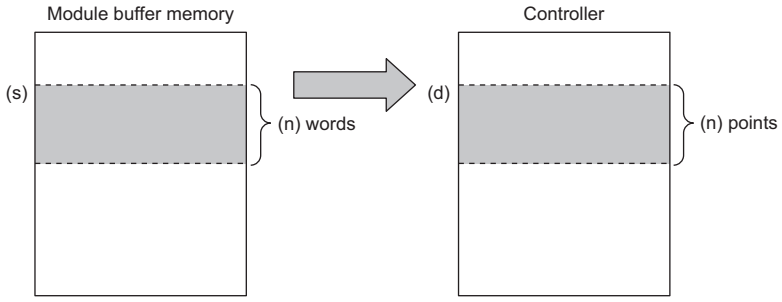
Specify the start I/O numbers of the controller as shown in the table below.

Controller	Start I/O number
Controller No.1	3E0H

■FROMD(P)

- This instruction reads (n) words of data from the buffer memory address specified by (s) in the module specified by (U/H), and stores in the device specified by (d) and later.

• Reading word data from module

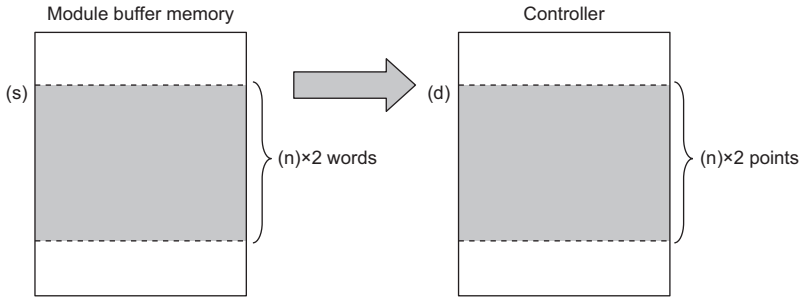


- If the read data (n) is 0, no processing is performed.
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

■DFROMD(P)

- This instruction reads (n)×2 words of data from the buffer memory address specified by (s) in the module specified by (U/H), and stores in the device specified by (d) and later.

• Reading double word data from module



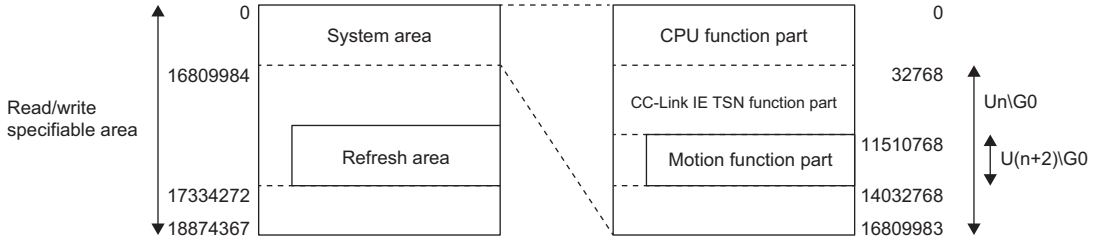
- If the read data (n) is 0, no processing is performed.
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

Operation error

Error code (SD0)	Description
2820H	The module with the I/O number specified by (U) does not have buffer memory.
	The address specified by (s) is outside the range of CPU buffer memory.
	The (n) points of data starting from the address specified by (s) are outside the range of CPU buffer memory. (FROMD(P) instruction)
	The 2×(n) points of data starting from the address specified by (s) are outside the range of CPU buffer memory. (DFROMD(P) instruction)
2823H	The module with the I/O number specified by (H) does not have buffer memory.
	The address specified by (s) is outside the range of buffer memory.
	The (n) points of data starting from the address specified by (s) are outside the range of buffer memory. (FROMD(P) instruction)
	The 2×(n) points of data starting from the address specified by (s) are outside the range of buffer memory. (DFROMD(P) instruction)

Point

- If refresh settings are not made for the refresh area of the read/write enabled area in the CPU buffer memory, the area can be used as a read/write specifiable area. Even when refresh settings are made, the area can be used as a read/write specifiable area in the reference send range and later. If the start I/O number Un\G0 or U(n+2)\G0 of the CC-Link IE TSN function part or motion function part is specified in the operand (U/H), the following range can be accessed.

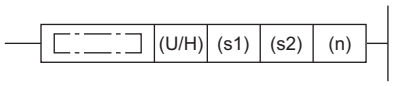


- A CPU buffer memory access device can be used to read data from the CPU buffer memory. (User's manual for the controller used)
- The FROMD(P) and DFROMD(P) instructions can read data from the buffer memory address with a capacity exceeding 64K bytes.

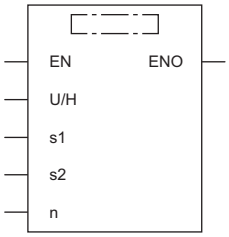
Writing 1-word/2-word data to a module (32-bit specification)

TOD(P), DTOD(P)

- TOD(P): This instruction writes the n points of data from the specified device to the buffer memory in the module/host controller.
- DTOD(P): This instruction writes the n×2 points of data from the specified device to the buffer memory in the module/host controller.

Ladder	ST
	ENO:=TOD(EN,U/H,s1,s2,n); ENO:=TODP(EN,U/H,s1,s2,n); ENO:=DTOD(EN,U/H,s1,s2,n); ENO:=DTODP(EN,U/H,s1,s2,n);

FBD/LD



■Execution condition

Instruction	Execution condition
TOD DTOD	
TODP DTODP	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U/H)		Start I/O number (first three digits in four-digit hexadecimal representation) of a module and controller	0H to FFH, 3E0H	16-bit unsigned binary	ANY16
(s1)		Buffer memory to which the data is written or the start device where the start address of CPU memory is stored	0 to 4294967295	32-bit unsigned binary	ANY32
(s2)	TOD(P)	Write data or the start device where the write data is stored	-32768 to 32767	16-bit signed binary	ANY16*1
	DTOD(P)		-2147483648 to 2147483647	32-bit signed binary	ANY32*1
(n)		Number of write data	0 to 4294967295	32-bit unsigned binary	ANY32
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

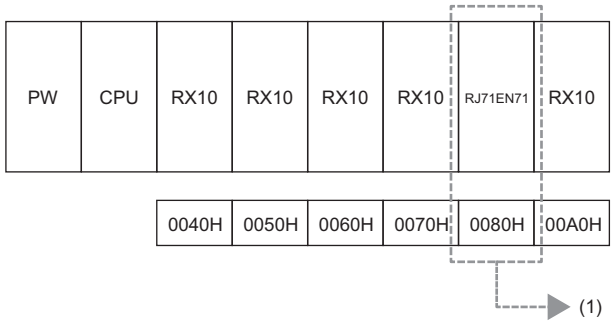
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U/H)	○	○	○	○	○	—	—	○	○	—	—	○
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	—	○	—	—	○*1	—	○	○	—	—	—
(n)	○	○	○	○	○	○	○	○	○	—	—	—

*1 Only the DTOD(P) instruction can be used.

Processing details

- For (U/H), specify the start I/O number of a module or controller with upper 3 digits when it is represented by 4 hexadecimal digits.



(1) Specify K8 or H8 as the start I/O number of the write-target module.

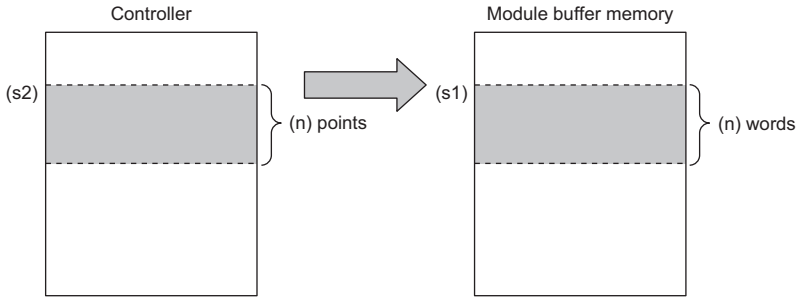
Specify the start I/O numbers of the controller as shown in the table below.

Controller	Start I/O number
Controller No.1	3E0H

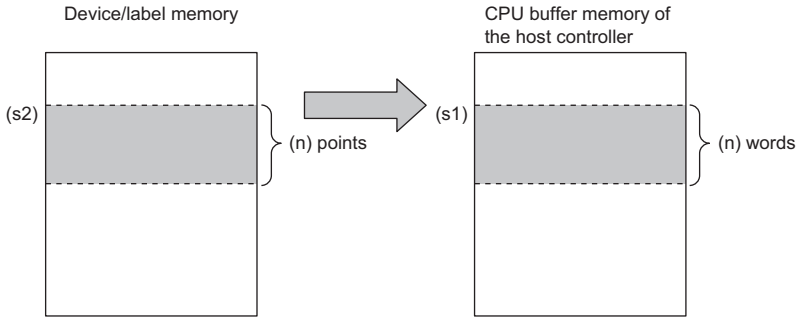
■TOD(P)

- This instruction writes the n points of data from the device specified by (s2) to the buffer memory address specified by (s1) and later in the buffer memory in the module or host controller specified by (U/H).

• Writing word data to a module

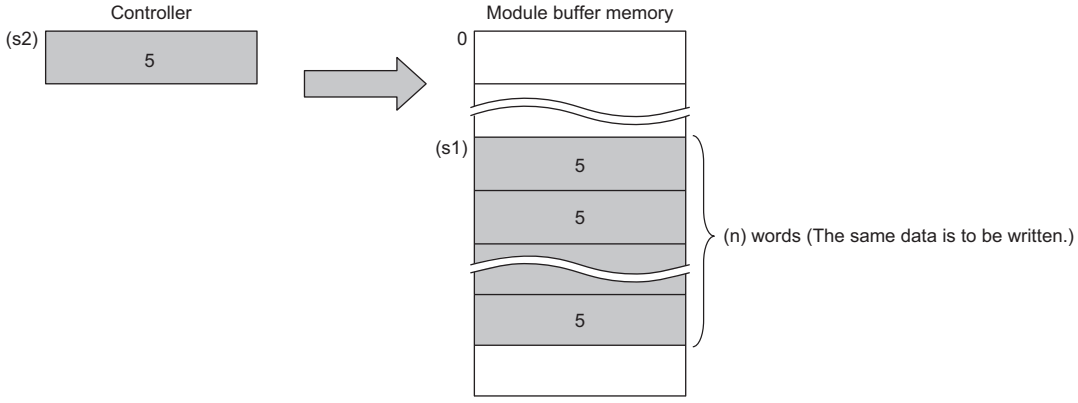


• Writing word data to a host controller

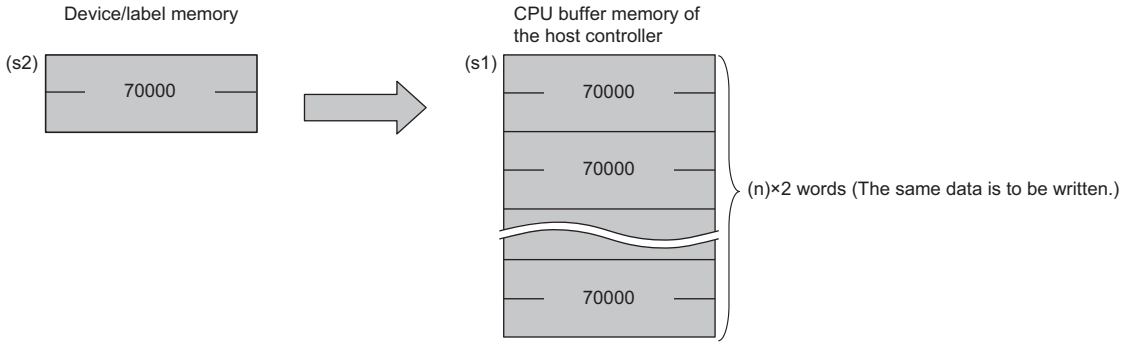


- If a constant is specified in (s2), the instructions write the same data (the value in the device specified by (s2)) to the (n) words from the specified buffer memory address.

• Writing word data to a module



• Writing word data to a host controller

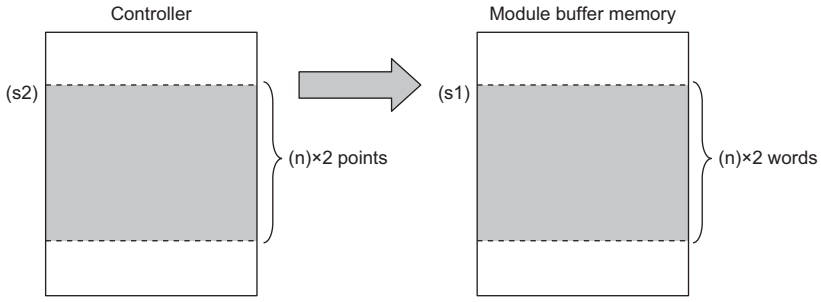


- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

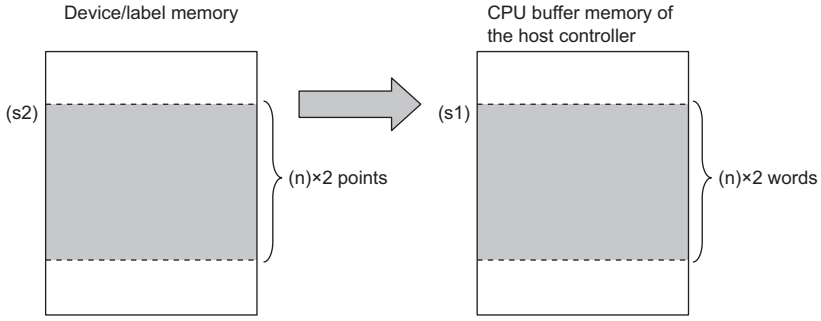
■DTOD(P)

- This instruction writes the (n)×2 points of data from the device specified by (s2) to the buffer memory address specified by (s1) and later in the buffer memory in the module or host controller specified by (U/H).

• Writing double word data to a module

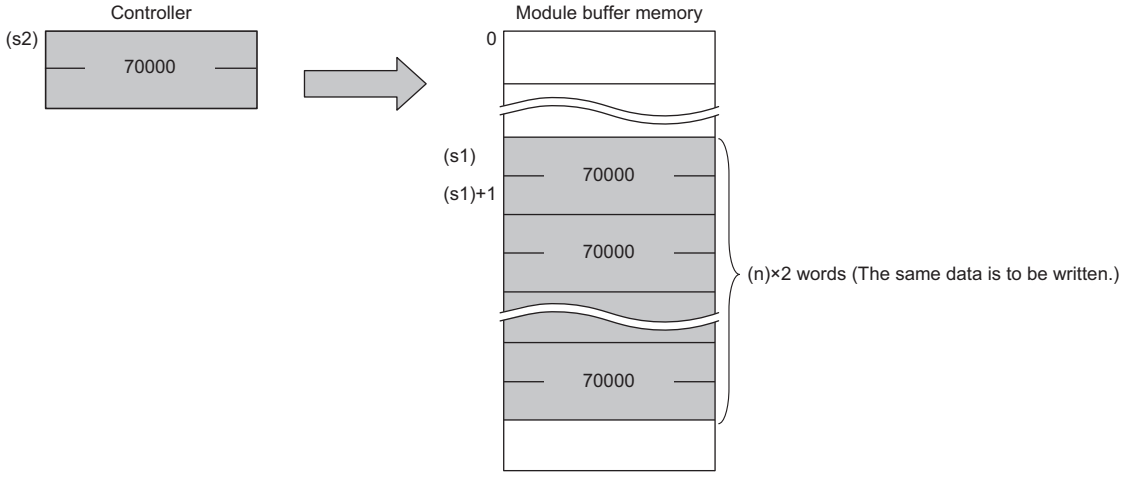


• Writing double word data to a host controller

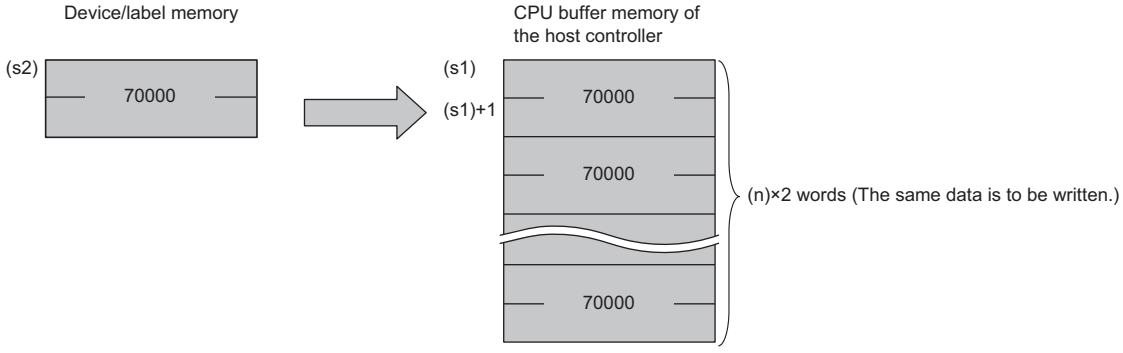


- If a constant is specified in (s2), the instructions write the same data (the value in the device specified by (s2)) to the (n)×2 words from the specified buffer memory address.

• Writing double word data to a module



• Writing double word data to a host controller



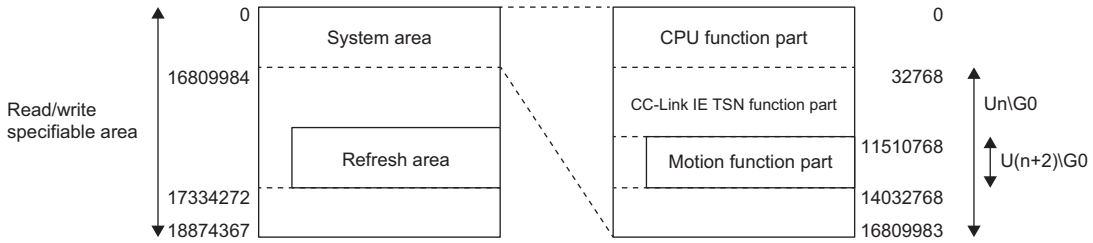
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

Operation error

Error code (SD0)	Description
2820H	The module with the I/O number specified by (U) does not have buffer memory.
	The address specified by (s1) is outside the range of CPU buffer memory.
	The (n) points of data starting from the address specified by (s1) are outside the range of CPU buffer memory. (TOD(P) instruction)
	The 2×(n) points of data starting from the address specified by (s1) are outside the range of CPU buffer memory. (DTOD(P) instruction)
2823H	The module with the I/O number specified by (H) does not have buffer memory.
	The address specified by (s1) is outside the range of buffer memory.
	The (n) points of data starting from the address specified by (s1) are outside the range of buffer memory. (TOD(P) instruction)
	The 2×(n) points of data starting from the address specified by (s1) are outside the range of buffer memory. (DTOD(P) instruction)

Point

- If refresh settings are not made for the refresh area of the read/write enabled area in the CPU buffer memory, the area can be used as a read/write specifiable area. Even when refresh settings are made, the area can be used as a read/write specifiable area in the reference send range and later. If the start I/O number Un\G0 or U(n+2)\G0 of the CC-Link IE TSN function part or motion function part is specified in the operand (U/H), the following range can be accessed.



- A CPU buffer memory access device can be used to write data to the CPU buffer memory. (User's manual for the controller used)
- The TOD(P) and DTOD(P) instructions can write data to the buffer memory address with a capacity exceeding 64K bytes.

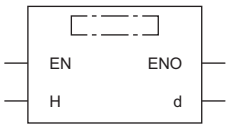
Reading the module model name

TYPERD(P)

These instructions read the module name of the specified slot.

Ladder	ST
	ENO:=TYPERD(EN,H,d); ENO:=TYPERDP(EN,H,d);

FBD/LD



■Execution condition

Instruction	Execution condition
TYPERD	
TYPERDP	

Setting data

■Description, range, data type

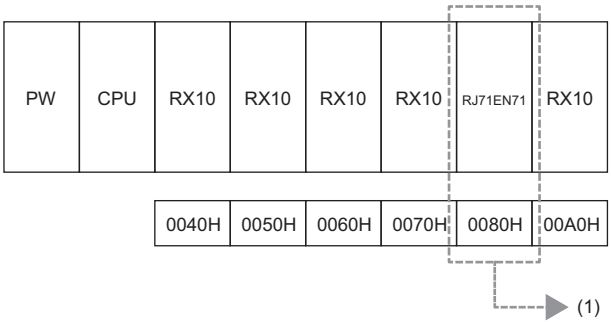
Operand	Description	Range	Data type	Data type (label)
(H)	Start I/O number (first three digits in four-digit hexadecimal representation) of a model read target module	0H to FFH, 3E0H	16-bit unsigned binary	ANY16
(d)	(d)+0: Instruction execution result (d)+1 to (d)+9: Module name	—	Word	ANY16_ARRAY (Number of elements: 10)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(H)	○	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions read the module name in the slot specified by (H), and store the model name in the device specified by (d) and later. The target modules are as follows.
  - Controller
  - Input module
  - Output module
  - I/O combined module
  - Intelligent function module
- For (H), specify the start I/O number of the target module with upper 3 digits when it is represented by 4 hexadecimal digits.



(1) Specify K8 or H8 as the start I/O number of the read-target module.

- For the slot to be specified for the read target when specifying a module that occupies two slots, refer to the number of occupied I/O points described in the manual for each module.
- Specify the start I/O numbers of the controller as shown in the table below.

Controller	Start I/O number
Controller No.1	3E0H

- The result of instruction execution is stored in (d)+0, and a module name is stored in (d)+1 to (d)+9. The following table lists the values to be stored in (d).

Condition	(d)+0	(d)+1 to (d)+9
The read target module has a module name.	0	Module name held by a module
The read target module does not have a module name.	1	String consisting of module type and the number of points
The read target slot is an empty slot.	-1	0000H
The read target module is in the course of online exchange.		
The I/O number specified in (H) is not the start number of a module.		
A restart of the read target module is in progress.		

- When the read target module has a module name, the module name to be stored in (d)+1 and later is as follows.
  - Nine words are used.
  - The name is stored in ASCII characters.
  - 00H is stored in the 18th character.
  - If the number of characters is less than 17, 00H is stored in the remaining characters.
  - The module name held by a module is stored. (Note that it may differ from the module name written to the rating plate.)

Ex.

The following table lists module name examples that are stored.

Target module	Module name example stored
Controller	MXR300-16
I/O module	INPUT_16
Network module	RJ71GP21-SX

Point

If the module name in the I/O assignment setting differs from that of the mounted module, the module name held by the mounted module is stored.

- When the read target module does not have a module name, the string to be stored in (d)+1 and later is as follows.
  - Nine words are used.
  - The name is stored in ASCII characters.
  - 00H is stored in the 18th character.
  - If the number of characters is less than 17, 00H is stored in the remaining characters.
  - A string consisting of a combination of "string indicating the module type" and "string indicating the number of points" is stored.

Ex.

The following table lists string examples that are stored.

Target module	String example stored
Input module	INPUT_16
Output module	OUTPUT_32
I/O combined module	MIXED_64
Intelligent function module	INTELLIGENT_128

The following table lists string examples that indicate the numbers of points.

Number of points	String example that indicates the number of points.
16 point	_16
32 point	_32
48 point	_48
64 point	_64
128 point	_128
256 point	_256
512 point	_512
1024 point	_1024

Point

If the number of points in the I/O assignment setting differs from that of the mounted module, the number of points of the mounted module is stored.

- When the module model name is read by specifying the start I/O number of the CC-Link IE TSN function part and motion function part in the controller, the model name is not stored as shown below. For more details on the controller's system configuration, check the user's manual for the controller used.

Ex.

The following table shows an example of reading the module model name of each function from the controller.

System to be specified	Start I/O number	(d)+0	(d)+1 to (d)+9
CPU function part	3E00H	0	MXR300-16
CC-Link IE TSN function part and motion function part	0000H	-1	0000H
	0010H		
	0020H		
	0030H		

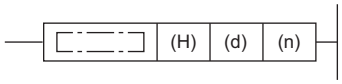
Operation error

Error code (SD0)	Description
2800H	The value set to (H) is other than 0 to FFH or 3E0H.
2810H	Communications with the read target module is disabled due to a failure of the module.

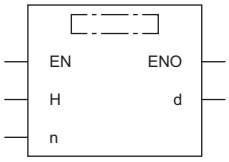
Reading module specific information

UNIINFRD(P)

These instructions read the specified points of module information.

Ladder	ST
	ENO:=UNIINFRD(EN,H,n,d); ENO:=UNIINFRDP(EN,H,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
UNIINFRD	
UNIINFRDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(H)	Start I/O number (first three digits in four-digit hexadecimal representation) of an information read target module	0H to FFH	16-bit unsigned binary	ANY16
(d)	Start device for storing module information	—	Word	ANY16*1
(n)	Number of read data points	0 to 256	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

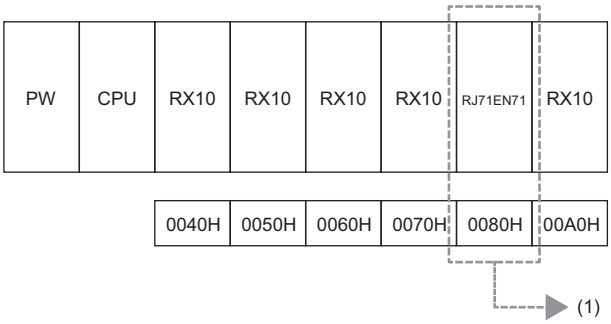
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(H)	○	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	—	○	—	—	—	—	○	○	—	—	—

Processing details

- These instructions read the module information by the number of points specified by (n) from the module specified by (H), and store the information in the device areas specified by (d) and later. Even if the module type or the number of points is changed in I/O assignment, the status of the mounted module is read.
- For (H), specify the start I/O number of the module, whose information is to be read, with upper 3 digits when it is represented by 4 hexadecimal digits.
- If an I/O number other than the start I/O number of the read target module is specified, module information in which only the module mount status is on and any other status is off is stored.



(1) Specify K8 or H8 as the start I/O number of the read-target module.

• The following shows detailed module information.

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
(d)																
(d)+1																

Device	Bit	Item name	Description
(d)	b0	Number of I/O points	0000: 16 points 0001: 32 points 0010: 48 points 0011: 64 points 0100: 128 points 0101: 256 points 0110: 512 points 0111: 1024 points
	b1		
	b2		
	b3		
	b4	Module type	0000: Input module, or not set 0010: Output module 0100: I/O combined module (different numbers) 0110: I/O combined module (same number) 1000: Intelligent function module
	b5		
	b6		
	b7		
	b8	Series type	011: MELSEC iQ-R series module 111: Unknown
	b9		
	b10		
	b11	Group number	000: CPU No.1 control
	b12		
	b13		
	b14	Start of slot	0: Not start of slot 1: Start of slot
	b15	Module mounted	0: Module not mounted 1: Module mounted*2
(d)+1	b0	Error	00: Normal 01: Minor error 10: Medium error 11: Major error
	b1		
	b2	Module ready	0: Not ready 1: Ready
	b3	Reserved	0: Fixed
	b4	Reserved	0: Fixed
	b5	Reserved	0: Fixed
	b6	Reserved	0: Fixed
	b7		
	b8	Reserved	0: Fixed
	b9	Reserved	0: Fixed
	b10	External power supply	0: During normal operation, or no external power supply 1: Power off
	b11	Fuse status	0: Normal operation 1: Fuse blown
	b12	Reserved	0: Fixed
	b13	Reserved	0: Fixed
	b14	Reserved	0: Fixed
	b15	Module access	0: Access disabled*1 1: Access enabled

*1 For example, the module is being connected or removed.

*2 For a module which occupies 32 points or more, information is stored in (d) + 0 and (d) + 1, and ON information is stored in the later devices only when the module is mounted.

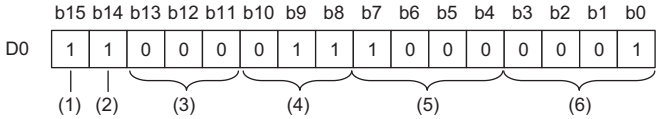
• In the following program example, when M0 turns on, the information on the module mounted at I/O number 0080H is stored in D0 and later. Specify K2 for (n) to read module information of the 32-point module.



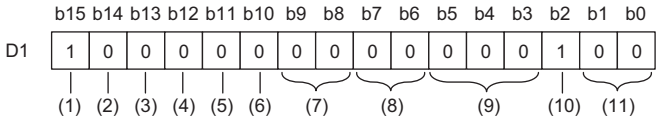


[Result of reading]

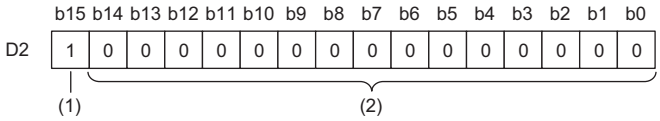
• Module information of the RJ71EN71 is read.



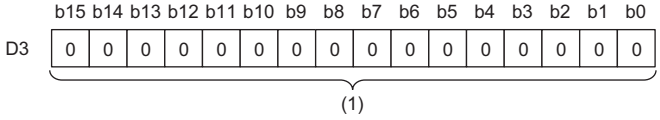
(1) Module mounting status
(2) Start of slot
(3) CPU No.1 control
(4) MELSEC iQ-R series module
(5) Intelligent function module
(6) 32-point module



(1) Module access enabled
(2) Fixed to 0
(3) Fixed to 0
(4) Fixed to 0
(5) No fuse blown
(6) Fixed to 0
(7) Fixed to 0
(8) Fixed to 0
(9) Fixed to 0
(10)Module ready
(11)Normal operation (no module error)

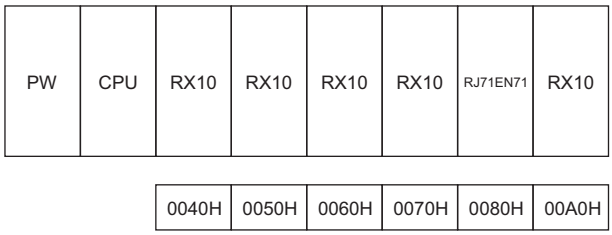


(1) A 32-point module is connected in the latter 16 points.
(2) All 0s because information is stored in D0 and D1.



(1) All 0s because information is stored in D0 and D1.

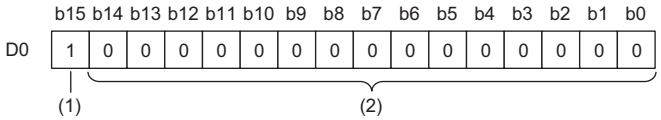
• If an I/O number other than the start I/O number is specified in (H) for a module having 32 or more I/O points, module information in which only the module mount status is on and any other status is off is stored. In the following program example, when M0 turns on, the information on the module mounted at I/O number 0080H is stored in D0 and later. If K2 is specified for (n), module information is read for the last 16 points of the 32-point unit and the next 16-point unit.



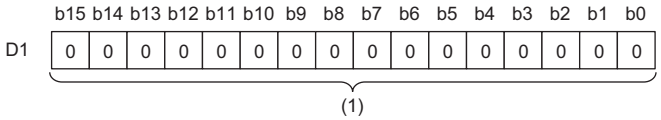


[Result of reading]

• Only the module mount status is on because the I/O number in the latter 16 points of the RJ71EN71 is specified by (H).

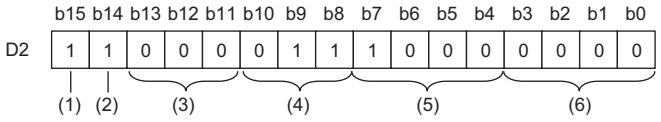


(1) A 32-point module is connected in the latter 16 points.
(2) All 0s because the latter 16 points of a 32-point module is specified in n1.

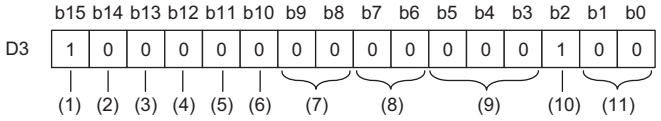


(1) All 0s because the latter 16 points of a 32-point module is specified in n1.

• Module information of RX10 is read.



(1) Module mounting status
(2) Start of slot
(3) CPU No.1 control
(4) MELSEC iQ-R series module
(5) Input module
(6) 16-point module



(1) Module access enabled
(2) Fixed to 0
(3) Fixed to 0
(4) Fixed to 0
(5) No fuse blown
(6) Fixed to 0
(7) Fixed to 0
(8) Fixed to 0
(9) Fixed to 0
(10)Module ready
(11)Normal operation (no module error)

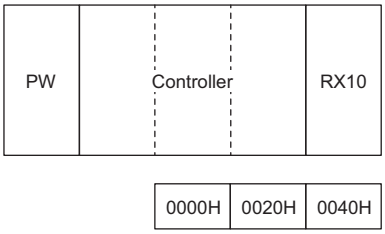
- When module information is read by specifying the start I/O numbers of the CC-Link IE TSN function part and motion function part in the controller, the following three items are stored in module information. The module information (OFF) is stored except for the following items.

Device	Bit	Item name	Description
(d)	b0	Number of I/O points	0000: 16 points 0001: 32 points 0010: 48 points 0011: 64 points 0100: 128 points 0101: 256 points 0110: 512 points 0111: 1024 points
	b1		
	b2		
	b3		
	b14	Start of slot	0: Not start of slot 1: Start of slot
	b15	Module mounted	0: Module not mounted 1: Module mounted

Ex.

When M0 turns on, the information on the module mounted at I/O number 0000H to 003FH is stored in D0 or later. The number of input points for each function is 32 for the CC-Link IE TSN function part and 32 for the motion function part.

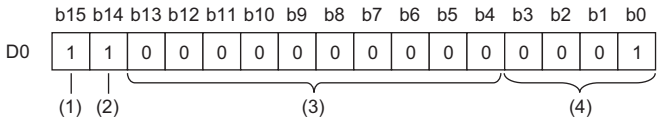
The start I/O number of the CC-Link IE TSN function part and motion function part is specified as 0000H



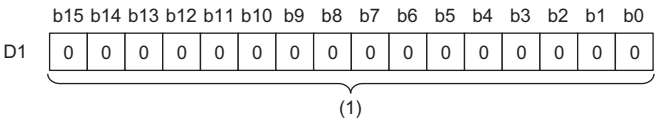
Each refers as:
0000H: CC-Link TSN function part
0020H: Motion function part



[Result of reading]
■[0000H: Start I/O number for network communication function]

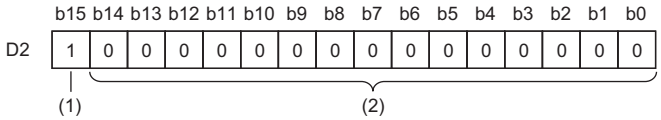


(1) Module mounting status
(2) Start of slot
(3) All 0s.
(4) 32-point module

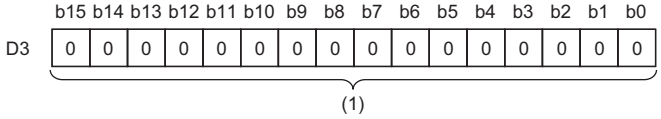


(1) All 0s.

■[0010H: Non-start I/O number for network communication function]

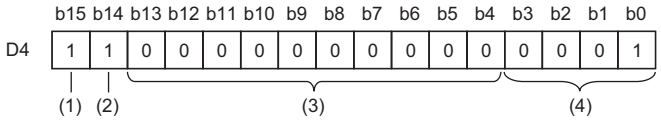


(1) Module mounting status
(2) All 0s.

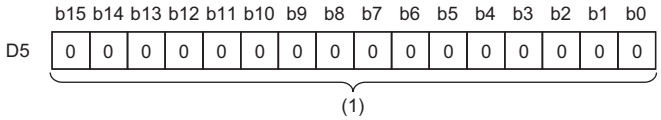


(1) All 0s.

■[0020H: Start I/O number for motion function part]

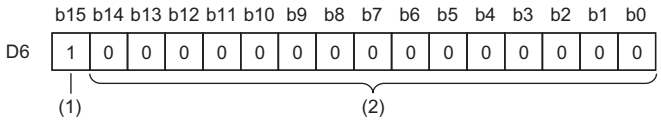


(1) Module mounting status
(2) Start of slot
(3) All 0s.
(4) 32-point module

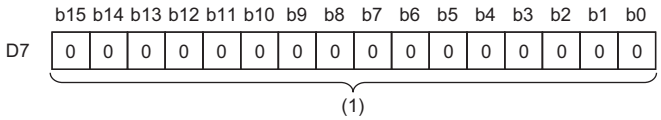


(1) All 0s.

■[0030H: Non-start I/O number for motion function part]



(1) Module mounting status
(2) All 0s.



(1) All 0s.

Operation error

Error code (SD0)	Description
2800H	The value set to (H) is out the range, 0 to FFH.
3285H	The value set to (n) is out of the range, 0 to 256.
	The total of the values in (H) and (n) is 257 or greater.

23 PID OPERATION INSTRUCTION

PID operation instruction and PID control instruction

There are two types of instructions for PID control.

Type	Application	Reference (Overview)	Reference (Details)
PID operation instruction	This instruction is used to perform PID control using auto tuning.	Page 1057 Overview	Page 1069 PID Operation Instruction
PID control instruction	This instruction is used to perform PID control equal to that of the MELSEC-Q series and MELSEC-L series.	Page 1073 Overview	Page 1082 PID Control Instructions (Inexact Differential),Page 1094 PID Control Instructions (Exact Differential)

The following table lists the specifications comparison between PID operation instruction and PID control instruction.

Item	PID operation instruction	PID control instruction
PID operation method	Inexact differential	Inexact differential/exact differential
Sampling period/sampling time	1 to 32767ms	10 to 60000ms
Number of control loops	1 loop per instruction	32 loops maximum
Auto tuning	Enabled (limit cycle method and step response method)	Disabled

23.1 Overview

This section describes an overview of PID control using the PID operation instruction.

PID operation instruction

The PID operation instruction calculates the manipulated value (MV) from the process value (PV) by combining the P action (proposal action), I action (integral action), and D action (derivative action) to get closer to the set value (SV).

■Alarm output function

This function turns on the alarm output for the variations of input (process value) and output (manipulated value).

■Output upper/lower limit value setting

This function suppresses the increase of integral terms in PID control by setting the output upper/lower limit values.

■Auto tuning function

This function automatically sets the proportional gain (K_P), integral time (T_I), and derivative time (T_D). The auto tuning is performed in two methods: limit cycle method and step response method.

■Operation method of the PID operation instruction

The instruction performs PID operation in the velocity type or the process value derivative type.

Basic operational expressions of PID operation instruction [Reference]

The instruction performs PID operation in the velocity type or the process value derivative type.

An operational expression of direct action or reverse action is executed depending on the value of bit 0 in the device specified by (s3)+1 (Action setting (ACT)).

The operation is performed using the control data stored in the device areas, (s3) and later.

• Operational expressions

Action (Bit 0 of (s3)+1)	Operational expression
Direct action (Off)	$\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} EV_n + D_n\}$ $EV_n = PV_{nf} - SV$ $D_n = \frac{T_D}{T_S + K_D \cdot T_D}(-2PV_{nf-1} + PV_{nf} + PV_{nf-2}) + \frac{K_D \cdot T_D}{T_S + K_D \cdot T_D} \cdot D_{n-1}$ $MV_n = \Sigma\Delta MV$
Reverse action (On)	$\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} EV_n + D_n\}$ $EV_n = SV - PV_{nf}$ $D_n = \frac{T_D}{T_S + K_D \cdot T_D}(2PV_{nf-1} - PV_{nf} - PV_{nf-2}) + \frac{K_D \cdot T_D}{T_S + K_D \cdot T_D} \cdot D_{n-1}$ $MV_n = \Sigma\Delta MV$

The meaning of the symbols in the operational expressions is as follows.

Symbol	Meaning
EV_n	Deviation in the sampling period this time
EV_{n-1}	Deviation in the sampling period last time
SV	Set value
PV_{nf}	Process value of the sampling period this time (after filtering)
PV_{nf-1}	Process value of the sampling period last time (after filtering)
PV_{nf-2}	Process value of the sampling period two times before (after filtering)
ΔMV	Output variation amount
MV_n	Manipulated value this time
D_n	Derivative term this time
D_{n-1}	Derivative term of the sampling period last time
K_P	Proportional gain
T_S	Sampling period
T_I	Integral constant
T_D	Derivative constant
K_D	Derivative gain

PV_{nf} (process value of the sampling period this time (after filtering)) is calculated by using the following operational expression. If the input filter coefficient is not set, the value will be the same as the input process value (PV).

$PV_{nf} = PV_n + L(PV_{nf-1} - PV_n)$

Where, PV_{nf}: Process value for the sampling period this time, L: Filter coefficient, PV_{nf-1}: Process value for the sampling period last time (after filtering)

Control data

The details on the control data used by the PID operation instruction are described.

Sampling time: (s3)

Setting range: 1 to 32767 [ms]

Set a cycle (ms) to perform PID operation.

• Auto tuning (limit cycle method)

Set a cycle so that the following condition is satisfied: Operation cycle of the programmable controller < Sampling time

• Auto tuning (step response method)

Set a cycle to 1000ms or longer.

■Maximum error

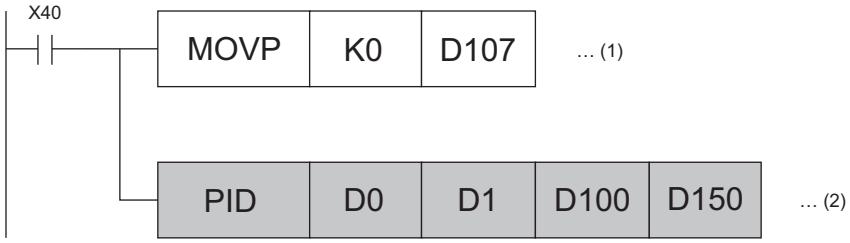
The maximum error of the sampling time (T_S) is -(1 operation cycle + 1ms) to +(1 operation cycle).

• When the sampling time (T_S) value is small

The variation in the maximum error may become a problem. Set the constant scan and execute the instruction, or program it in the timer interrupt routine.

• When the sampling time value is shorter than one operation cycle of the programmable controller

A PID operation error (11A6H) occurs, but PID operation is executed assuming that the sampling time (T_S) equals to the operation cycle. In this case, program the instruction in the timer interrupt routine, clear the value in (s3)+7, and then execute the instruction.



(1) Reset the value in (s3)+7. (The internal processing register is cleared by the pulse conversion command at the first execution of the interrupt routine.)
(2) Execute the PID operation.

Action setting: (s3)+1

■Direction (direct action/reverse action): Bit 0 of (s3)+1

Setting range: Off = Direct action, On = Reverse action

Select the direction of PID control, direction action or reverse action.

• Auto tuning (limit cycle method)

The PID control direction must be set.

• Auto tuning (step response method)

At completion of auto tuning executed in whichever mode, direct action or reverse action, setting is made automatically.

[Direct action (bit 0 of (s3)+1 = Off)]

[Reverse action (bit 0 of (s3)+1 = On)]

■Alarm setting (input variation, output variation): Bit 1 and bit 2 of (s3)+1

Setting range: Off = Alarm disabled, On = Alarm enabled

The input and output variation amounts can be checked. The check result can be checked in (s3)+24. (☞ Page 1065 Alarm output flag: (s3)+24)

• Input variation (bit 1 of (s3)+1)

To use the input variation alarm, the following bit needs to be on and the values need to be set to the following devices.

Setting item			Description	Setting range
Action setting (ACT)	(s3)+1	Bit 1	Input variation alarm	On: Enabled Off: Disabled
Input variation alarm setting value	(s3)+20		Input variation (increase) alarm setting value	0 to 32767
	(s3)+21		Input variation (decrease) alarm setting value	

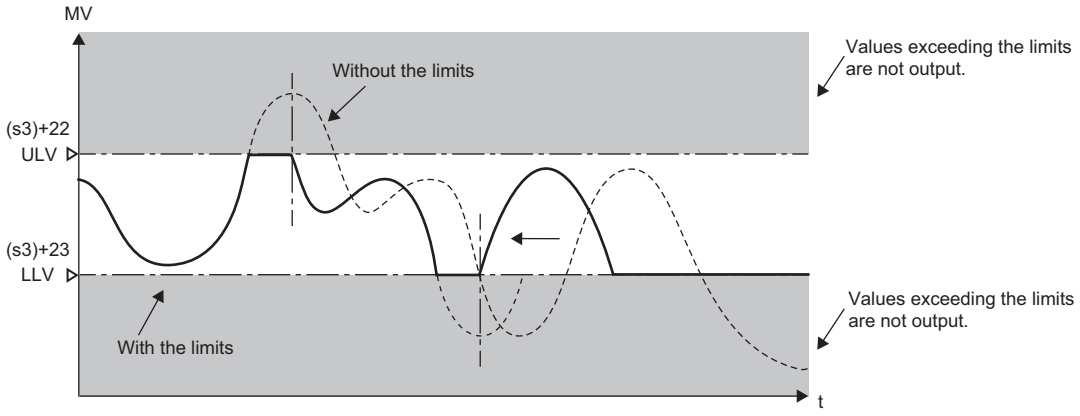
• Output variation (bit 2 of (s3)+1)

To use the output variation alarm, the following bits need to be on and the values need to be set to the following devices.

Setting item			Description	Setting range
Action setting (ACT)	(s3)+1	Bit 2	Output variation alarm	On: Enabled Off: Disabled
		Bit 5	Output upper/lower limit value setting	Off (always)
Output variation alarm setting value	(s3)+22		Output variation (increase) alarm setting value	0 to 32767
	(s3)+23		Output variation (decrease) alarm setting value	

■Output upper/lower limit value setting: Bit 5 of (s3)+1

The manipulated value (MV) will be as follows according to this setting.



MV: Manipulated value
ULV: Output upper limit value
LLV: Output lower limit value
t: Time

This setting suppresses the increase of integral terms in PID control. To use this function, turn off the bit 2 of (s3)+1.

Setting item			Description	Setting range
Action setting (ACT)	(s3)+1	Bit 2	Output variation alarm	Off (always)
		Bit 5	Output upper/lower limit value setting	On: Enabled Off: Disabled

Input filter: (s3)+2

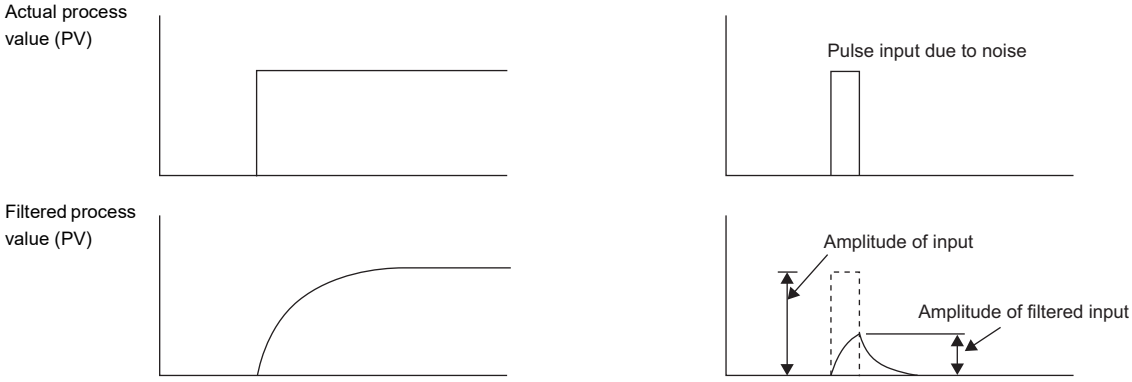
Setting range: 0 to 99 [%]

PID control: Proportional action, integral action, derivative action

The input filter (α) is a software filter used to reduce the variations caused by noise in the process value (PV). The influence of noise can be suppressed by setting the input filter (α) properly according to the characteristics and noise level of the control target.

- If the filter coefficient is too small, the effect will be reduced.
- If the filter coefficient is too large, the input response will deteriorate.

The input filter (α) acts on the set value (SV) and thus affects the proportional action, integral action, and derivative action.



Proportional gain: (s3)+3

Setting range: 1 to 32767 [%]

PID control: Proportional action

The manipulated value (MV) increases in proportion to the deviation (difference between the set value (SV) and the process value (PV)) in proportional operation. This ratio is called the proportional gain (K_P) and represented by the following relational expression.

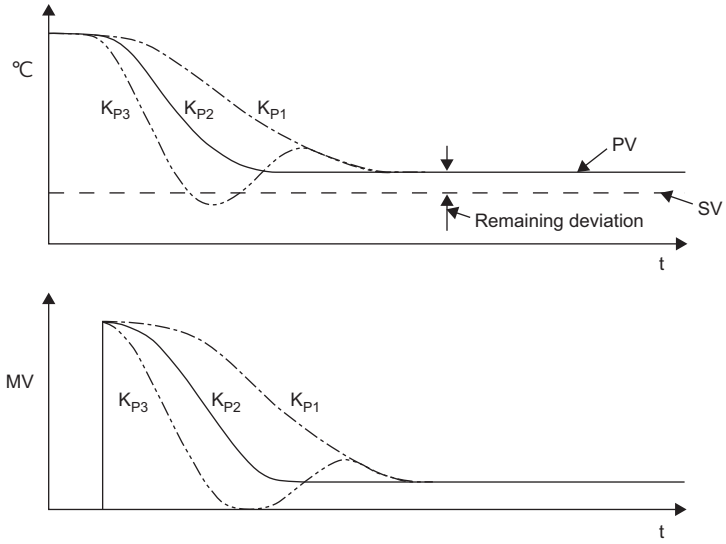
Manipulated value (MV) = Proportional gain (K_P) × Deviation (EV)

The reciprocal of the proportional gain (K_P) is called the proportional band.

As the proportional gain (K_P) increases, the motion to get the process value (PV) closer to the set value (SV) becomes strong.

Ex.

Proportional action (P action) in the case of cooling (direct action)



Proportional gain (K_P): $K_{P3} > K_{P2} > K_{P1}$

℃: Temperature

SV: Set value

PV: Process value

MV: Manipulated value

t: Time

Integral time: (s3)+4

Setting range: 0 to 32767 [× 100ms] (0 = ∞) (No integration)

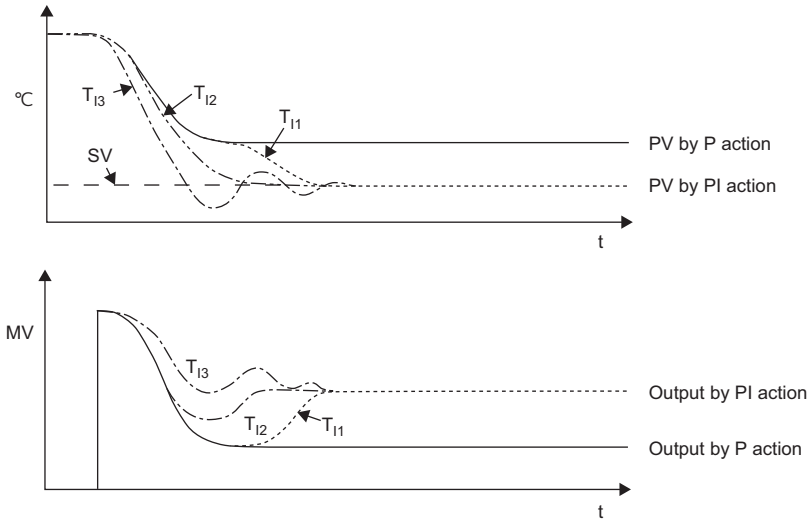
PID control: Integral action

The integral time (T_I) is the time from when a deviation occurs in integral action to when the output of the integral action becomes the output of proportional action.

Reducing the integral time (T_I) accelerates the integral operation.

Ex.

PI action in the case of cooling (direct action)



Integral time (T_I): $0 < T_{I3} < T_{I2} < T_{I1}$

℃: Temperature

SV: Set value

MV: Manipulated value

t: Time

Derivative gain: (s3)+5

Setting range: 0 to 200 [%]

PID control: Derivative action

The output of the derivative action is filtered. The derivative gain (K_D) affects only the derivative action.

- If the derivative gain (K_D) is decreased, the output responds instantaneously to a change in the process value (PV) caused by a disturbance.
- If the derivative gain (K_D) is increased, the output takes time to respond to a change in the process value (PV) caused by a disturbance.

Point

First, set the derivative gain (K_D) to 0 and adjust it using the input filter (α). If the change in the output responds too sensitive to the disturbance, increase the value.

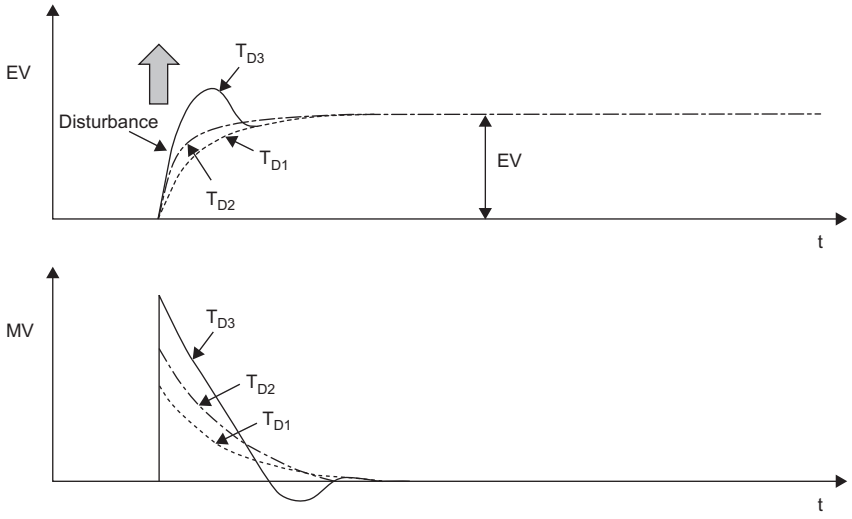
Derivative time (T_D): (s3)+6

Setting range: 0 to 32767 [× 10ms] (0 = no derivation)

PID control: Derivative action

The derivative time is used to be sensitive to the change in the process value (PV) caused by a disturbance and minimize the changes.

- Increasing the derivative time (T_D) prevents more positively the control target from fluctuating due to a disturbance.

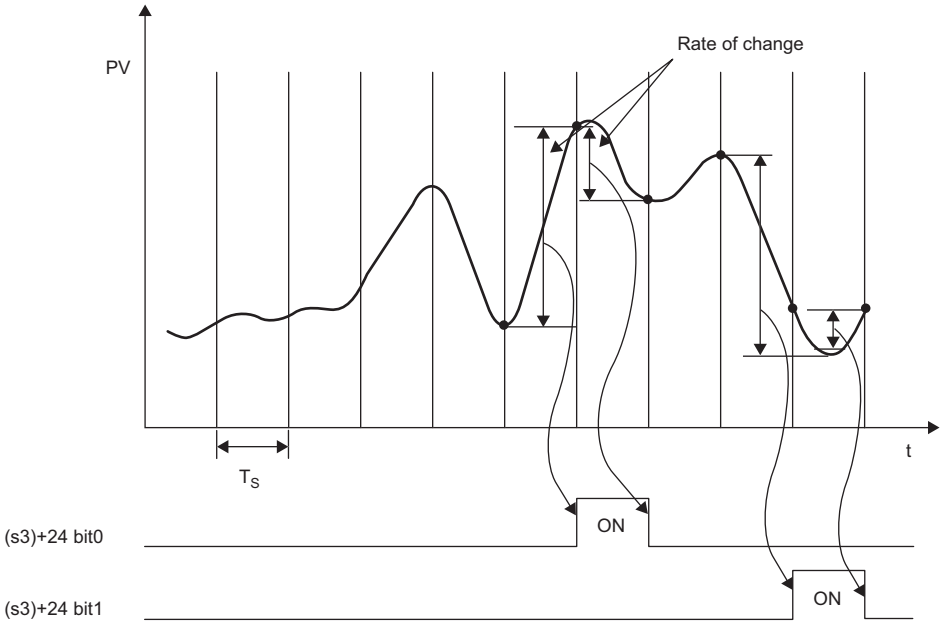


Derivative time (T_D): $T_{D3} > T_{D2} > T_{D1}$
EV: Deviation
MV: Manipulated value
t: Time

Alarm output flag: (s3)+24

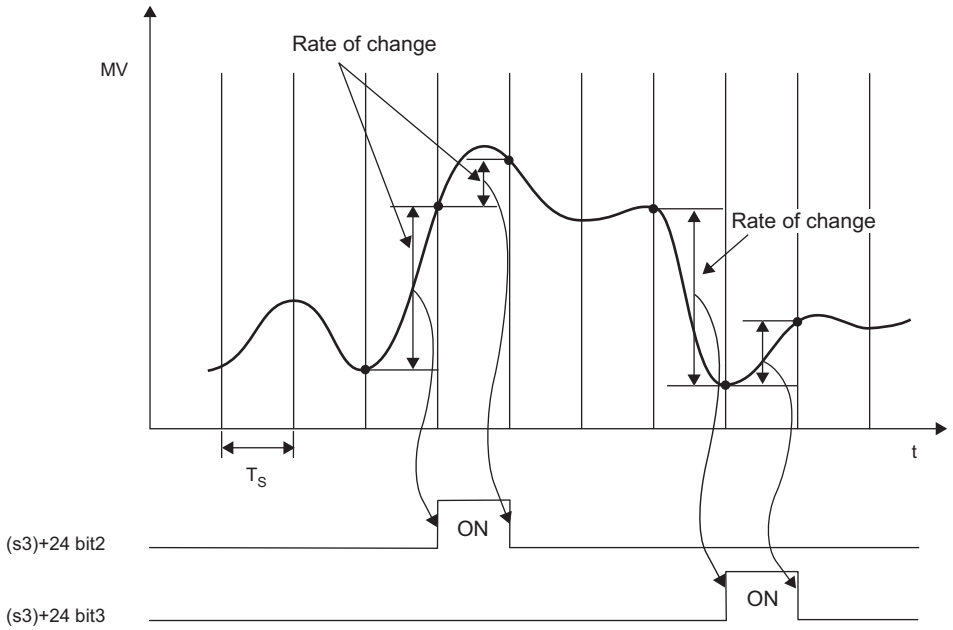
When the set input/output variation is exceeded, bits of (s3)+24 turn on as an alarm flag immediately after execution of the PID operation instruction.

- When the input variation alarm (bit 1 of (s3)+1) is on



PV: Process value
T_S: Sampling time
t: Time
Bit 0 and bit 1 of (s3)+24: Alarm flag

• When the output variation alarm (bit 2 of (s3)+1) is on



MV: Manipulated value
T_S: Sampling time
t: Time
Bit 2 and bit 3 of (s3)+24: Alarm flag

Auto tuning

To obtain satisfactory results in PID control, it is required to determine the optimum values of each constant (control data) appropriate for the control target. The auto tuning function automatically sets the important constants for PID control, proportional gain, integral time, and derivative time.
The auto tuning function is performed in two methods: limit cycle method and step response method.

Point

Start auto tuning after the system becomes stable. If not, auto tuning may not be performed correctly.

Limit cycle method

For an overview of the limit cycle method, refer to the following.
☞ Page 2006 Overview of limit cycle method

■Control data set by auto tuning (limit cycle method)

- Proportional gain (K_P): (s3)+3
- Integral time (T_I): (s3)+4
- Derivative time (T_D): (s3)+6

■Procedure

1. Set the action of PID operation.

Set the action (direct action or reverse action) with the bit 0 of (s3)+1 (Action setting (ACT)).

2. Select the limit cycle method.

Turn on the bit 6 of (s3)+1 (Action setting (ACT)).
(When the bit is off, the step response method is selected.)

3. Turn on the auto tuning execution flag.

Turn on the bit 4 of (s3)+1 (Action setting (ACT)).

4. Set an input filter constant.

Set a value in (s3)+2 (Input filter (α)).

5. Set a sampling time.

Set a value in (s3)+0 (Sampling time (T_S)).

6. Set the output upper limit value (ULV).

Set the upper limit of the manipulated value (MV) in (s3)+26 (Output upper limit value (ULV)).

7. Set the output lower limit value (LLV).

Set the lower limit of the manipulated value (MV) in (s3)+27 (Output lower limit value (LLV)).

8. Set the PV value threshold (hysteresis) width (SHPV).

Set a value in (s3)+25 (PV value threshold (hysteresis) width (SHPV)).

9. Set a set value (SV).

Set a set value (SV) in (s1) of the PID operation instruction.

10. Start auto tuning.

Auto tuning starts based on the process value (PV) when the start contact device of the PID operation instruction turns on.
The auto tuning related flags (bit 4 and bit 6) of (s3)+1 (Action setting (ACT)) turn off upon completion of auto tuning.

Step response method

For an overview of the step response method, refer to the following.

☞ Page 2007 Overview of step response method

■Control data set by auto tuning (step response method)

- Direction (direct action/reverse action): Bit 0 of (s3)+1
- Proportional gain (K_P): (s3)+3
- Integral time (T_I): (s3)+4
- Derivative time (T_D): (s3)+6

■Procedure

1. Set the output value for auto tuning.

Transfer the output value for auto tuning to the manipulated value (MV).

Set the output value for auto tuning to the "maximum allowable output value of the external device × 0.5 to 1".

2. Set the data that is not set by auto tuning.

Set the following items, which are not set by auto tuning, depending on the system.

Setting item		Remarks
(s1)	Set value (SV)	Set a value so that the difference from the process value (PV) becomes 150 or more.*1
(s3)	Sampling time (T_S)	Set a cycle to 1000ms or longer.*2
(s3)+2	Input filter (α)	—
(s3)+5	Derivative gain (K_D)	When the input filter is used, set the derivative gain to 0.
Others		Set other control data as needed.

*1 Difference between the set value (SV) and the process value (PV)
To perform auto tuning correctly, the difference between the process value (PV) and the set value (SV) must be 150 or more at the time of auto tuning start. If the difference is less than 150, set a set value (SV) for auto tuning.
After completion of auto tuning, set the SV back to the original value.

*2 Sampling time (T_S) setting
To perform auto tuning, set the sampling time (T_S) to 1000ms or longer.
The sampling time should also be sufficiently longer than the output change period.

3. Start auto tuning.

Turn on the bit 4 of (s3)+1 (Action setting (ACT)) to start auto tuning.

When the amount of variation from the process value (PV) at start of auto tuning to the set value (SV) becomes 1/3 or more, auto tuning is completed and the bit 4 of (s3)+1 (Action setting (ACT)) turns off automatically.

Precautions

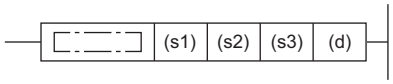
- Measures to be taken when the process value (PV) does not change

If the process value (PV) does not change properly due to an analog input disconnection or any other problems, auto tuning does not end. Detect and avoid such problems by creating a program that monitors the input value and the elapsed time from the start of auto tuning.

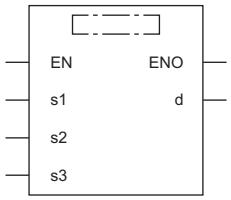
23.2 PID Operation Instruction

PID

This instruction performs PID operation using the values set in (s1) to (s3), and stores the operation result in (d) at each cycle of sampling time.

Ladder	ST
	ENO:=PID(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
PID	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Device where the set value (SV) is stored	-32768 to 32767	16-bit signed binary	ANY16
(s2)	Device where the process value (PV) is stored	-32768 to 32767	16-bit signed binary	ANY16
(s3)	Start device where the control data are stored	—	16-bit signed binary	ANY16
(d)	Device for storing the manipulated value (MV)	-32768 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	○	—	—	—	○	—	—	—	—
(s2)	—	—	○	○	—	—	—	○	—	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	○	—	—	—	○	—	—	—	—

Processing details

When the set value (s1), process value (s2), and control data (s3) to (s3)+6 are set and a program is executed, the operation result, manipulated value (MV), is stored to (d) at every cycle of sampling time. For details, refer to the following.

☞ Page 1057 Overview

■Setting items (arguments)

Setting item		Description	Number of occupied points
(s1)	Set value (SV)	Set the set value (SV). ■Auto tuning (limit cycle method) If the set value for auto tuning differs from that for normal PID control, set a value obtained by adding a bias value, and then set the SV back to the original value when the auto tuning flag turns off.	1 point
(s2)	Process value (PV)	Set the input value for PID operation.	1 point
(s3)	Control data*1	■Auto tuning (limit cycle method) The device areas of 29 points starting from the device specified in (s3) are used.	29 points
		■Auto tuning (step response method) The device areas of 25 points starting from the device specified in (s3) are used.	25 points
(d)	Manipulated value (MV)	■Normal PID control The user sets the initial manipulated value before execution of the instruction. After execution, the operation result is stored. ■Auto tuning (limit cycle method) During auto tuning, the output upper limit value (ULV) or output lower limit value (LLV) is output automatically. Upon completion of auto tuning, the given MV is set. ■Auto tuning (step response method) The user sets the step manipulated value before execution of the instruction. During auto tuning, the MV cannot be changed by the PID instruction.	1 point

*1 When auto tuning is not used, the same number of points are occupied as when the step response method is used.

■Setting items (control data)

Operand: (s3)				
Device	**Item**		**Description**	**Remarks**
+0	Sampling time (T_S)		1 to 32767 [ms]	The sampling time must be longer than the operation cycle of the programmable controller.
+1	Action setting (ACT)	Bit 0	0: Direct action 1: Reverse action	Action direction specification
		Bit 1	0: Input variation alarm disabled 1: Input variation alarm enabled	—
		Bit 2	0: Output variation alarm disabled 1: Output variation alarm enabled	Do not turn on bit 2 and bit 5 at the same time.
		Bit 3	Reserved	—
		Bit 4	0: Auto tuning disabled 1: Auto tuning enabled	—
		Bit 5	0: No output upper/lower limit value setting 1: Output upper/lower limit value setting enabled	Do not turn on bit 2 and bit 5 at the same time.
		Bit 6	0: Step response method 1: Limit cycle method	Auto tuning mode selection
		Bits 7 to 15	Reserved	—
+2	Input filter constant (α)		0 to 99 [%]	0 = No input filter
+3	Proportional gain (K_P)		1 to 32767 [%]	—
+4	Integral time (T_I)		1 to 32767 [× 100ms]	0 = ∞ (No integration)
+5	Derivative gain (K_D)		0 to 200 [%]	0 = No derivative gain
+6	Derivative time (T_D)		1 to 32767 [× 10ms]	0 = No derivation
+7 to +19	These areas are used for internal processing of PID operation, and therefore data cannot be changed.			
+20*1	Input variation (increase) alarm setting value		0 to 32767	Operation setting (ACT): (s3) +1 Enabled when bit 1 is 1
+21*1	Input variation (decrease) alarm setting value		0 to 32767	Operation setting (ACT): (s3) +1 Enabled when bit 1 is 1
+22*1	Output variation (increase) alarm setting value		0 to 32767	Operation setting (ACT): (s3) +1 Enabled when bit 2 is 1 and bit 5 is 0
	Output upper limit setting value		-32768 to 32767	Operation setting (ACT): (s3) +1 Enabled when bit 2 is 0 and bit 5 is 1
+23*1	Output variation (decrease) alarm setting value		0 to 32767	Operation setting (ACT): (s3) +1 Enabled when bit 2 is 1 and bit 5 is 0
	Output lower limit setting value		-32768 to 32767	Operation setting (ACT): (s3) +1 Enabled when bit 2 is 0 and bit 5 is 1
+24*1	Alarm output	Bit 0	0: Input variation (increase) not exceeded 1: Input variation (increase) exceeded	Operation setting (ACT): (s3) +1 Enabled when bit 1 is 1 or bit 2 is 1
		Bit 1	0: Input variation (decrease) not exceeded 1: Input variation (decrease) exceeded	—
		Bit 2	0: Output variation (increase) not exceeded 1: Output variation (increase) exceeded	—
		Bit 3	0: Output variation (decrease) not exceeded 1: Output variation (decrease) exceeded	—
+25*2	PV value threshold (hysteresis) width (SHPV)		Set according to the fluctuation of the process value (PV).	Operation setting (ACT): (s3) +1 Occupied when bit 6 is 1 (with the limit cycle method selected)
+26*2	Output upper limit value (ULV)		Upper limit (ULV) of the manipulated value (MV)	
+27*2	Output lower limit value (LLV)		Lower limit (LLV) of the manipulated value (MV)	
+28*2	Wait time setting parameter from the end of tunning cycle to the start of PID control (K_W)		-50 to 32717 [%]	

*1 Occupied when the bit 1 of (s3)+1 (Action direction (ACT)) is 1, the bit 2 is 1, or the bit 5 is 1.
*2 Occupied when the bit 6 of (s3)+1 (Action direction (ACT)) is 1 (limit cycle method).

Operation error

Error code (SD0)	Description
11A0H	A value set for the sampling time (T_S) is out of the range, $T_S \leq 0$.
11A1H	A value set for the input filter constant (α) is out of the range, $\alpha < 0$ or $100 \leq \alpha$.
11A2H	A value the proportional gain (K_P) is out of the range, $K_P < 0$.
11A3H	A value set for the integral time (T_I) is out of the range, $T_I < 0$.
11A4H	A value set for the derivative gain (K_D) is out of the range, $K_D < 0$ or $201 \leq K_D$.
11A5H	A value set for the derivative time (T_D) is out of the range, $T_D < 0$.
11A6H	A value set for the sampling time (T_S) is less than the operation cycle of the programmable controller.
11A7H	The process value variation (ΔPV) overflowed.
11A8H	Deviation (EV) overflowed.
11A9H	The calculated integral value overflowed.
11AAH	The derivative gain (K_D) value overflowed.
11ABH	The calculated derivative value overflowed.
11ACH	The PID operation result overflowed.
11ADH	A value less than the output lower limit value is set for the output upper limit value.
11AEH	A value less than 0 is set for the input variation alarm setting value or output variation alarm setting value.
11AFH	■Step response method Improper auto tuning result • The deviation at the start of auto tuning (step response method) is 150 or less. • The deviation at the end of auto tuning (step response method) is one-third or more of the deviation at the time of start.
11B0H	■Step response method Auto tuning operation direction mismatch • The action direction estimated according to the relation between the set value (SV) and the process value (PV) at the start of auto tuning (step response method) and the action direction of the manipulated value (MV) did not match.
11B1H	■Step response method Improper auto tuning operation • Auto tuning (step response method) failed to operate correctly because the process value (PV) did not change properly.
11B2H	■Limit cycle method A value equal to or less than the output lower limit value (LLV) is set for the output upper limit value (ULV) for auto tuning (limit cycle method).
11B3H	■Limit cycle method A value set for the PV value threshold (hysteresis) width (SHPV) for auto tuning (limit cycle method) is out of the range, SHPV < 0.
11B4H	■Limit cycle method The system area used for auto tuning (limit cycle method) has been overwritten.
11B5H	■Limit cycle method The measurement time for auto tuning (limit cycle method) has been exceeded, and τ and τon time cannot be obtained properly. (τon > τ, τon < 0, τ < 0)
11B6H	■Limit cycle method The proportional gain (K_P) calculated by auto tuning (limit cycle method) overflowed.
11B7H	■Limit cycle method The integral time (T_I) calculated by auto tuning (limit cycle method) is out of the range, 0 to 32767.
11B8H	■Limit cycle method The derivative time (T_D) calculated by auto tuning (limit cycle method) is out of the range, 0 to 32767.

24 PID CONTROL INSTRUCTIONS

This instruction is for PID control.

- PID operation instructions
- PID control instructions

For how to use or compare them, refer to the following.

☞ Page 1057 PID operation instruction and PID control instruction

24.1 Overview

This section describes the operation methods, procedures, and helpful functions of PID control by using PID control instructions.

Point

The PID control instructions include those for inexact differential and exact differential.

Inexact differential is PID control that applies a primary delay filter to the input of a differentiation term, and is useful for the following.

- For control susceptible to high-frequency noise
- When energy effective to actuate an operation end is not provided when a step change occurs in an exact differential system

Exact differential is PID control that uses the input of a differential term as it is.

Operation method

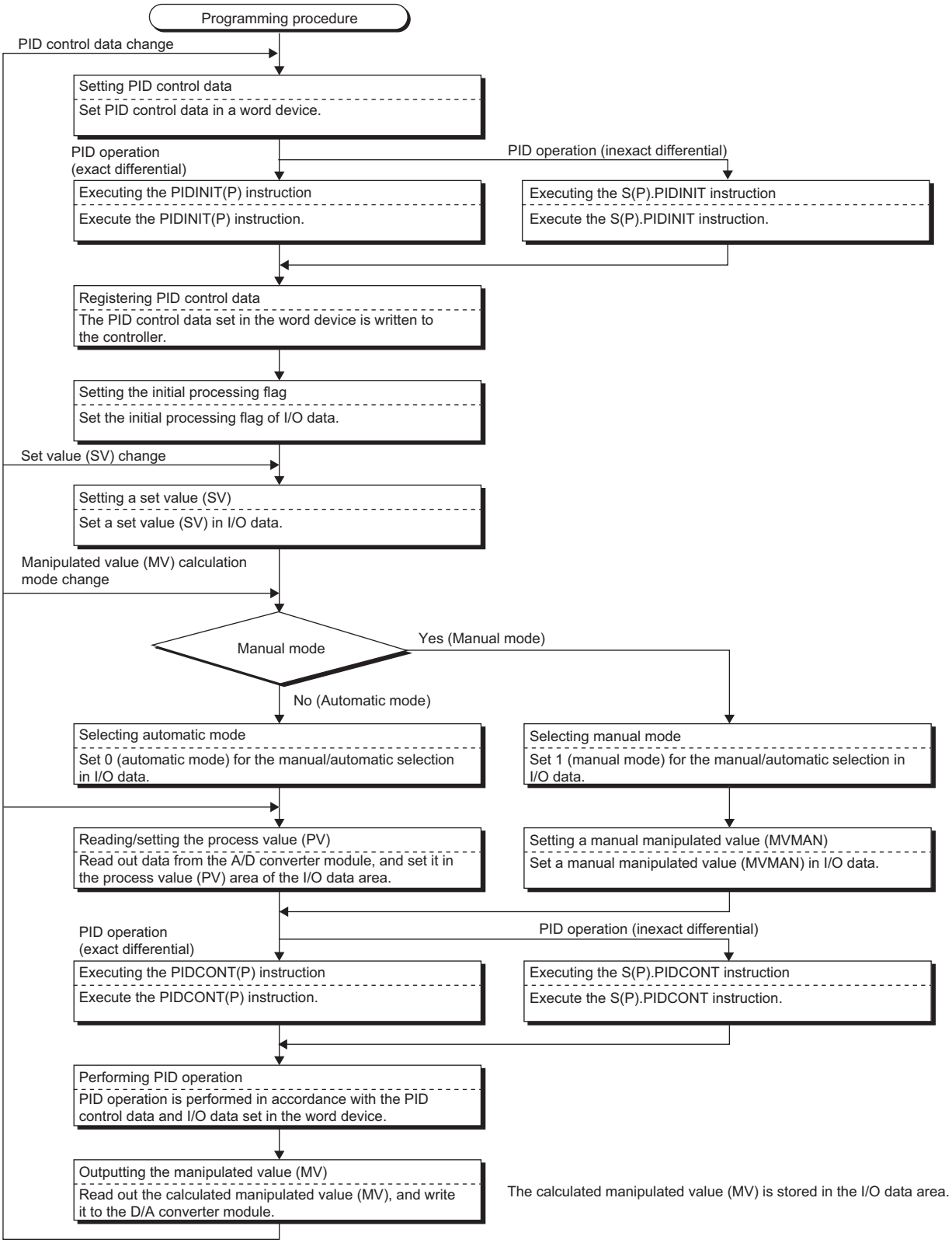
Two types of operation methods are available for PID control by using PID control instructions: velocity type and process value differentiation type.

The following table summarizes each operation methods.

Operation method	Description
Velocity type	This type of control calculates variations in the manipulated values (MV) during PID operation. The actual MV is the accumulated value of variations calculated each sampling period.
Process value differentiation type	This type of control performs PID operation by differentiating the process value (PV). Since the deviation is not subject to differentiation, an abrupt change in output due to derivative action can be reduced when the deviation is generated by changing the set value.

PID control procedure

Create a program for PID control following the procedure below. (Devices are used in the procedure.)



Point

- PID control data can be registered or changed every scan of the program. Whenever the data are registered or changed, execute the PIDINIT instruction. If not executed, the registered or changed data are not used when the PIDCONT instruction is executed.
- Note that the PIDINIT instruction does not need to be executed if the PID control data are changed for the specified loop by using the PIDPRMW instruction.

PID control data

PID control data is used to set the reference values for PID operation. The set values need to be registered to the controller by using the PIDINIT instruction before PID operation by the PIDCONT instruction starts. The PID control data can be set to a word device area with any numbers. Note that all the data for the number of loops used must be set in the area with consecutive device numbers.

For data assignment, refer to the following.

- Inexact differential: Page 1084 S(P).PIDINIT
- Exact differential: Page 1096 PIDINIT(P)

PID control data are classified into two types: data common to all loops and data for each loop.

Type	Item	Description	Setting range	Processing when set data is out of the valid range
Data common to all loops	Number of loops used	Number of loops where PID operation is performed	1 to 32	An error occurs, and PID operation is not performed for all loops.
	Number of loops in one scan	Number of loops to be used in one PID operation when multiple loops reach the sampling period at a same time	1 to 32	

Type	Item	Description	Setting range	Processing when set data is out of the valid range
Data for each loop	Operational expression selection	Selects direct action or reverse action.	0: Direct action 1: Reverse action	An error occurs, and PID operation is not performed for the corresponding loop.
	Sampling period (T_S)	Sets a cycle of PID operation.	1 to 6000 (in increments of 10ms)	
	Proportional constant (K_P)	A constant of proportionality in PID operation	1 to 10000 (in increments of 0.01)	
	Integral constant (T_I)	A constant that expresses the magnitude of the integral action effect. Increasing the integral constant slows down the variation in the manipulated value.	1 to 32767 (in increments of 100ms)	
	Derivative constant (T_D)	A constant that expresses the magnitude of the derivative action effect. Increasing the derivative constant causes a significant change in the manipulated value even with slight change of the control objective.	0 to 30000 (in increments of 10ms)	
	Filter coefficient (α)	Degree of filtering applied to the process value (input value from the A/D converter module). The filtering effect decreases as the value gets closer to 0.	1 to 100	
	Manipulated value lower limit (MVLL)	Lower limit of the manipulated value (MV) calculated by PID operation in automatic mode. If the MV is smaller than the MVLL, the MVLL is used as the MV.	-50 to 2050*1 -32768 to 32767*2	In the case of *1, if the MVLL or MVHL value is out of the valid range, it will be converted as follows. • If the value is smaller than -50, -50 is used. • If the value is bigger than 2050, 2050 is used.
	Manipulated value upper limit (MVHL)	Upper limit of the manipulated value (MV) calculated by PID operation in automatic mode. If the MV is bigger than the MVHL, the MVHL is used as the MV.	-50 to 2050*1 -32768 to 32767*2	
	Manipulated value variation rate limit (ΔMVL)	Limit of variation in the manipulated values calculated last time and this time. If the variation exceeds the limit, 1 is set to b1 of the alarm device. Note that the variation amount will not be limited actually. Even if the variation exceeds the limit, it is used as it is, and the MV is calculated.	0 to 2000*1 0 to 32767*2	In the case of *1, if the ΔMVL or ΔPVL value is out of the valid range, it will be converted as follows. • If the value is smaller than 0, 0 is used. • If the value is bigger than 2000, 2000 is used.
	Process value variation rate limit (ΔPVL)	Limit of variation in the process values input last time and this time. If the variation exceeds the limit, 1 is set to b0 of the alarm device. Note that the variation amount will not be limited actually. Even if the variation exceeds the limit, it is used as it is, and the PID operation is performed.	0 to 2000*1 0 to 32767*2	
	Derivative gain (K_D)*3	A time period (operation delay) for derivative action. The time period decreases as the value gets bigger. (The operation becomes closer to exact differential.)	0 to 32767 (in increments of 0.01)	An error occurs, and PID operation is not performed for the corresponding loop.

*1 When the PID limit is restricted

*2 When the PID limit is not restricted

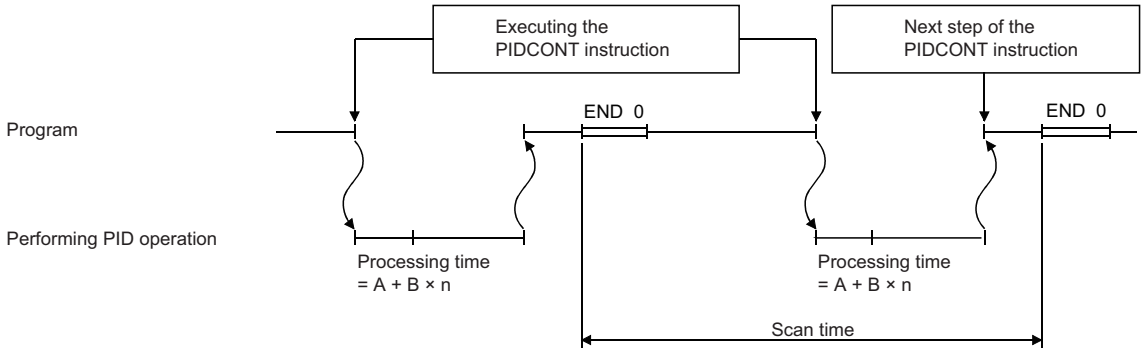
*3 For PID control (inexact differential) only

■Number of loops used and number of execution loops in one scan

The number of loops used is the number of loops where PID operation is performed. The sampling period is measured for the set number of loops when the PIDCONT instruction is executed, and PID operation is performed for the loops that reach the specified sampling period.

Processing time of the PIDCONT instruction increases in proportion to the number of loops where PID operation is performed.

The number of execution loops in one scan is the number of loops where PID operation is performed in one scan when there are multiple loops that reach the specified sampling period. If this number is set, PID operation is performed only for the set number of loops, and PID operation for the rest of the loops that reach the sampling period will be performed in the next scan.



A: Time required to measure sampling period

B: Time required to perform PID operation for a single loop

n: Number of loops

Point

When the number of loops that reach the sampling period exceeds the number of execution loops in one scan, the priority order will be as follows:

- The loop with the smallest loop number is given the highest priority.
- If there are loops where PID operation is performed and not performed in the last scan, the priority is given to the ones where PID operation is not performed in the last scan.

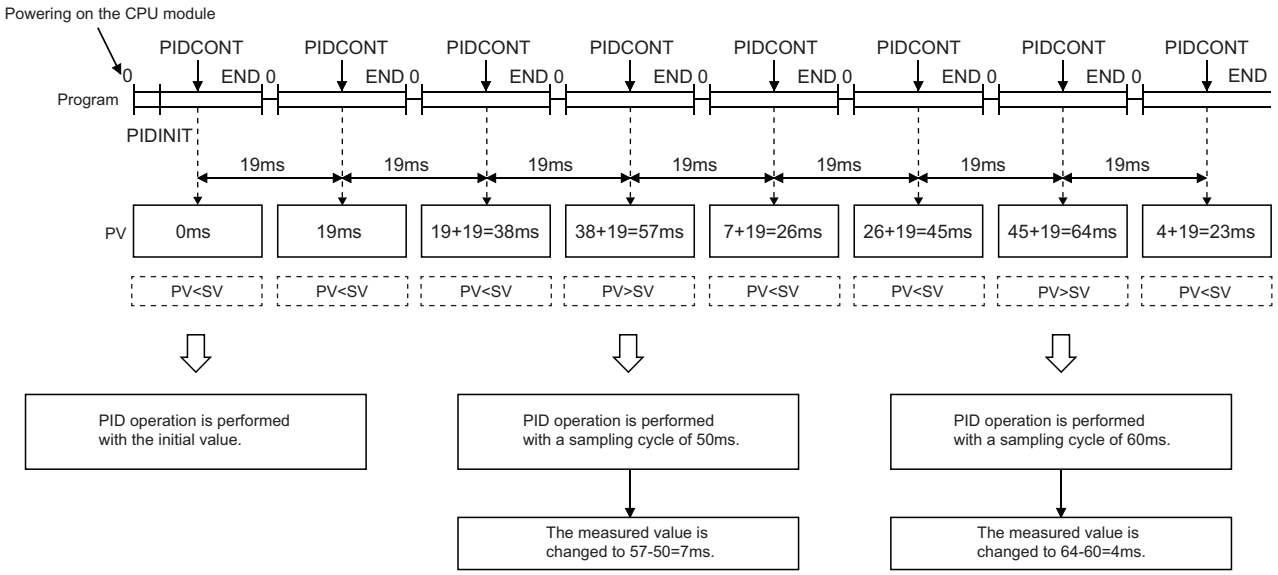
■Sampling period

Sampling period is a cycle in which PID operation is performed. The measurement time of a single scan is added to the total measurement time up to the last scan every time the PIDCONT instruction is executed. The total measurement time reaches or exceeds the set sampling period, PID operation of the corresponding loop is performed.

Set the sampling period used in PID operation in increments of 10ms.

Ex.

When the sampling period is 50ms



PV: Measured value

SV: Set value

Restriction

Since the sampling period is measured at execution of the PIDCONT instruction, a value smaller than the program scan time cannot be set as the sampling period. If set, PID operation is performed with a scan time.

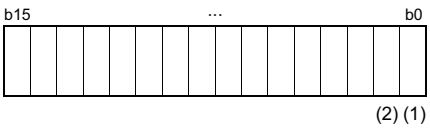
I/O data

I/O data consists of input data, such as the set value (SV) and process value (PV), which are set to perform PID operation, and output data, such as operation results. The I/O data can be set to a word device area with any numbers. Note that all the data for the number of loops used must be set in the area with consecutive device numbers.

For data assignment, refer to the following.

- Inexact differential: Page 1087 S(P).PIDCONT
- Exact differential: Page 1098 PIDCONT(P)

The I/O data area is divided into two areas: data area assigned for each loop and work area used by the system.

Item		Description	Setting range	Remarks
Set value	SV	A target value in PID control	0 to 2000*1 -32768 to 32767*2	In the case of *1, if the SV is out of the valid range, it will be converted as follows. • If the value is smaller than 0, 0 is used. • If the value is bigger than 2000, 2000 is used.
Process value	PV	A feedback data from the control target to the A/D converter module	-50 to 2050*1 -32768 to 32767*2	In the case of *1, if the PV is out of the valid range, it will be converted as follows. • If the value is smaller than -50, -50 is used. • If the value is bigger than 2050, 2050 is used.
Automatic manipulated value	MV	A manipulated value calculated by PID operation. The value is output from the D/A converter module to the control target.	-50 to 2050*1 -32768 to 32767*2	—
Process value after filtering	PVf	A process value calculated by the following operational expression $PV_{fn} = PV_n + \alpha(PV_{fn-1} - PV_n)$	-50 to 2050*1 -32768 to 32767*2	—
Manual manipulated value	MV_{MAN}	Data output from the D/A converter module in manual mode	-50 to 2050*1 -32768 to 32767*2	In the case of *1, if the MV_{MAN} is out of the valid range, it will be converted as follows. • If the value is smaller than -50, -50 is used. • If the value is bigger than 2050, 2050 is used.
Manual/automatic selection	MAN/AUTO	Select the type of output data to the D/A converter module: manual manipulated value or automatic manipulated value. In manual mode, the automatic manipulated value remains unchanged.	0: Automatic manipulated value 1: Manual manipulated value	An error occurs if the set value is other than 0 or 1, and PID operation of the corresponding loop is not executed.
Alarm	ALARM	Checks whether the variation rate of the automatic manipulated value (MV) or process value (PV) is within the valid range or not. Once the value is set, it is held until the controller is reset.  (1) If the PV is out of the limited range, 1 is set to b0. (2) If the MV is out of the limited range, 1 is set to b1.	—	—

*1 When the PID limit is restricted

*2 When the PID limit is not restricted

Helpful functions

During PID operation by using PID control instructions, bumpless transfer and manipulated value upper/lower limit control are automatically executed.

Bumpless transfer

This function controls the manipulated value (MV) continuously when the control mode is switched from manual to automatic, or vice-versa. When the mode is switched, data are transferred between the MV storage areas for manual mode and automatic mode.

- From manual to automatic: MV in manual mode is transferred to the MV storage area for automatic mode.
- From automatic to manual: MV in automatic mode is transferred to the MV storage area for manual mode.

Switch the mode in I/O data. (☞ Page 1079 I/O data)

Point

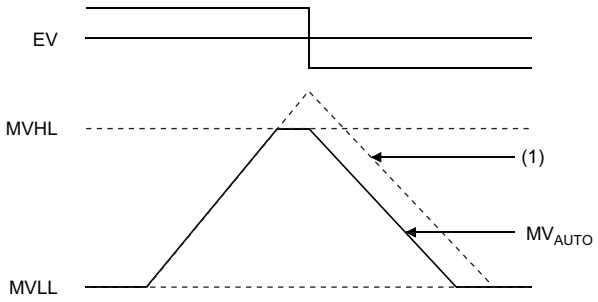
PID control is performed in each mode as described below.

- Automatic mode: The controller controls the target with the MV calculated by PID operation using the PID control instructions.
- Manual mode: The controller controls the target with the MV calculated without performing PID operation using the PID control instructions. Loops set in manual mode stores the process value (PV) in the set value storage area every sampling cycle.

Manipulated value upper/lower limits control

This function controls the upper or lower limit of the manipulated value (MV) calculated by PID operation. The function is only enabled in automatic mode and is not executed in manual mode.

The manipulated value (MV) calculated by PID operation can be limited within the range set by the manipulated value upper limit (MVHL) and manipulated value lower limit (MVLL). The following figure shows the operation of the function.



(1) MV_{AUTO} without limit control

The manipulated value upper limit (MVHL) and manipulated value lower limit (MVLL) can be set to each loop within the range of -50 to 2050 or within any desired range. The following values are set by default.

- Manipulated value upper limit: 2000
- Manipulated value lower limit: 0

An error occurs if the MVHL is smaller than the MVLL.

Transferring PV to SV in manual mode

This function executes the PIDCONT instruction even in manual mode. In manual mode, whether transferring the process value (PV) input from the A/D converter module to the set value (SV) storage area during execution of the PIDCONT instruction or not can be selected by turning on or off SM792 or SM794. SM792 and SM794 are set to off by default. The PV and SV are stored in the specified device or label in the I/O data area by using the PIDCONT instruction.

SM792, SM794	Operation
Off	• The PV is transferred to the SV storage device or label during execution of the PIDCONT instruction. • When the mode is switched to automatic mode, the MV output in manual mode is continued. • If the SV is changed after the mode is switched to automatic mode, control from the MV output to the SV is enabled.
On	• The PV is not transferred to the SV storage device or label during execution of the PIDCONT instruction. • When the mode is switched to automatic mode, control from the MV output in manual mode to the SV is enabled. • Before switching to automatic mode, store the SV to the SV storage device or label.

Point

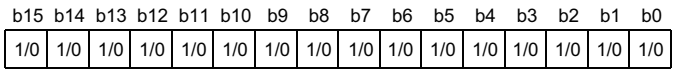
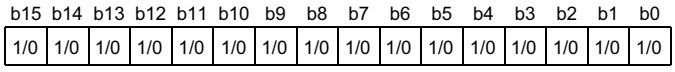
Depending on the on/off status of SM792 or SM794, the following differences apply when switching manual mode to automatic mode.

- When SM792 or SM794 is off, the PV is transferred to the SV storage device or label. Therefore, there is no difference between the PV and SV and an abrupt change does not occur in MV when the mode is switched. Note, however, that since the SV after the mode is switched differs from the target value in automatic mode, the user needs to change the SV step by step in the program so that it matches the target value.
- When SM792 or SM794 is on, the PV is not transferred to the SV storage device or label. Therefore, a difference exists between the PV and SV when the mode is switched. If the difference is large when the mode is switched, an abrupt change may occur in MV. Use this method in a system where the mode is switched when the PV has fully neared the SV. PID control in automatic mode can be executed immediately without the SV being changed step by step in the program.

Changing the setting range of PID control data and I/O data

The setting range of PID control data and I/O data can be changed as desired.

To change the range, turn on the bit corresponding to the target loop in SD792, SD793, SD794, and SD795.

Inexact differential	Exact differential	Operation
SD794	SD792	 b0: Loop 1 b1: Loop 2 ⋮ b14: Loop 15 b15: Loop 16
SD795	SD793	 b0: Loop 17 b1: Loop 18 ⋮ b14: Loop 31 b15: Loop 32

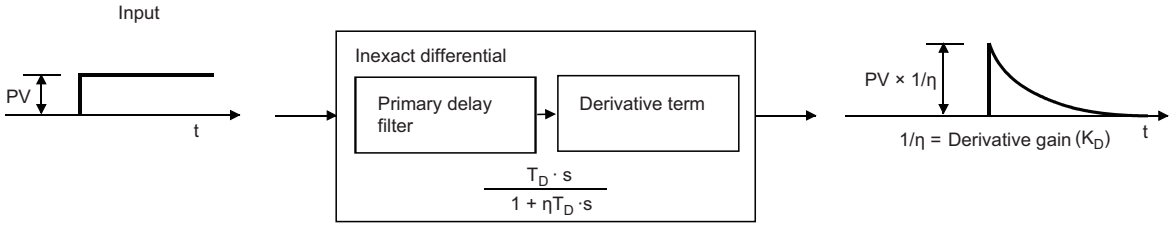
0: PID limit restricted (default)

1: PID limit not restricted

24.2 PID Control Instructions (Inexact Differential)

Inexact differential is PID control that applies a primary delay filter to the input of a differentiation term. Inexact differential is effective in the following cases:

- For control susceptible to high-frequency noise
- When energy effective to actuate an operation end is not provided when a step change occurs in an exact differential system

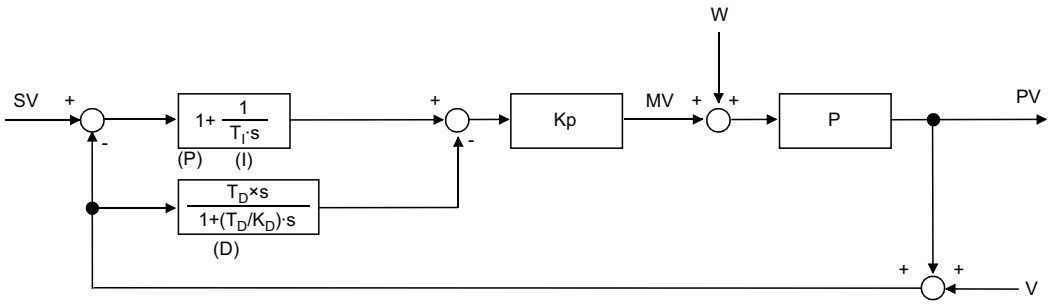


The following table summarizes the performance specifications of PID control instructions for inexact differential.

Item			When the PID limit is restricted	When the PID limit is not restricted
Number of PID control loops		—	32 maximum	
Sampling period		T_S	0.01 to 60.00s	
PID operation method		—	Process value differentiation type (inexact differential) (direct action/reverse action)	
PID constants setting range	Proportional constant	K_P	0.01 to 100.00	
	Integral constant	T_I	0.1 to 3000.0s	
	Derivative constant	T_D	0.00 to 300.00s	
	Derivative gain	K_D	0.00 to 300.00	
Set value setting range		SV	0 to 2000	-32768 to 32767
Process value setting range		PV	-50 to 2050	-32768 to 32767
Manipulated value output range		MV		

The following are the block diagram and operational expressions of PID operation.

- Block diagram of PID operation (inexact differential)



K_P: Gain
W: Disturbance
P: Controlled system
V: Detected noise

• Operational expressions

Action	Operational expression
Direct action	$EV_n = PV_{fn} - SV$ $\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} \cdot EV_n + D_n\}$ $D_n = \frac{T_D}{T_S + \frac{T_D}{K_D}}(PV_{fn} - 2PV_{fn-1} + PV_{fn-2}) + \frac{\frac{T_D}{K_D}}{T_S + \frac{T_D}{K_D}} \cdot D_{n-1}$ $MV_n = \Sigma\Delta MV$
Reverse action	$EV_n = SV - PV_{fn}$ $\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} \cdot EV_n + D_n\}$ $D_n = \frac{T_D}{T_S + \frac{T_D}{K_D}}(- PV_{fn} + 2PV_{fn-1} - PV_{fn-2}) + \frac{\frac{T_D}{K_D}}{T_S + \frac{T_D}{K_D}} \cdot D_{n-1}$ $MV_n = \Sigma\Delta MV$

The meaning of the symbols in the operational expressions is as follows.

Symbol	Meaning
EV_n	Deviation in the sampling period this time
EV_{n-1}	Deviation in the sampling period last time
SV	Set value
PV_{fn}	Process value of the sampling period this time (after filtering)
PV_{fn-1}	Process value of the sampling period last time (after filtering)
PV_{fn-2}	Process value of the sampling period second last time (after filtering)
ΔMV	Output variation amount
MV_n	Manipulated value this time
D_n	Derivative term this time
D_{n-1}	Derivative term of the sampling period last time
T_S	Sampling period
K_P	Proportional constant
T_I	Integral constant
T_D	Derivative constant
K_D	Derivative gain

The PV_{fn} is calculated by using the following operational expression. If the input filter coefficient is not set, the value will be the same as the input process value (PV).

$PV_{fn} = PV_n + \alpha(PV_{fn-1} - PV_n)$

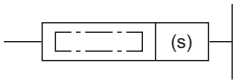
PV_n: Process value for the sampling period this time, α: Filter coefficient, PV_{fn-1}: Process value for the sampling period last time (after filtering)

PV_{fn} is stored in the I/O data area. (☞ Page 1079 I/O data)

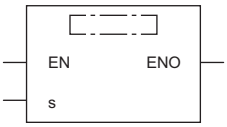
Registering the PID control data to the controller

S(P).PIDINIT

These instructions store the PID control data by the number of loops used that is set in the specified device number and later altogether in the controller module.

Ladder	ST
	ENO:=S_PIDINIT(EN,s); ENO:=SP_PIDINIT(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
S.PIDINIT	
SP.PIDINIT	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the PID control data is set	—	Word	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions store the PID control data by the number of loops used that is set in the device number specified (s) and later altogether in the controller to enable PID control. (☞ Page 1075 PID control data)
- The PID control data are assigned as follows.

Item	PID control data	
Common to all loops	(s)+0	Number of loops used
	(s)+1	Number of loops in one scan
For loop No.1 (14 words)	(s)+2	Operational expression selection
	(s)+3	Sampling period (T_S)
	(s)+4	Proportional constant (K_P)
	(s)+5	Integral constant (T_I)
	(s)+6	Derivative constant (T_D)
	(s)+7	Filter coefficient (α)
	(s)+8	Manipulated value lower limit (MVLL)
	(s)+9	Manipulated value upper limit (MVHL)
	(s)+10	Manipulated value variation rate limit (ΔMVL)
	(s)+11	Process value variation rate limit (ΔPVL)
	(s)+12	Fixed to "0". (An error occurs if a value other than "0" is specified.)
	(s)+13	Derivative gain (K_D)
	(s)+14	Fixed to "0". (An error occurs if a value other than "0" is specified.)
	(s)+15	Fixed to "0". (An error occurs if a value other than "0" is specified.)
⋮	⋮	⋮
For loop No.n (14 words)*1	(s)+(m+0)	Operational expression selection
	(s)+(m+1)	Sampling period (T_S)
	(s)+(m+2)	Proportional constant (K_P)
	(s)+(m+3)	Integral constant (T_I)
	(s)+(m+4)	Derivative constant (T_D)
	(s)+(m+5)	Filter coefficient (α)
	(s)+(m+6)	Manipulated value lower limit (MVLL)
	(s)+(m+7)	Manipulated value upper limit (MVHL)
	(s)+(m+8)	Manipulated value variation rate limit (ΔMVL)
	(s)+(m+9)	Process value variation rate limit (ΔPVL)
	(s)+(m+10)	Fixed to "0". (An error occurs if a value other than "0" is specified.)
	(s)+(m+11)	Derivative gain (K_D)
	(s)+(m+12)	Fixed to "0". (An error occurs if a value other than "0" is specified.)
	(s)+(m+13)	Fixed to "0". (An error occurs if a value other than "0" is specified.)

*1 $m = (n-1) \times 14 + 2$

(1) Fixed to "0". An error results if a value other than "0" is specified.

- The number of device points used for PID control data setting is calculated by the following formula.

Number of device points = $2 + 14 \times n$ (n: number of loops used)

- Specify each data in binary.
- If the total number of device points for the number of loops used exceeds the last device number, an error occurs and no processing is performed.
- If the S(P).PIDINIT instruction is executed at two or more locations during a single scan, the setting value of the S(P).PIDINIT instruction executed nearest to the S(P).PIDCONT instruction will be valid.
- Execute the S(P).PIDINIT instruction before execution of the S(P).PIDCONT instruction. To perform PID control, the S(P).PIDINIT instruction must be executed.

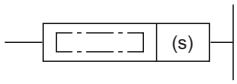
Operation error

Error code (SD0)	Description
3285H	Out-of-range data is set in the device specified by (s). • The value set for the PID control data is out of the setting range. • (Number of loops used) < (Number of loops executed in one scan) • (Manipulated value upper limit) < (Manipulated value lower limit) • The area fixed to 0 in the PID control data is not 0.

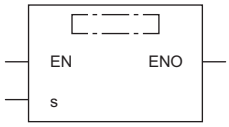
Performing PID operation

S(P).PIDCONT

These instructions measure the sampling cycle and perform PID operation when the execution command turns on.

Ladder	ST
	ENO:=S_PIDCONT(EN,s); ENO:=SP_PIDCONT(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
S.PIDCONT	
SP.PIDCONT	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device assigned to the I/O data area	—	Word	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- The S(P).PIDCONT instructions measure the sampling cycle and perform PID operation.
- Based on the setting value (SV) and process value (PV) in the I/O data area allocated to the device number specified by (s) and later, these instructions perform PID operation and store the operation result in the automatic manipulated value (MV) area in the I/O data area.
- The S(P).PIDCONT instructions perform PID operation when it is executed for the first time after a lapse of the specified sampling cycle.
- During PID control, be sure to turn on the control command to allow the S(P).PIDCONT instruction to be executed every scan. Failure to execute the instruction every scan disables PID operation in normal sampling cycles. The S(P).PIDCONT instruction cannot be executed more than once in a single scan. Executing the instruction more than once in a single scan disables PID operation in normal sampling cycles.
- The S(P).PIDCONT instruction cannot be written and used in interrupt programs. Writing an S(P).PIDCONT instruction in the interrupt program disables PID operation in normal sampling cycles.
- In (s), specify the head of the device number specified in the I/O data area. (☞ Page 1079 I/O data)
- If a file register is specified as an I/O data area, do not apply memory protection for the file register. If memory protection is applied, normal PID operation is disabled although no error results.
- The I/O data are assigned as follows.

Item		I/O data	
Write		(s)+0	Initial processing flag
Read/write disabled		(s)+1 ⋮ (s)+9	Work area for PID control (system use only)
I/O data area for loop No.1 (23 words)	Write	(s)+10	Set value (SV)
		(s)+11	Process value (PV)
	Read	(s)+12	Automatic manipulated value (MV)
		(s)+13	Process value after filtering (PVf)
	Write	(s)+14	Manual manipulated value (MV_{MAN})
		(s)+15	Manual/automatic selection (MAN/AUTO)
	Read/write	(s)+16	Alarm (ALARM)
	Read/write disabled	(s)+17 ⋮ (s)+32	Work area for loop No.1 (system use only)
⋮	⋮	⋮	⋮
I/O data area for loop No.n (23 words)*1	Write	(s)+(m+0)	Set value (SV)
		(s)+(m+1)	Process value (PV)
	Read	(s)+(m+2)	Automatic manipulated value (MV)
		(s)+(m+3)	Process value after filtering (PVf)
	Write	(s)+(m+4)	Manual manipulated value (MV_{MAN})
		(s)+(m+5)	Manual/automatic selection (MAN/AUTO)
	Read/write	(s)+(m+6)	Alarm (ALARM)
	Read/write disabled	(s)+(m+7) ⋮ (s)+(m+22)	Work area for loop No.n (system use only)

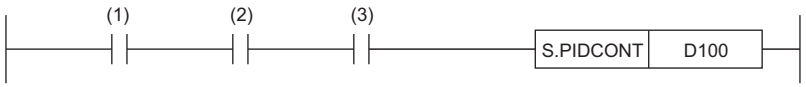
*1 $m = (n-1) \times 23 + 10$

- The number of device points used for I/O data setting is calculated by the following formula.

Number of device points = 10 +23 × n (n: number of loops used)

- Specify each data in binary.
- The initial processing flag sets the processing to be performed at the start of PID operation.
  - Initial operation processing is performed assuming that the sampling cycle that has been set has been reached.
  - If the initial processing flag is 0, PID operations for the number of loops used are performed altogether in a single scan. If it is not 0, PID operations for the number of loops used are divided and processed in several scans, and sampling is started sequentially from the loop that has completed initial processing. The number of processing loops per scan is the number of execution loops per scan that has been set.

- Write data to the I/O data "Write" area by users with the program. Users can read data from the I/O data "Read" area with the program. Never attempt to write data to the area indicated by "Read/write disabled" or "Read"; otherwise, normal operation can no longer be performed. Note that, when starting control from the initial status, the data areas must be cleared by the program.
- If the total number of device points for the number of loops used exceeds the last device number, an error occurs and no processing is performed.
- Even when the manual manipulated value (MV_{MAN}) is output in manual mode, execute the S(P).PIDCONT instruction every scan. Unless the S(P).PIDCONT instruction is executed, the bumpless function cannot be performed.
- Apply an interlock using the READY signal of each module so that the S(P).PIDCONT instruction is executed only when the A/D converter module used to obtain the process value (PV) and the D/A converter module used to output the manipulated value (MV) are normal.



(1) Control command
(2) READY signal of the A/D converter module
(3) READY signal of the D/A converter module

If the instruction is executed when these modules are not normal, PID operation cannot be performed normally as the result of failure in normal acquisition of process values (PV) or in normal output of manipulated values (MV).

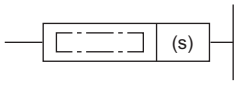
Operation error

Error code (SD0)	Description
3285H	The value of the data set in the I/O data area specified by (s) is out of the setting range.
3292H	The S(P).PIDINIT instruction is not executed before the S(P).PIDCONT instruction.

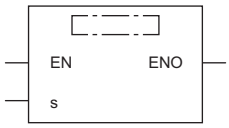
Stopping the operation of specified loop number

S(P).PIDSTOP

These instructions stop the PID operation of the specified loop number.

Ladder	ST
	ENO:=S_PIDSTOP(EN,s); ENO:=SP_PIDSTOP(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
S.PIDSTOP	
SP.PIDSTOP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Loop number to be stopped	1 to 32	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	○	—	—	—

Processing details

- These instructions stop the PID operation of the loop number in the device specified by (s). The loop stopped by the S(P).PIDSTOP instruction does not restart PID operation even if the S(P).PIDINIT instruction is executed.
- Each instruction holds operation data while the loop is stopped.

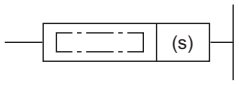
Operation error

Error code (SD0)	Description
3285H	Out-of-range data is set in the device specified by (s). • The specified loop number does not exist. • The specified value is other than 1 to 32.
3292H	The S(P).PIDINIT and S(P).PIDCONT instructions are not executed before the S(P).PIDSTOP instruction.

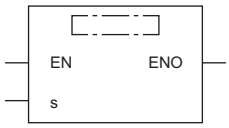
Starting the operation of specified loop number

S(P).PIDRUN

These instructions start the PID operation of the specified loop number.

Ladder	ST
	ENO:=S_PIDRUN(EN,s); ENO:=SP_PIDRUN(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
S.PIDRUN	
SP.PIDRUN	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Loop number to be stopped	1 to 32	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	○	—	—	—

Processing details

- These instructions start the PID operation of the loop number in the device specified by (s). These instructions are used to re-execute the PID operation of the loop number that has been stopped by the S(P).PIDSTOP instruction.
- The S(P).PIDRUN instruction, if executed for a loop number already in progress of PID operation, performs no processing.

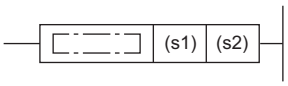
Operation error

Error code (SD0)	Description
3285H	The loop number specified by (s) does not exist.
	(s) is outside the range from 1 to 32.
3292H	The S(P).PIDINIT and S(P).PIDCONT instructions are not executed before the S(P).PIDRUN instruction.

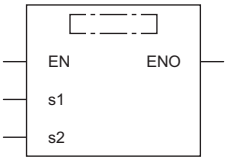
Changing the parameters of specified loop number

S(P).PIDPRMW

These instructions change the operation parameter of the specified loop number to the PID control data stored in the specified device number and later.

Ladder	ST
	ENO:=S_PIDPRMW(EN,s1,s2); ENO:=SP_PIDPRMW(EN,s1,s2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.PIDPRMW	
SP.PIDPRMW	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Loop number to be changed	1 to 32	16-bit unsigned binary	ANY16
(s2)	Start device where the change-target PID control data is stored	—	Word	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions changes the operation parameter of the loop number in the device specified by (s1) to the PID control data stored in the device number specified by (s2) and later.
- The following figure shows the configuration of the PID control data in the device specified by (s2) and later.

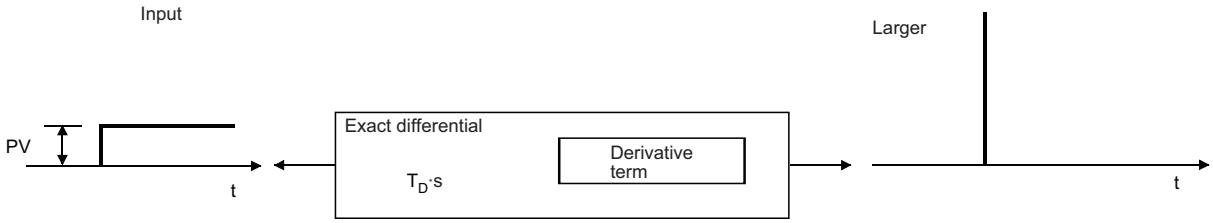
PID control data	
(s2)+0	Operational expression selection
(s2)+1	Sampling period (T_S)
(s2)+2	Proportional constant (K_P)
(s2)+3	Integral constant (T_I)
(s2)+4	Derivative constant (T_D)
(s2)+5	Filter coefficient (α)
(s2)+6	Manipulated value lower limit (MVLL)
(s2)+7	Manipulated value upper limit (MVHL)
(s2)+8	Manipulated value variation rate limit (ΔMVL)
(s2)+9	Process value variation rate limit (ΔPVL)
(s2)+10	0
(s2)+11	Derivative gain (K_D)
(s2)+12	0
(s2)+13	0

Operation error

Error code (SD0)	Description
3285H	Out-of-range data is set in the device specified by (s1). • The specified loop number does not exist. • The specified value is other than 1 to 32.
	Out-of-range data is set in the device specified by (s2). • The PID control data is out of the setting range. • The PID control data in the devices specified by (s2)+10, (s2)+12, and (s2)+13 is not 0.
3292H	The S(P).PIDINIT instruction is not executed before the S(P).PIDPRMW instruction.

24.3 PID Control Instructions (Exact Differential)

Exact differential is PID control that uses the input of a differential term as it is.

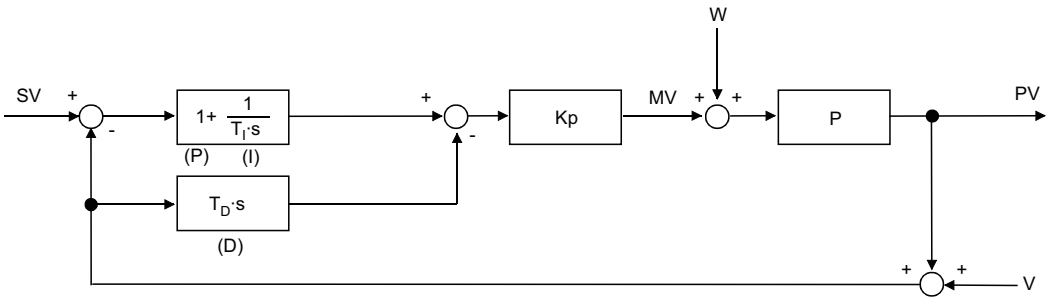


The following table summarizes the performance specifications of PID control instructions for exact differential.

Item			When the PID limit is restricted	When the PID limit is not restricted
Number of PID control loops		—	32 maximum	
Sampling period		T_S	0.01 to 60.00s	
PID operation method		—	Process value differentiation type (exact differential) (direct action/reverse action)	
PID constants setting range	Proportional constant	K_P	0.01 to 100.00	
	Integral constant	T_I	0.1 to 3000.0s	
	Derivative constant	T_D	0.00 to 300.00s	
Set value setting range		SV	0 to 2000	-32768 to 32767
Process value setting range		PV	-50 to 2050	-32768 to 32767
Manipulated value output range		MV		

The following are the block diagram and operational expressions of PID operation.

- Block diagram of PID operation (exact differential)



K_P: Gain
W: Disturbance
P: Controlled system
V: Detected noise

• Operational expressions

Action	Operational expression
Direct action	$EV_n = PV_{fn} - SV$ $\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} \cdot EV_n + D_n\}$ $D_n = \frac{T_D}{T_S}(PV_{fn} - 2PV_{fn-1} + PV_{fn-2})$ $MV_n = \Sigma\Delta MV$
Reverse action	$EV_n = SV - PV_{fn}$ $\Delta MV = K_p\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I} \cdot EV_n + D_n\}$ $D_n = \frac{T_D}{T_S}(- PV_{fn} + 2PV_{fn-1} - PV_{fn-2})$ $MV_n = \Sigma\Delta MV$

The meaning of the symbols in the operational expressions is as follows.

Symbol	Meaning
EV_n	Deviation in the sampling period this time
EV_{n-1}	Deviation in the sampling period last time
SV	Set value
PV_{fn}	Process value of the sampling period this time (after filtering)
PV_{fn-1}	Process value of the sampling period last time (after filtering)
PV_{fn-2}	Process value of the sampling period second last time (after filtering)
ΔMV	Output variation amount
MV_n	Manipulated value this time
D_n	Derivative term this time
T_S	Sampling period
K_P	Proportional constant
T_I	Integral constant
T_D	Derivative constant

The PV_{fn} is calculated by using the following operational expression. If the input filter coefficient is not set, the value will be the same as the input process value (PV).

$PV_{fn} = PV_n + \alpha(PV_{fn-1} - PV_n)$

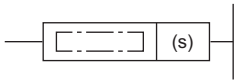
PV_n: Process value for the sampling period this time, α: Filter coefficient, PV_{fn-1}: Process value for the sampling period last time (after filtering)

PV_{fn} is stored in the I/O data area. (☞ Page 1079 I/O data)

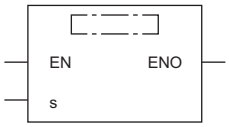
Registering the PID control data to the controller

PIDINIT(P)

These instructions store the PID control data by the number of loops used that is set in the specified device number and later altogether in the controller module.

Ladder	ST
	ENO:=PIDINIT(EN,s); ENO:=PIDINITP(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
PIDINIT	
PIDINITP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where the PID control data is set	—	Word	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions store the PID control data by the number of loops used that is set in the device number specified (s) and later altogether in the controller to enable PID control. (☞ Page 1075 PID control data)
- The PID control data are assigned as follows.

Item	PID control data	
Common to all loops	(s)+0	Number of loops used
	(s)+1	Number of loops in one scan
For loop No.1 (10 words)	(s)+2	Operational expression selection
	(s)+3	Sampling period (T_S)
	(s)+4	Proportional constant (K_P)
	(s)+5	Integral constant (T_I)
	(s)+6	Derivative constant (T_D)
	(s)+7	Filter coefficient (α)
	(s)+8	Manipulated value lower limit (MVLL)
	(s)+9	Manipulated value upper limit (MVHL)
	(s)+10	Manipulated value variation rate limit (ΔMVL)
	(s)+11	Process value variation rate limit (ΔPVL)
⋮	⋮	⋮
For loop No.n (10 words)*1	(s)+(m+0)	Operational expression selection
	(s)+(m+1)	Sampling period (T_S)
	(s)+(m+2)	Proportional constant (K_P)
	(s)+(m+3)	Integral constant (T_I)
	(s)+(m+4)	Derivative constant (T_D)
	(s)+(m+5)	Filter coefficient (α)
	(s)+(m+6)	Manipulated value lower limit (MVLL)
	(s)+(m+7)	Manipulated value upper limit (MVHL)
	(s)+(m+8)	Manipulated value variation rate limit (ΔMVL)
	(s)+(m+9)	Process value variation rate limit (ΔPVL)

*1 $m = (n-1) \times 10 + 2$

- The number of device points used for PID control data setting is calculated by the following formula.

Number of device points = $2 + 10 \times n$ (n: number of loops used)

- Specify each data in binary.
- If the total number of device points for the number of loops used exceeds the last device number, an error occurs and no processing is performed.
- If the PIDINIT(P) instruction is executed at two or more locations during a single scan, the setting value of the PIDINIT(P) instruction executed nearest to the PIDCONT(P) instruction will be valid.
- Execute the PIDINIT(P) instruction before execution of the PIDCONT(P) instruction. To perform PID control, the PIDINIT(P) instruction must be executed.

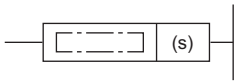
Operation error

Error code (SD0)	Description
3285H	Out-of-range data is set in the device specified by (s). • The value set for the PID control data is out of the setting range. • (Number of loops used) < (number of execution loops in one scan) • (Manipulated value upper limit) < (Manipulated value lower limit)

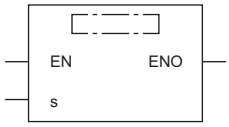
Performing PID operation

PIDCONT(P)

These instructions measure the sampling cycle and perform PID operation when the execution command turns on.

Ladder	ST
	ENO:=PIDCONT(EN,s); ENO:=PIDCONTP(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
PIDCONT	
PIDCONTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device assigned to the I/O data area	—	Word	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- The PIDCONT(P) instructions measure the sampling cycle and perform PID operation.
- Based on the setting value (SV) and process value (PV) in the I/O data area allocated to the device number specified by (s) and later, these instructions perform PID operation and store the operation result in the automatic manipulated value (MV) area in the I/O data area.
- The PIDCONT(P) instructions perform PID operation when it is executed for the first time after a lapse of the specified sampling cycle.
- During PID control, be sure to turn on the control command to allow the PIDCONT(P) instruction to be executed every scan. Failure to execute the instruction every scan disables PID operation in normal sampling cycles. The PIDCONT(P) instruction cannot be executed more than once in a single scan. Executing the instruction more than once in a single scan disables PID operation in normal sampling cycles.
- The PIDCONT(P) instruction cannot be written and used in interrupt programs. Writing a PIDCONT(P) instruction in the interrupt program disables PID operation in normal sampling cycles.
- In (s), specify the head of the device number specified in the I/O data area. (☞ Page 1079 I/O data)
- If a file register is specified as an I/O data area, do not apply memory protection for the file register. If memory protection is applied, normal PID operation is disabled although no error results.
- The I/O data are assigned as follows.

Item		I/O data	
Write		(s)+0	Initial processing flag
Read/write disabled		(s)+1 ⋮ (s)+9	Work area for PID control (system use only)
I/O data area for loop No.1 (18 words)	Write	(s)+10	Set value (SV)
		(s)+11	Process value (PV)
	Read	(s)+12	Manual manipulated value (MV)
		(s)+13	Process value after filtering (PVf)
	Write	(s)+14	Manual manipulated value (MV_{MAN})
		(s)+15	Manual/automatic selection (MAN/AUTO)
	Read/write	(s)+16	Alarm (ALARM)
	Read/write disabled	(s)+17 ⋮ (s)+27	Work area for loop No.1 (system use only)
⋮	⋮	⋮	⋮
I/O data area for loop No.n (18 words)*1	Write	(s)+(m+0)	Set value (SV)
		(s)+(m+1)	Process value (PV)
	Read	(s)+(m+2)	Automatic manipulated value (MV)
		(s)+(m+3)	Process value after filtering (PVf)
	Write	(s)+(m+4)	Manual manipulated value (MV_{MAN})
		(s)+(m+5)	Manual/automatic selection (MAN/AUTO)
	Read/write	(s)+(m+6)	Alarm (ALARM)
	Read/write disabled	(s)+(m+7) ⋮ (s)+(m+17)	Work area for loop No.n (system use only)

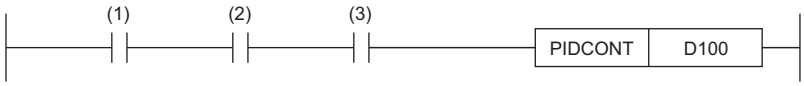
*1 m = (n-1) × 18 + 10

- The number of device points used for I/O data setting is calculated by the following formula.

Number of device points = 10 + 18 × n (n: number of loops used)

- Specify each data in binary.
- The initial processing flag sets the processing to be performed at the start of PID operation.
  - Initial operation processing is performed assuming that the sampling cycle that has been set has been reached.
  - If the initial processing flag is 0, PID operations for the number of loops used are performed altogether in a single scan. If it is not 0, PID operations for the number of loops used are divided and processed in several scans, and sampling is started sequentially from the loop that has completed initial processing. The number of processing loops per scan is the number of execution loops per scan that has been set.

- Write data to the I/O data "Write" area by users with the program. Users can read data from the I/O data "Read" area with the program. Never attempt to write data to the area indicated by "Read/write disabled" or "Read"; otherwise, normal operation can no longer be performed. Note that, when starting control from the initial status, the data areas must be cleared by the program.
- Even when the manual manipulated value (MV_{MAN}) is output in manual mode, execute the PIDCONT(P) instruction every scan. Unless the PIDCONT(P) instruction is executed, the bumpless function cannot be performed.
- Apply an interlock using the READY signal of each module so that the PIDCONT(P) instruction is executed only when the A/D converter module used to obtain the process value (PV) and the D/A converter module used to output the manipulated value (MV) are normal. If the instruction is executed when these modules are not normal, PID operation cannot be performed normally as the result of failure in normal acquisition of process values (PV) or in normal output of manipulated values (MV).



(1) Control command
(2) READY signal of the A/D converter module
(3) READY signal of the D/A converter module

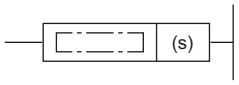
Operation error

Error code (SD0)	Description
3285H	The value of the data set in the I/O data area specified by (s) is out of the setting range.
3292H	The PIDINIT(P) instruction is not executed before the PIDCONT(P) instruction.

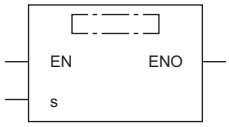
Stopping the operation of specified loop number

PIDSTOP(P)

These instructions stop the PID operation of the loop number in the device specified by (s).

Ladder	ST
	ENO:=PIDSTOP(EN,s); ENO:=PIDSTOPP(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
PIDSTOP	
PIDSTOPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Loop number to be stopped	1 to 32	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	—	—	—	○	○	—	—	—

Processing details

- These instructions stop the PID operation of the loop number in the device specified by (s). The loop stopped by the PIDSTOP(P) instruction does not restart PID operation even if the PIDINIT(P) instruction is executed.
- Each instruction holds operation data while the loop is stopped.

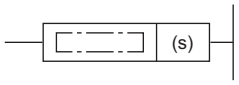
Operation error

Error code (SD0)	Description
3285H	Out-of-range data is set in the device specified by (s). • The specified loop number does not exist. • The specified value is other than 1 to 32.
3292H	The PIDINIT(P) and PIDCONT(P) instructions are not executed before the PIDSTOP(P) instruction.

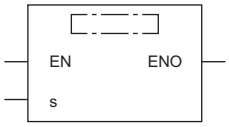
Starting the operation of specified loop number

PIDRUN(P)

These instructions start the PID operation of the specified loop number.

Ladder	ST
	ENO:=PIDRUN(EN,s); ENO:=PIDRUNP(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
PIDRUN	
PIDRUNP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Loop number to be stopped	1 to 32	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	—	—	—	○	○	—	—	—

Processing details

- These instructions start the PID operation of the loop number in the device specified by (s). These instructions are used to re-execute the PID operation of the loop number that has been stopped by the PIDSTOP(P) instruction.
- The PIDRUN(P) instruction, if executed for a loop number already in progress of PID operation, performs no processing.

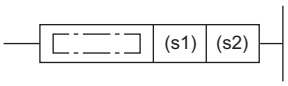
Operation error

Error code (SD0)	Description
3285H	Out-of-range data is set in the device specified by (s). • The specified loop number does not exist. • The specified value is other than 1 to 32.
3292H	The PIDINIT(P) and PIDCONT(P) instructions are not executed before the PIDRUN(P) instruction.

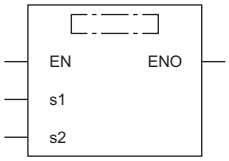
Changing the parameters of specified loop number

PIDPRMW(P)

These instructions change the operation parameter of the specified loop number to the PID control data stored in the specified device number and later.

Ladder	ST
	ENO:=PIDPRMW(EN,s1,s2); ENO:=PIDPRMWP(EN,s1,s2);

FBD/LD



■Execution condition

Instruction	Execution condition
PIDPRMW	
PIDPRMWP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Loop number to be changed	1 to 32	16-bit unsigned binary	ANY16
(s2)	Start device where the change-target PID control data is stored	—	Word	ANY16*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions changes the operation parameter of the loop number in the device specified by (s1) to the PID control data stored in the device number specified by (s2) and later.
- The following figure shows the configuration of the PID control data in the device specified by (s2) and later.

PID control data	
(s2)+0	Operational expression selection
(s2)+1	Sampling period (T_S)
(s2)+2	Proportional constant (K_P)
(s2)+3	Integral constant (T_I)
(s2)+4	Derivative constant (T_D)
(s2)+5	Filter coefficient (α)
(s2)+6	Manipulated value lower limit (MVLL)
(s2)+7	Manipulated value upper limit (MVHL)
(s2)+8	Manipulated value variation rate limit (ΔMVL)
(s2)+9	Process value variation rate limit (ΔPVL)

Operation error

Error code (SD0)	Description
3285H	Out-of-range data is set in the device specified by (s1). The specified loop number does not exist. The specified value is other than 1 to 32.
	The PID control data in the device specified by (s2) is out of the setting range.
3292H	The PIDINIT(P) instruction is not executed before the PIDPRMW(P) instruction.

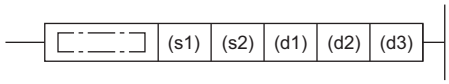
25 ADD-ON

25.1 Add-on Instructions

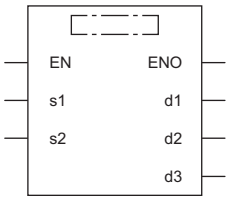
Add-on execution instruction

ADDONEXE(P)

This instruction specifics the execution of installed add-ons.

Ladder	ST
	ENO:= ADDONEXE (EN, s1, s2, d1, d2, d3); ENO:= ADDONEXEP (EN, s1, s2, d1, d2, d3);

FBD/LD



■Execution condition

Instruction	Execution condition
ADDONEXE	
ADDONEXEP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Add-on ID (Only SQ type is supported)*1	1 to 64	Double word	ANY32_S
(s2)	Start number of a device where the request data is stored	—	Word	ANY16
(d1)	Start number of a device where the response data is stored	—	Word	ANY16
(d2)	Specifies a bit device that turns on upon completion of processing.*2	—	Bit	ANYBIT_ARRAY
(d3)	Specifies a device where the completion status is stored upon completion of the instruction.	—	Word	ANY16

*1 For more details on the add-on types, check the user's manual for the controller used.

*2 When the instruction completes with an error, (d2)+1 also turns on.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (P)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○*1	—	—	—	—	○	○	—	—	○
(s2)	—	—	○*1	—	—	—	—	○	—	—	—	○
(d1)	—	—	○*1	—	—	—	—	○	—	—	—	○
(d2)	○*2	—	○*3	—	—	—	—	—	—	—	—	○
(d3)	—	—	○*1	—	—	—	—	○	—	—	—	○

*1 FD cannot be used.
*2 FX and FY cannot be used.
*3 T, ST, C, and FD cannot be used.

The following devices are used in operand (s2).

Operand	Item	Description	Setting range	Set by
(s2)	Allowable amount of response data	Sets the allowable number of words of response data.	0 to 8192	User
(s2)+1	Request data length	Specifies the number of words of the request data length.	0 to 8192	User
(s2)+2	Request data	Specifies the request data.	—	—

The following devices are used in operand (d1).

Operand	Item	Description	Setting range	Set by
(d1)	Response data length	Specifies the number of words of the response data length.	0 to 8192	System
(d1)+1	Response data	Specifies the response data.	—	System

The following devices are used in operand (d2).

Operand	Item	Description	Setting range	Set by
(d2)	Completion device	Turns on when the instruction execution is completed.	0 to 1	System
(d2)+1	Error completion device	Turns on when the instruction execution is completed with an error.	0 to 1	System

Processing details

- These instructions specify the execution of installed add-ons
- The request data stored in the device specified by (s2) and later is handed over to the function of the add-on associated with the add-on ID specified by (s1), and the response data is stored in the device specified by (d1)+1 and later. However, if the received response data is larger than the allowed number of response data specified in (s2), only the allowed number of response data will be stored and the remaining will be discarded. (Dedicated instruction will be completed successfully.) In this case, the response data length (d1) will be the number of data actually stored.
- The completion of this instruction can be checked with (d2) and (d2)+1.

Operand	Item	Description
(d2)	Completion device	This device turns off at the instruction execution and turns on upon completion of the add-on execution. After the add-on execution has been completed, the device remains on. If this device turns off after instruction execution, the add-on execution has not been completed. The device is used for the system at the instruction execution. Do not change the value by user operation.
(d2)+1	Completion device	This device turns off at the instruction execution and remains off when the add-on is completed successfully. When the add-on is completed with an error, the device turns on. After the add-on has been completed with an error, the device remains on. The device is used for the system at the instruction execution. Do not change the value by user operation.

- The execution result of this instruction can be checked in (d3).

Operand	Item	Description
(d3)	Completion status	0000H is stored at the instruction execution and remains unchanged when the add-on is completed successfully. When the add-on is completed with an error, a completion status that corresponds to the error is stored. After the add-on has been completed with an error, the status remains as it is. Clear it as needed. A completion status that corresponds to the error remains. The device is used for the system at the instruction execution. Do not change the value by user operation.

Precautions

- Set (s2) according to the processing to be executed. The content of (d1) varies depending on the processing to be executed.
- Specify operand even when request data and response data are not required.
- Do not change each specified request data until the execution of this instruction is completed.
- Specify the label "Type_AddonLibraryName_System.Md.AddonMemory.AddonID" to specify an add-on ID. For more details on labels, check the user's manual for the controller used.
- This instruction varies the number of scans until completion based on the return value of the add-on being executed. The return value of the add-on indicates whether processing is complete or incomplete. The execution of this instruction is also complete when the scan in which the add-on has completed is performed.
- During the add-on execution, ensure that the contact is turned on and this instruction is executed at every scan. If the contact is turned off during the add-on execution, the add-on may not be completed properly.
- If the add-on execution is completed after the ADDONEXE instruction has been executed and if the contact is on in the next scan, the add-on will be executed again. If the add-on execution is completed after the ADDONEXEP instruction has been executed and if the contact is on in the next scan, the add-on will not be executed. To execute the next add-on, turn the contact off once, and then turn it on again.
- Use the request data (s2) in every scan to execute the add-on during the add-on execution.

Operation error

Error code (SD0)	Description
3285H	The add-on ID specified by (s1) is outside the setting range.
	The allowable number of response data specified by (s2) is outside the setting range.
	The request data length specified by (s2+1) is outside the setting range.

If this instruction completes with an error, the following error code is stored in the device specified by (d3).

Error code	Description
1000H	No add-on corresponding to the add-on ID specified by (s1) exists.

PART 5 NETWORK INSTRUCTIONS

This part consists of the following chapter.
For the dedicated instruction in using the intelligent function module with the MX-R model, refer to the following.
MELSEC iQ-R Programming Manual (Module Dedicated Instructions)

26 Ethernet INSTRUCTIONS

26 Ethernet INSTRUCTIONS

For the dedicated instruction in using the intelligent function module with the MX-R model, refer to the following.
MELSEC iQ-R Programming Manual (Module Dedicated Instructions)

26.1 Open/Close Processing Instructions

Point

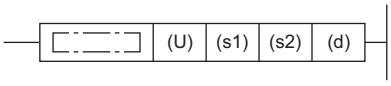
For connections where any of the following instructions is being executed, a request is ignored even if the following multiple instructions are attempted to be executed.
At this time, as long as the execution conditions of the instruction executed later are met, the request is executed after the first instruction being executed completes. SM699 (Dedicated instruction skip flag) turns on while the request is ignored.

- SOCOPEN instruction
- SOCCLOSE instruction
- SOCSND instruction
- SOCRCV instruction
- ECPRTCL instruction

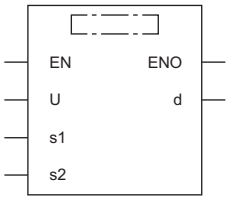
Opening a connection

GP.SOCOPEN, SP.SOCOPEN

This instruction opens the specified connection.

Ladder	ST
	ENO:=GP_SOCOPEN(EN,U,s1,s2,d); ENO:=SP_SOCOPEN(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.SOCOPEN SP.SOCOPEN	

Setting data

■Description, range, data type

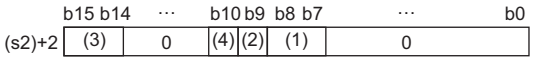
Operand		Description	Range	Data type	Data type (label)
(U)	GP.SOCOPEN	Start I/O number of target (Specifies the first 3 digits of the start I/O number)	3E0H	16-bit unsigned binary	ANY16
	SP.SOCOPEN	Dummy	—	String	ANYSTRING_SINGLE
(s1)		Connection number	1 to 64	16-bit unsigned binary	ANY16
(s2)	GP.SOCOPEN	Start device where control data is stored	Refer to the control data.	Device	ANY16
	SP.SOCOPEN			Word	ANY16_ARRAY (Number of elements: 10)
(d)		Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others (U)
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	GP.SOCOPEN	—	—	○	—	—	—	—	○	○	—	—	○
	SP.SOCOPEN	—	—	○	—	—	—	—	○	—	—	○	○
(s1)		—	—	○	—	—	—	—	○	○	—	—	—
(s2)		—	—	○	—	—	—	—	○	—	—	—	—
(d)		○	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution/completion type	Specify whether to use the parameter value set by the engineering tool or the value set in (s2)+2 to (s2)+9 of control data for opening a connection. • 0000H: Performs open processing according to the "open setting" by the engineering tool. • 8000H: Performs open processing according to the setting in (s2)+2 to (s2)+9 of control data.*1	0000H 8000H	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+2	Application setting area	Specify the application of a connection.  (1) Communication method (protocol) (b8 and 7) • 00: TCP • 01: TLS • 10: UDP • 11:DTLS (2) Socket communications function procedure (b9) • 1: No procedure (fixed) (3) Open method (b15, b14) • 00: Active open or UDP • 10: Unpassive open • 11: Fullpassive open (4) Predefined protocol setting (b10) • 0: Do not use the predefined protocol support function. (Use the socket communications function.) • 1: Use the predefined protocol support function.	—	User
+3	Own station port number	Specify the port number of the own station.	1 to 4999, 5010 to 65534 (0001H to 1387H, 1392H to FFFEH)	User
+4 +5	IP address of external device*2	Specify the IP address of an external device. To enable broadcast communications, specify FFFFFFFFH.	00000001H to FFFFFFFFH	User
+6	Destination port number*2	Specify the destination port number. When receiving data from all port numbers, specify FFFFH.*3	1 to 65535 (0001H to FFFFH)	User
+7 to +9	—	Reserved	—	System

*1 The setting items that are not included in the control data are as follows.
- Alive check: KeepAlive (for TCP/TLS), UDP (for UDP/DTLS)
- Connection monitoring timer setting: Invalid

*2 In Unpassive open mode, the IP address of the external device and destination port number are ignored.

*3 If 65535 (FFFFH) is specified for the port number, data cannot be sent. To send data, specify a value within the range of 1 to 65534 (1 to FFFEH).

Point

Since port numbers 0001H to 03FFH are generally reserved port numbers (WELL KNOWN PORT NUMBERS) and F000H to FFFEH are used dynamically by the other functions, use of port numbers 0400H to 1387H and 1392H to EFFFH is recommended.

Processing details

- This instruction opens the connection specified by (s1). The setting value used for open processing is selected by (s2)+0.
- The execution status and the completion status of this instruction can be checked with the completion device (d) specified by the setup data and the completion status indication device (d)+1.

• Completion device (d)

This device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing.

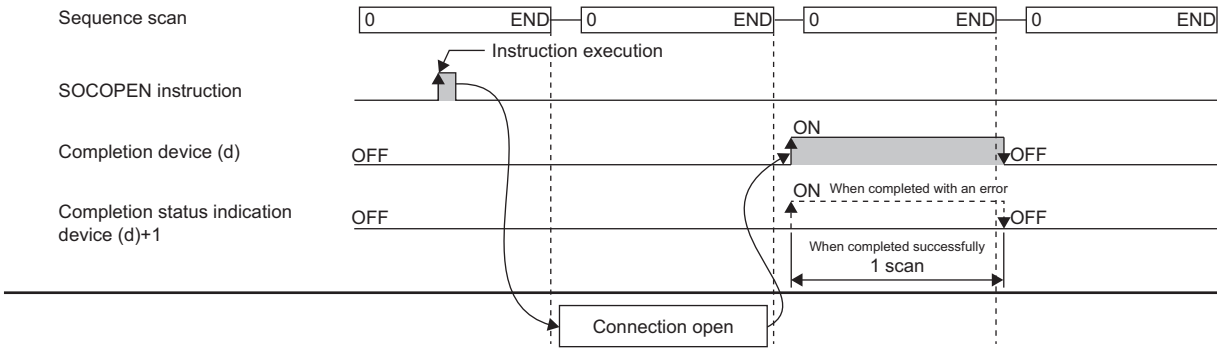
• Completion status indication device (d)+1

This device turns on or off depending on the completion status of this instruction.

Completed successfully: The device remains off.

Completed with an error: The device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of this instruction.



- A connection which has not been set by a parameter (a connection whose protocol field is left blank) can be opened and used. To do so, set (s2)+0 to 8000H and specify the details of open in (s2)+1 to (s2+9 of control data.

Point

This instruction supports the encrypted predefined protocols (TLS and DTLS). For more details, check the user's manual for the controller used.

Precautions

- The following table describes precautions to be taken when establishing a connection on an Ethernet port. For details on how to check the connection status, check the user's manual for the controller used.

Item	Description
A connection is established with TCP/IP or TLS protocol, and in FullPassive mode	After executing this instruction, if a connection request is received from an external device with an IP address that is not set in the external device connection configuration, a normal response is returned, but then the connection is disconnected. From the perspective of the external device, the connection is established once and then disconnected. For how to check whether the connection establishment is successful, check the completion status of the following instructions. If the connection establishment is successful, 0000H is stored. If the connection establishment fails, C1B1H is stored. • SOCSND instruction • SOCRCV instruction • SOCCSET instruction • ECPRTCL instruction
A connection is established again after establishing a connection with TCP/IP or TLS protocol, and in FullPassive mode/UnPassive mode	While a connection has been established by a connection request from the external device after executing this instruction, if another connection request is received from the external device, the request will not be accepted and the connection will be disconnected. (For example, the external device may be reset before executing the SOCCLOSE instruction after establishing the connection, and the external device may request to open the connection again.) The controller requires several hundred milliseconds to be able to accept the connection request again, so the external device should wait 1 to 2 seconds after checking the disconnection before sending another connection request. Also, to determine whether the reconnection request on the external device was successful, check whether the connection status on the external device is open.

Item	Description
A connection is established with TCP/IP protocol and TLS protocol	Even if this instruction is completed successfully, the connection is not established and data communication is not guaranteed. To determine whether the connection was disconnected after successful completion by external device, check whether the connection status on the external device is open. The following shows the operation of each case. ■Active mode (controller): TCP/IP, Passive mode: TLS When this instruction is executed from the controller in Active mode of TCP/IP protocol to the external device that is open in Passive mode of TLS protocol, the instruction will be completed successfully and the external device will be disconnected. ■Active mode: TLS, Passive mode (controller): TCP/IP When this instruction is executed from the controller in Passive mode of TCP/IP protocol and a connection request is received from the external device in Active mode of TLS protocol, the instruction will be completed successfully and the external device will be disconnected.

- Do not execute the Ethernet function add-on and the Ethernet instruction at the same time. If they are executed at the same time, malfunction may occur.

Operation error

Error code (SD0)	Description
3285H	The connection number specified by (s1) is outside the range.

Upon completion with an error, the completion status indication device (d)+1 is turned on and an error code is stored in the completion status (s2)+1.
For the error code stored in the completion status (s2)+1, refer to the following.
User's manual for the controller used

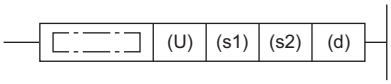
Point

When the SOCOPEN instruction is executed for TCP/TLS/DTLS Passive connection, the instruction waits for the connection and is being executed until the connection is completed by a connection request from the external device.

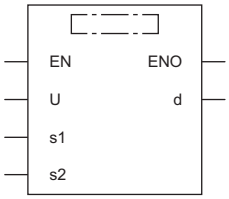
Closing a connection

GP.SOCCLOSE, SP.SOCCLOSE

This instruction closes the specified connection.

Ladder	ST
	ENO:=GP_SOCCLOSE(EN,U,s1,s2,d); ENO:=SP_SOCCLOSE(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.SOCCLOSE SP.SOCCLOSE	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.SOCCLOSE	Start I/O number of target (Specifies the first 3 digits of the start I/O number)	3E0H	16-bit unsigned binary	ANY16
	SP.SOCCLOSE	Dummy	—	String	ANYSTRING_SINGLE
(s1)		Connection number	1 to 64	16-bit unsigned binary	ANY16
(s2)	GP.SOCCLOSE	Start device where control data is stored	Refer to the control data.	Device	ANY16
	SP.SOCCLOSE			Word	ANY16_ARRAY (Number of elements: 2)
(d)		Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others (U)
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	GP.SOCCLOSE	—	—	○	—	—	—	—	○	○	—	—	○
	SP.SOCCLOSE	—	—	○	—	—	—	—	○	—	—	○	○
(s1)		—	—	○	—	—	—	—	○	○	—	—	—
(s2)		—	—	○	—	—	—	—	○	—	—	—	—
(d)		○	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System

Processing details

- This instruction performs close processing for the connection specified by (s1).
- The execution status and the completion status of this instruction can be checked with the completion device (d) specified by the setup data and the completion status indication device (d)+1.

• Completion device (d)

This device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing.

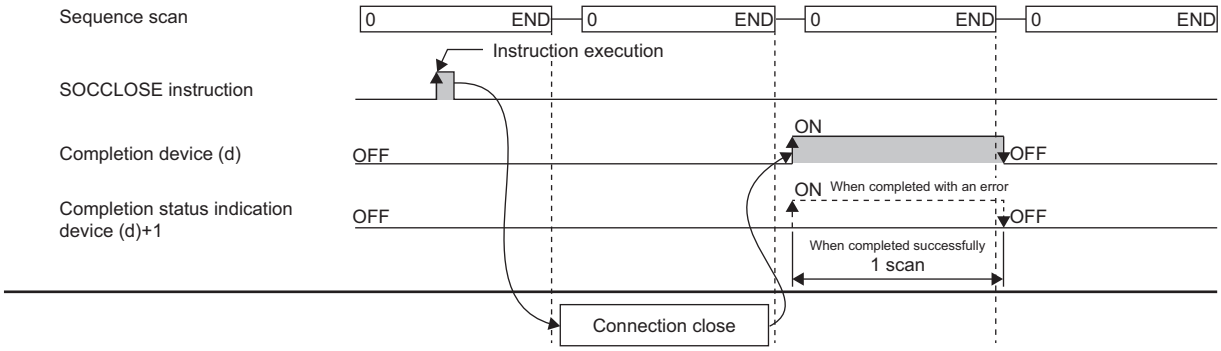
• Completion status indication device (d)+1

This device turns on or off depending on the completion status of this instruction.

Completed successfully: The device remains off.

Completed with an error: The device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of this instruction.



Operation error

Error code (SD0)	Description
3285H	The connection number specified by (s1) is outside the range.

Upon completion with an error, the completion status indication device (d)+1 is turned on and an error code is stored in the completion status (s2)+1.

For the error code stored in the completion status (s2)+1, refer to the following.

📖User's manual for the controller used

26.2 Socket Communications Instructions

Point

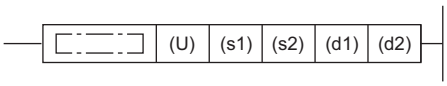
For connections where any of the following instructions is being executed, a request is ignored even if the following multiple instructions are attempted to be executed.
At this time, as long as the execution conditions of the instruction executed later are met, the request is executed after the first instruction being executed completes. SM699 (Dedicated instruction skip flag) turns on while the request is ignored.

- SOCOPEN instruction
- SOCCLOSE instruction
- SOCSND instruction
- SOCRCV instruction
- ECPRTCL instruction

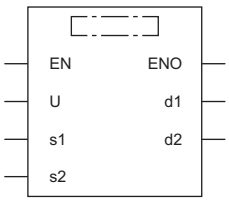
Reading receive data during the END processing

GP.SOCRCV, SP.SOCRCV

This instruction reads the receive data of the specified connection, during END processing after instruction execution, from the socket communications receive data area.

Ladder	ST
	ENO:=GP_SOCRCV(EN,U,s1,s2,d1,d2); ENO:=SP_SOCRCV(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.SOCRCV SP.SOCRCV	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.SOCRCV	Start I/O number of target (Specifies the first 3 digits of the start I/O number)	3E0H	16-bit unsigned binary	ANY16
	SP.SOCRCV	Dummy	—	String	ANYSTRING_SINGLE
(s1)		Connection number	1 to 64	16-bit unsigned binary	ANY16
(s2)	GP.SOCRCV	Start device where control data is stored	Refer to the control data.	Device	ANY16
	SP.SOCRCV			Word	ANY16_ARRAY (Number of elements: 2)
(d1)	GP.SOCRCV	Start device for storing the receive data	—	Device	ANY16*1
	SP.SOCRCV			Word	
(d2)		Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d2)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others (U)
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	GP.SOCRCV	—	—	○	—	—	—	—	○	○	—	—	○
	SP.SOCRCV	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	GP.SOCRCV	○	—	○	—	—	—	—	○	○	—	—	—
	SP.SOCRCV	—	—	○	—	—	—	—	○	○	—	—	—
(s2)		—	—	○	—	—	—	—	○	—	—	—	—
(d1)		—	—	○	—	—	—	—	○	—	—	—	—
(d2)		○	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Receive data length	The length of the data read from the socket communications receive data area is stored. (Number of bytes)	0 to 10238	System
+1 to +□	Receive data	The data read from the socket communications receive data area is stored sequentially in ascending order of addresses.*1	—	System

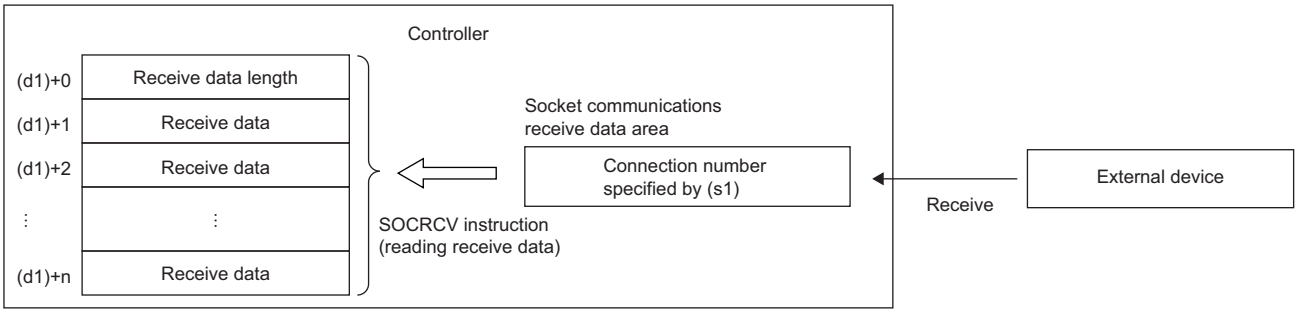
*1 The received data is stored in units of bytes sequentially from lower bytes. When an odd number of bytes of data is received, the last receive data is stored in the lower byte of the last data storage area.

Point

- When this instruction is executed, receive data is read from the socket communications receive data area during END processing. For this reason, executing this instruction prolongs the scan time.
- When an odd number of bytes of data is received, invalid data is stored in the upper byte of the device where the last receive data is stored.

Processing details

- This instruction reads the receive data of the connection specified by (s1) from the socket communications receive data area (where the data received from an external device in each connection is stored) by the END processing after the instruction execution.



- The execution status and the completion status of this instruction can be checked with the completion device (d2) specified by the setup data and the completion status indication device (d2)+1.

• Completion device (d2)

This device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing.

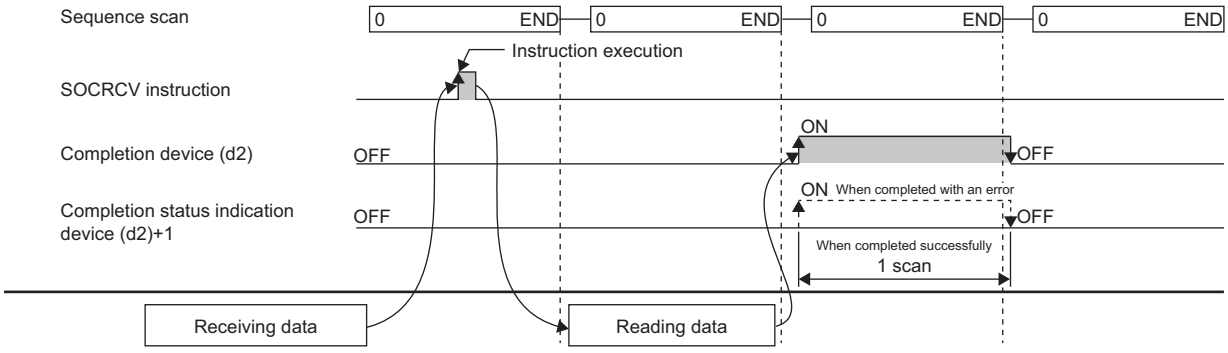
• Completion status indication device (d2)+1

This device turns on or off depending on the completion status of this instruction.

Completed successfully: The device remains off.

Completed with an error: The device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of this instruction.



Point

This instruction supports the encrypted predefined protocols (TLS and DTLS). For more details, check the user's manual for the controller used.

Precautions

When reading receive data from the same connection, do not use this command together with the SOCRCVS instruction.

Operation error

Error code (SD0)	Description
3285H	The connection number specified by (s1) is outside the range.

Upon completion with an error, the completion status indication device (d2)+1 is turned on and an error code is stored in the completion status (s2)+1.

For the error code stored in the completion status (s2)+1, refer to the following.

User's manual for the controller used

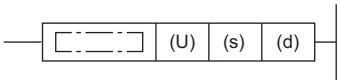
Point

- To avoid receiving too much amount of data, the SOCRMODE instruction can be used to set the size of receive data to limit the amount of receive data.
- By connecting the completion device of this instruction to the execution instruction through a normally closed contact, data can be read continuously even when it is received continuously.

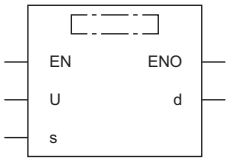
Reading receive data when the instruction is executed

G.SOCRCVS, S.SOCRCVS

This instruction reads the receive data of the specified connection from the socket communications receive data area.

Ladder	ST
	ENO:=G_SOCRCVS(EN,U,s,d); ENO:=S_SOCRCVS(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.SOCRCVS S.SOCRCVS	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	G.SOCRCVS	Start I/O number of target (Specifies the first 3 digits of the start I/O number)	3E0H	16-bit unsigned binary	ANY16
	S.SOCRCVS	Dummy	—	String	ANYSTRING_SINGLE
(s)		Connection number	1 to 64	16-bit unsigned binary	ANY16
(d)	G.SOCRCVS	Start device for storing the receive data	—	Device	ANY16*1
	S.SOCRCVS			Word	
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others (U)
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	G.SOCRCVS	—	—	○	—	—	—	—	○	○	—	—	○
	S.SOCRCVS	—	—	○	—	—	—	—	○	—	—	○	○
(s)	G.SOCRCVS	○	—	○	—	—	—	—	○	○	—	—	—
	S.SOCRCVS	—	—	○	—	—	—	—	○	○	—	—	—
(d)		—	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (d)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Receive data length	The length of the data read from the socket communications receive data area is stored. (Number of bytes)	0 to 10238	System
+1 to +□	Receive data	The data read from the socket communications receive data area is stored sequentially in ascending order of addresses.*1	—	System

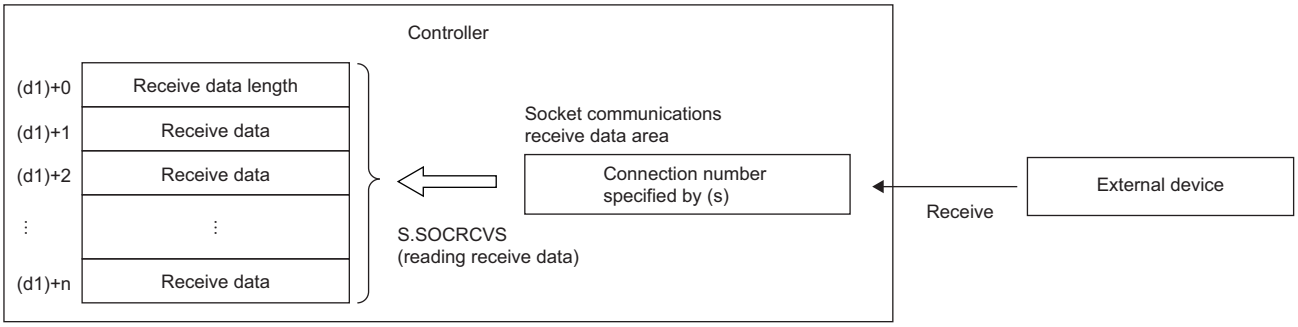
*1 The received data is stored in units of bytes sequentially from lower bytes. When an odd number of bytes of data is received, the last receive data is stored in the lower byte of the last data storage area.

Point

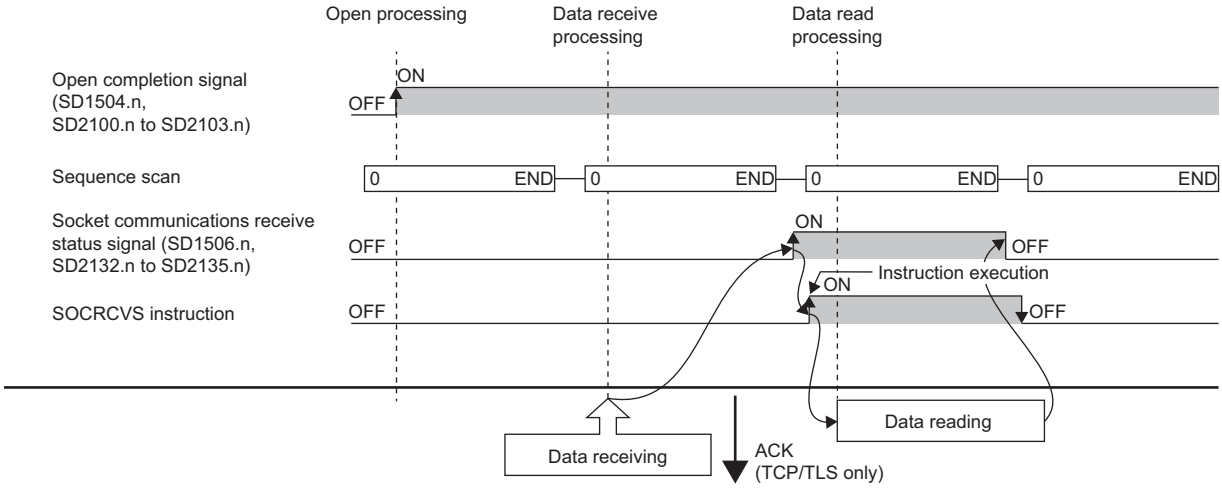
- The default receive data size is 2046 bytes. To receive 2047 bytes of data or more, change the receive data size using the SOCRMODE instruction.
- When an odd number of bytes of data is received, invalid data is stored in the upper byte of the device where the last receive data is stored.

Processing details

- The S.SOCRCVS instruction reads the receive data of the connection specified by (s) from the socket communications receive data area (where the data received from an external device in each connection is stored).



- The following figure shows the timing of receive processing using this instruction.



Point

This instruction supports the encrypted predefined protocols (TLS and DTLS). For more details, check the user's manual for the controller used.

Precautions

When reading receive data from the same connection, do not use this command together with the SOCRCV instruction.

Operation error

Error code (SD0)	Description
3285H	The connection number specified by (s) is outside the range.

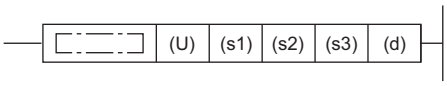
Point

To avoid receiving too much amount of data, the SOCRMODE instruction can be used to set the size of receive data to limit the amount of receive data.

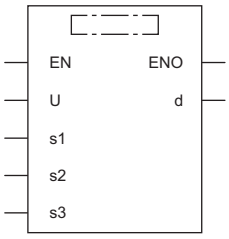
Sending data

GP.SOCSND, SP.SOCSND

This instruction sends the data to the external device of the specified connection.

Ladder	ST
	ENO:=GP_SOCSND(EN,U,s1,s2,s3,d); ENO:=SP_SOCSND(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.SOCSND SP.SOCSND	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.SOCSND	Start I/O number of target (Specifies the first 3 digits of the start I/O number)	3E0H	16-bit unsigned binary	ANY16
	SP.SOCSND	Dummy	—	String	ANYSTRING_SINGLE
(s1)		Connection number	1 to 64	16-bit unsigned binary	ANY16
(s2)	GP.SOCSND	Start device where control data is stored	Refer to the control data.	Device	ANY16
	SP.SOCSND			Word	ANY16_ARRAY (Number of elements: 2)
(s3)	GP.SOCSND	Start device for storing the send data	—	Device	ANY16*1
	SP.SOCSND			Word	
(d)		Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others (U)
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	GP.SOCSND	—	—	○	—	—	—	—	○	○	—	—	○
	SP.SOCSND	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	GP.SOCSND	○	—	○	—	—	—	—	○	○	—	—	—
	SP.SOCSND	—	—	○	—	—	—	—	○	○	—	—	—
(s2)		—	—	○	—	—	—	—	○	—	—	—	—
(s3)		—	—	○	—	—	—	—	○	—	—	—	—
(d)		○	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System

Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Send data length	Specify the send data length. (Number of bytes)	1 to 10238	User
+1 to +□	Send data	Specify the send data.*1	—	User

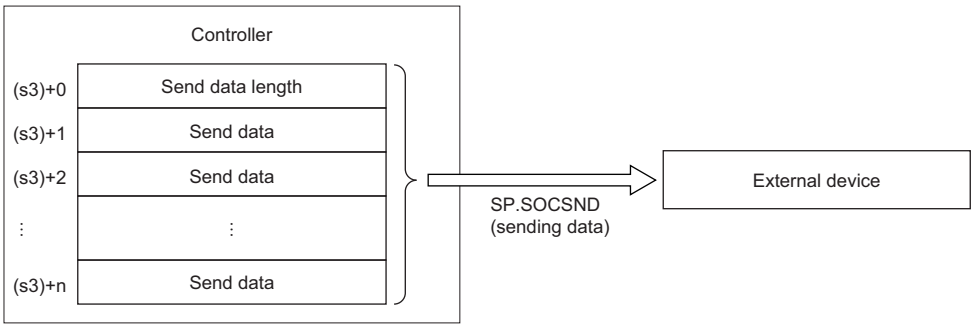
*1 The send data is sent in units of bytes sequentially from lower bytes. When an odd number of bytes of data is sent, store the last send data in the lower byte of the last data storage area.

Point

When TCP is used, the send data length should be equal to or less than the maximum window size (receive buffer) of the external device.
When TLS is used, the length of send data before encrypted should be equal to or less than the maximum window size (receive buffer) of the external device.
Data which exceeds the maximum window size of the external device cannot be sent.

Processing details

- Sends the data in the device specified by (s3) to the external device of the connection specified by (s1).



- The execution status and the completion status of this instruction can be checked with the completion device (d) specified by the setup data and the completion status indication device (d)+1.

• Completion device (d)

This device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing.

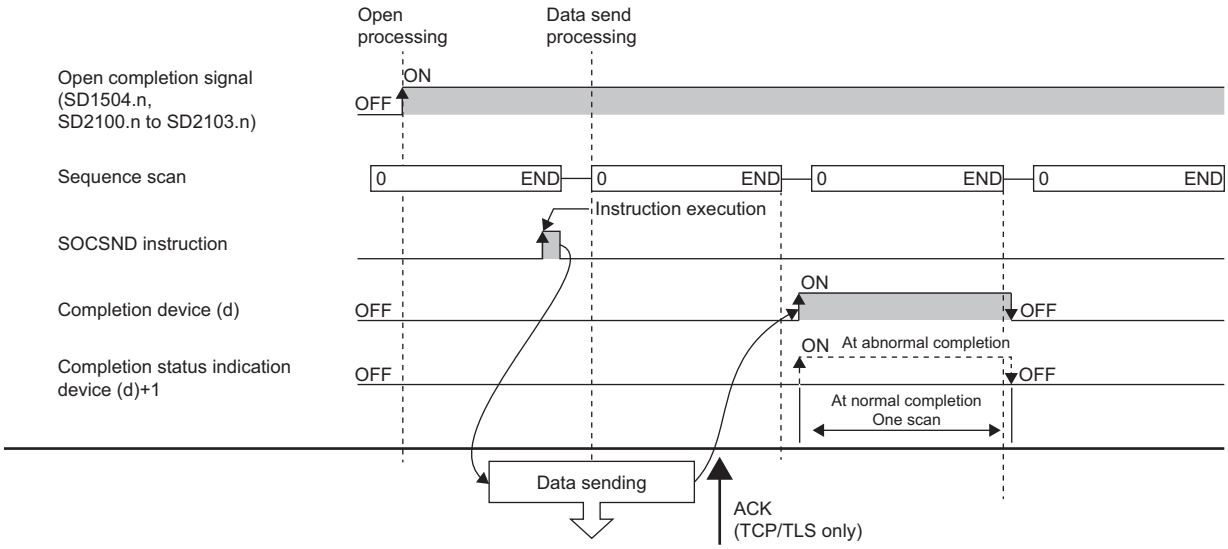
• Completion status indication device (d)+1

This device turns on or off depending on the completion status of this instruction.

Completed successfully: The device remains off.

Completed with an error: The device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing.

- The following figure shows the timing of send processing using this instruction.



Even after the completion device turns on, data may be sent continuously. Check the completion of the send processing on the receiving side.

Precautions

Create a means of arrival check on both the own station and the external device to check that the send data arrives at the external device and data transmission is completed successfully after the instruction is executed.
If a connection is established with the external device by using TCP/IP or TLS protocol, and then this instruction is executed to send data to the external device, the execution of this instruction is completed without checking the arrival of that data at the external device, and the completion status ((S2)+1) becomes completed successfully.
For example, create a program that notifies the external device when data was successfully sent, to notify the external device of arrival.

Operation error

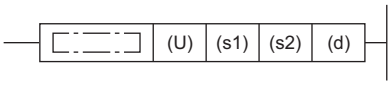
Error code (SD0)	Description
3285H	The connection number specified by (s1) is outside the range.
3293H	The length of send data specified by (s3) is outside the range.

Upon completion with an error, the completion status indication device (d)+1 is turned on and an error code is stored in the completion status (s2)+1.
For the error code stored in the completion status (s2)+1, refer to the following.
User's manual for the controller used

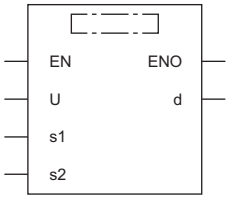
Reading connection information

GP.SOCCINF, SP.SOCCINF

This instruction reads the connection information of the connection specified by (s1) and stores it in the device specified by (d) and later.

Ladder	ST
	ENO:=GP_SOCCINF(EN,U,s1,s2,d); ENO:=SP_SOCCINF(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.SOCCINF SP.SOCCINF	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.SOCCINF	Start I/O number of target (Specifies the first 3 digits of the start I/O number)	3E0H	16-bit unsigned binary	ANY16
	SP.SOCCINF	Dummy	—	String	ANYSTRING_SINGLE
(s1)		Connection number	1 to 64	16-bit unsigned binary	ANY16
(s2)	GP.SOCCINF	Start device where control data is stored	Refer to the control data.	Device	ANY16
	SP.SOCCINF			Word	ANY16_ARRAY (Number of elements: 2)
(d)		Head device for storing connection information	—	Word	ANY16_ARRAY (Number of elements: 5)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

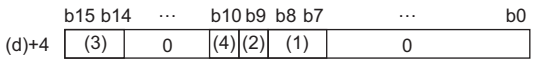
Operand		Bit		Word			Double word		Indirect specification	Constant			Others (U)
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	GP.SOCCINF	—	—	○	—	—	—	—	○	○	—	—	○
	SP.SOCCINF	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	GP.SOCCINF	○	—	○	—	—	—	—	○	○	—	—	—
	SP.SOCCINF	—	—	○	—	—	—	—	○	○	—	—	—
(s2)		—	—	○	—	—	—	—	○	—	—	—	—
(d)		—	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (s2)

Device	Item	Description	Setting range	Set by
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System

Operand: (d)

Device	Item	Description	Setting range	Set by
+0 +1	IP address of external device	Store the IP address of an external device.	00000001H to FFFFFFFFH 00000000H: No communication destination (FFFFFFFFH: simultaneous broadcast)	System
+2	Destination port number	Store the destination port number of an external device.	0001H to FFFFH (FFFFH: Simultaneous broadcast)	System
+3	Own station port number.	Store the own station port number.	0001H to 1387H 1392H to FFFEH	System
+4	Connection use application	Store the usage of a connection.  (1) Communication method (protocol) (b8 and 7) • 00: TCP • 01: TLS • 10: UDP • 11: DTLS (2) Socket communications function procedure (b9) • 1: No procedure (fixed) (3) Open method (b15, b14) • 00: Active open or UDP • 10: Unpassive open • 11: Fullpassive open (4) Predefined protocol setting (b10) • 0: Do not use the predefined protocol support function. (Use the socket communications function.) • 1: Use the predefined protocol support function.	—	System

Processing details

Reads the connection information of the connection specified by (s1).

Point

This instruction supports the encrypted predefined protocols (TLS and DTLS). For more details, check the user's manual for the controller used.

In TCP/TLS/DTLS Passive mode, if this instruction is executed when connection is open and a connection with the external device is not established, data other than information of the external device is stored as follows. In addition, when connection is not open, 0H is stored.

Device	Item	Data to be stored	
		Connection is open but connection with the external device is not established	Connection is not open
d+0 d+1	IP address of external device	00000000H	00000000H
d+2	Destination port number	0000H	0000H
d+3	Own station port number	Data when connection is open	0000H
d+4	Connection use application	Data when connection is open	0000H

Operation error

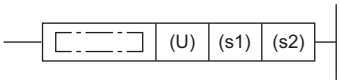
Error code (SD0)	Description
3285H	The connection number specified by (s1) is outside the range.

When completed with an error, an error code is stored in the completion status (s2)+1.
For the error code stored in the completion status (s2)+1, refer to the following.
User's manual for the controller used

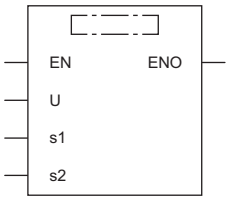
Changing the communication target (UDP/IP)

GP.SOCCSET, SP.SOCCSET

This instruction changes the communication target IP address and port number of the specified connection.

Ladder	ST
	ENO:=GP_SOCCSET(EN,U,s1,s2); ENO:=SP_SOCCSET(EN,U,s1,s2);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.SOCCSET SP.SOCCSET	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.SOCCSET	Start I/O number of target (Specifies the first 3 digits of the start I/O number)	3E0H	16-bit unsigned binary	ANY16
	SP.SOCCSET	Dummy	—	String	ANYSTRING_SINGLE
(s1)		Connection number	1 to 64	16-bit unsigned binary	ANY16
(s2)	GP.SOCCSET	Start device where control data is stored	Refer to the control data.	Device	ANY16
	SP.SOCCSET			Word	ANY16_ARRAY (Number of elements: 5)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others (U)
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	GP.SOCCSET	—	—	○	—	—	—	—	○	○	—	—	○
	SP.SOCCSET	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	GP.SOCCSET	○	—	○	—	—	—	—	○	○	—	—	—
	SP.SOCCSET	—	—	○	—	—	—	—	○	○	—	—	—
(s2)		—	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+2 +3	IP address of external device	Store the IP address of an external device.	00000001H to FFFFFFFFH (FFFFFFFFH: simultaneous broadcast)	User
+4	Destination port number	Store the destination port number of an external device.	0001H to FFFFH (FFFFH: simultaneous broadcast)	User

Processing details

In UDP communications, this instruction changes the communication target IP address and port number of the connection specified by (s1).

Point

- Using this instruction allows the user to change the communication destination without closing the connection.
- After the instruction is executed, the IP address and port number of the external device may be changed in the next scan or later. Use the SOCCINF instruction to check that the value is the one specified by this instruction.

Precautions

Do not use this instruction to change the communication destination during execution of the SOCSND instruction.

Operation error

Error code (SD0)	Description
3285H	The connection number specified by (s1) is outside the range.

When completed with an error, an error code is stored in the completion status (s2)+1.

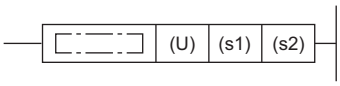
For the error code stored in the completion status (s2)+1, refer to the following.

User's manual for the controller used

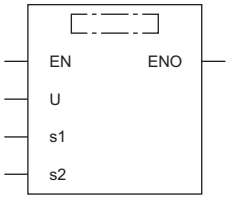
Changing the receive mode

GP.SOCRMODE, SP.SOCRMODE

This instruction changes the TCP/TLS receive mode and receive data size for the specified connection (invalid for UDP/DTLS communications connections).

Ladder	ST
	ENO:=GP_SOCRMODE(EN,U,s1,s2); ENO:=SP_SOCRMODE(EN,U,s1,s2);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.SOCRMODE SP.SOCRMODE	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.SOCRMODE	Start I/O number of target (Specifies the first 3 digits of the start I/O number)	3E0H	16-bit unsigned binary	ANY16
	SP.SOCRMODE	Dummy	—	String	ANYSTRING_SINGLE
(s1)		Connection number	1 to 64	16-bit unsigned binary	ANY16
(s2)	GP.SOCRMODE	Start device where control data is stored	Refer to the control data.	Device	ANY16
	SP.SOCRMODE			Word	ANY16_ARRAY (Number of elements: 4)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others (U)
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	GP.SOCRMODE	—	—	○	—	—	—	—	○	○	—	—	○
	SP.SOCRMODE	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	GP.SOCRMODE	○	—	○	—	—	—	—	○	○	—	—	—
	SP.SOCRMODE	—	—	○	—	—	—	—	○	○	—	—	—
(s2)		—	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+2	TCP/TLS receive mode*1	Store the TCP/TLS receive mode. • 0: TCP/TLS standard receive mode • 1: TCP/TLS fixed-length receive mode	0, 1	User
+3	Receive data size	Store the size of socket communications receive data (Number of bytes)	1 to 10238	User

*1 This item is invalid for UDP/DTLS communications connections.

Processing details

- This instruction changes the TCP/TLS receive mode and receive data size for the connection (other than a UDP/DTLS communications connection) specified by (s1).
- For TCP/TLS connections, the function enables the mode specified by (s2)+2.

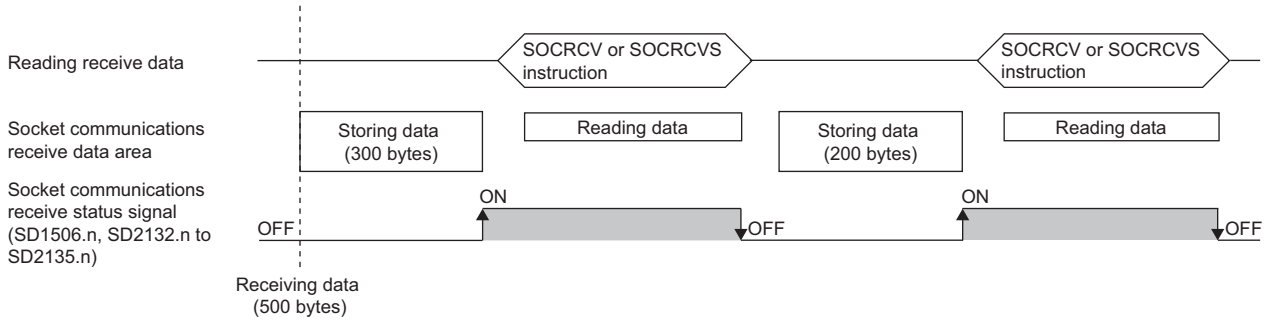
Point

- This instruction supports the encrypted predefined protocol (TLS). For more details, check the user's manual for the controller used.
- Start receiving data from the connected device at least 100ms after executing this instruction.

■TCP/TLS standard receive mode

Upon receipt of data, the instruction stores the data in the socket communications receive data area and turns on SD1506, and SD2132 to SD2135 (socket communications receive status signal).
If the received data exceeds the specified receive data size, the excess data becomes the next receive data.
If data is received later before data is read from the socket communications receive data area using the SOCRCV instruction or SOCRCVS instruction, it is stored in the receive data area in the OS.
If the receive data area in the OS contains data when data is read from the socket communications receive data area using the SOCRCV instruction or SOCRCVS instruction, the instruction stores the data in the socket communications receive data area and turns on SD1506 and SD2132 to SD2135 (socket communications receive status signal).

Ex.
When 500 bytes of data is received while the receive data size is set to 300 bytes



■TCP/TLS fixed-length receive mode

Upon receipt of data, the instruction stores the data in the socket communications receive data area. If the specified receive data size is not reached, SD1506 and SD2132 to SD2135 (socket communications receive status signal) does not turn on. Data reception is repeated until the received data reaches the receive data size. When it reaches the receive data size, SD1506 and SD2132 to SD2135 (socket communications receive status signal) turn on.

If the received data exceeds the specified receive data size, the excess data becomes the next receive data.

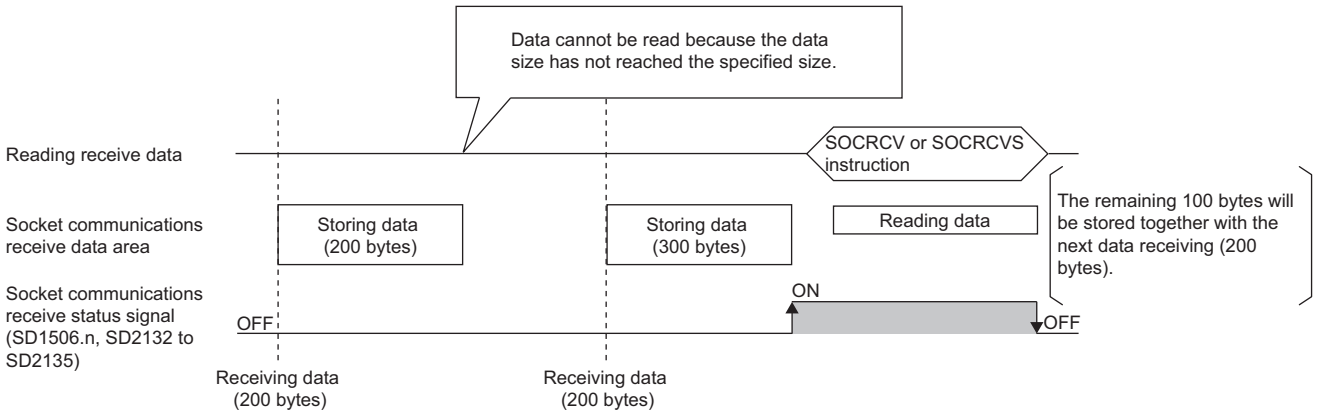
If data is received later before data is read from the socket communications receive data area using the SOCRCV instruction or SOCRCVS instruction, it is stored in the receive data area in the OS.

If the receive data area in the OS contains data when data is read from the socket communications receive data area using the SOCRCV instruction or SOCRCVS instruction, the instruction stores the data in the socket communications receive data area, but does not turn on SD1506 and SD2132 to SD2135 (socket communications receive status signal) if the data has not reached the specified receive data size.

Then data reception is repeated until the received data reaches the receive data size. When it reaches the receive data size, SD1506 and SD2132 to SD2135 (socket communications receive status signal) turn on.

Ex.

When 200 bytes of data is received continuously while the receive data size is set to 300 bytes



Point

• Effective use of devices

The receive data storage device used by the SOCRCV instruction or SOCRCVS instruction needs a 5119-word area (10238 bytes). Decreasing the receive data size enables effective use of the device.

• Preventing receive data from being divided

Depending on the line type, data to be received from the external device may be divided before arrival. In this case, specifying the receive data size in TCP/TLS fixed-length receive mode can prevent receive data from being divided.

• Preventing receive data from being connected

Due to a delay in receive processing of the sequence program, data which has been divided and sent may be connected before receiving depending on the external device.

Specifying the receive data size in TCP/TLS fixed-length receive mode enables data to be correctly divided and received.

- The size of the receive data to be read once by the SOCRCV instruction or SOCRCVS instruction is specified in (s2)+3. In the case of UDP/DTLS, if the received data exceeds the specified receive data size, the excess data becomes the next receive data.

Operation error

Error code (SD0)	Description
3285H	The connection number specified by (s1) is outside the range.

When completed with an error, an error code is stored in the completion status (s2)+1.

For the error code stored in the completion status (s2)+1, refer to the following.

User's manual for the controller used

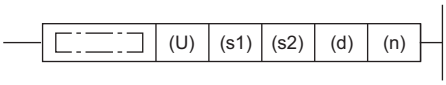
Point

When the receive status signal does not turn on in TCP/TLS fixed-length receive mode, the data received as of the current time can be read with the SOCRDATA instruction to check whether the data sent from the external device is missing.

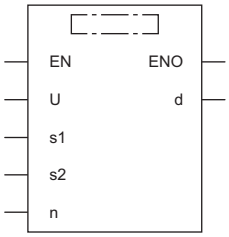
Reading socket communications receive data

G(P).SOCRDATA, S(P).SOCRDATA

These instructions read data by the number of words specified by (n) from the socket communications receive data area of the connection specified by (s1), and store them in the device specified by (d) and later.

Ladder	ST
	ENO:=G_SOCRDATA(EN,U,s1,s2,n,d); ENO:=GP_SOCRDATA(EN,U,s1,s2,n,d); ENO:=S_SOCRDATA(EN,U,s1,s2,n,d); ENO:=SP_SOCRDATA(EN,U,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
G.SOCRDATA S.SOCRDATA	
GP.SOCRDATA SP.SOCRDATA	

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(U)	G(P).SOCRDATA	Start I/O number of target (Specifies the first 3 digits of the start I/O number)	3E0H	16-bit unsigned binary	ANY16
	S(P).SOCRDATA	Dummy	—	String	ANYSTRING_SINGLE
(s1)		Connection number	1 to 64	16-bit unsigned binary	ANY16
(s2)	G(P).SOCRDATA	Start device where control data is stored	Refer to the control data.	Device	ANY16
	S(P).SOCRDATA			Word	ANY16_ARRAY (Number of elements: 2)
(d)		Head device for storing the data that has been read	—	Word	ANY16*1
(n)		Number of read data	1 to 5120	16-bit unsigned binary	ANY16
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others (U)
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	G(P).SOCRDATA	—	—	○	—	—	—	—	○	○	—	—	○
	S(P).SOCRDATA	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	G(P).SOCRDATA	○	—	○	—	—	—	—	○	○	—	—	—
	S(P).SOCRDATA	—	—	○	—	—	—	—	○	○	—	—	—
(s2)		—	—	○	—	—	—	—	○	—	—	—	—
(d)		—	—	○	—	—	—	—	○	—	—	—	—
(n)		○	—	○	—	—	—	—	○	○	—	—	—

■Control data

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System

Processing details

These instructions read data by the number of words specified by (n) from the socket communications receive data area of the connection specified by (s1), and store them in the device specified by (d) and later. If the read data (n) is 0, no processing is performed.

Point

- This instruction supports the encrypted predefined protocols (TLS and DTLS). For more details, check the user's manual for the controller used.
- The receive data length can be read by setting the number of read data to 1 word. As a result, the device for storing receive data when the SOCRCV instruction or SOCRCVS instruction is executed can be changed.
- After issuing this instruction to check the data to be received this time and issuing the SOCRMODE instruction to specify the size of the data to be received next time, the SOCRCV instruction or SOCRCVS instruction can be used to read the data of this time. As a result, based on the data received this time, the size of data to be received next can be specified.

Operation error

Error code (SD0)	Description
3285H 3293H	The connection number specified by (s1) is outside the range.
	When the number of read data specified by (n) exceeds the maximum range.

When completed with an error, an error code is stored in the completion status (s2)+1.
For the error code stored in the completion status (s2)+1, refer to the following.
User's manual for the controller used

26.3 Predefined Protocol Support Function Instruction

Point

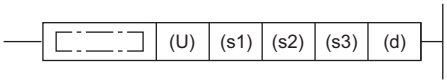
For connections where any of the following instructions is being executed, a request is ignored even if the following multiple instructions are attempted to be executed.
At this time, as long as the execution conditions of the instruction executed later are met, the request is executed after the first instruction being executed completes. SM699 (Dedicated instruction skip flag) turns on while the request is ignored.

- SOCOPEN instruction
- SOCCLOSE instruction
- SOCSND instruction
- SOCRCV instruction
- ECPRTCL instruction

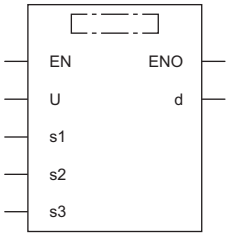
Executing the registered protocols

GP.ECPRTCL, SP.ECPRTCL

This instruction executes the protocol registered using the engineering tool.

Ladder	ST
	ENO:=GP_ECPRTCL(EN,U,s1,s2,s3,d); ENO:=SP_ECPRTCL(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.ECPRTCL SP.ECPRTCL	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.ECPRTCL	Start I/O number of target (Specifies the first 3 digits of the start I/O number)	3E0H	16-bit unsigned binary	ANY16
	SP.ECPRTCL	Dummy	—	String	ANYSTRING_SINGLE
(s1)		Connection number	1 to 64	16-bit unsigned binary	ANY16
(s2)		Number of protocols to be executed continuously	1 to 8	16-bit unsigned binary	ANY16
(s3)	GP.ECPRTCL	Start device where control data is stored	Refer to the control data.	Device	ANY16_ARRAY (Number of elements: 18)
	SP.ECPRTCL			Word	ANY16_ARRAY (Number of elements: 2)
(d)		Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others (U)
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	GP.ECPRTCL	—	—	○	—	—	—	—	○	○	—	—	○
	SP.ECPRTCL	—	—	○	—	—	—	—	○	—	—	○	○
(s1)		○	—	○*1	—	—	—	—	○*1	○	—	—	—
(s2)		○	—	○*1	—	—	—	—	○*1	○	—	—	—
(s3)	GP.ECPRTCL	—	—	○*1	—	—	—	—	○*1	—	—	—	—
	SP.ECPRTCL	○*2	—	○*1	—	—	—	—	○*1	—	—	—	—
(d)		○*1	—	○	—	—	—	—	○*1	—	—	—	—

*1 A local device and a file register which is set for each program cannot be used.
*2 Available only when 16-bit (4 digits) digit specification is used and the start bit device is a multiple of 16 (10H).

■Control data

Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Resulting number of executed protocols	The number of protocols executed by this instruction is stored. Any protocol where an error occurred is also included in the execution number. If the setting of setting data or control data contains an error, "0" is stored.	0, 1 to 8	System
+1	Completion status	The completion status is stored upon completion of the instruction. When two or more protocols are executed, the execution result of the protocol executed last is stored. • 0: Completed successfully • Other than 0: Completed with an error (error code)	—	System
+2	Execution protocol number 1	Specify the number of the protocol to be executed first.	1 to 128	User
+3	Execution protocol number 2	Specify the number of the protocol to be executed second.	0, 1 to 128	User
+4	Execution protocol number 3	Specify the number of the protocol to be executed third.	0, 1 to 128	User
+5	Execution protocol number 4	Specify the number of the protocol to be executed fourth.	0, 1 to 128	User
+6	Execution protocol number 5	Specify the number of the protocol to be executed fifth.	0, 1 to 128	User
+7	Execution protocol number 6	Specify the number of the protocol to be executed sixth.	0, 1 to 128	User
+8	Execution protocol number 7	Specify the number of the protocol to be executed seventh.	0, 1 to 128	User
+9	Execution protocol number 8	Specify the number of the protocol to be executed eighth.	0, 1 to 128	User
+10	Collation match Receive packet number 1	If receiving is included in the communication type of the protocol that has been executed first, the receive packet number successful in collation match is stored. If the communication type is "Send only", "0" is stored. If an error occurs during execution of the first protocol, "0" is stored.	0, 1 to 16	System
+11	Collation match Receive packet number 2	If receiving is included in the communication type of the protocol that has been executed second, the receive packet number successful in collation match is stored. If the communication type is "Send only", "0" is stored. If an error occurs during execution of the second protocol, "0" is stored. If the number of protocols executed is less than 2, "0" is stored.	0, 1 to 16	System
+12	Collation match Receive packet number 3	If receiving is included in the communication type of the protocol that has been executed third, the receive packet number successful in collation match is stored. If the communication type is "Send only", "0" is stored. If an error occurs during execution of the third protocol, "0" is stored. If the number of protocols executed is less than 3, "0" is stored.	0, 1 to 16	System
+13	Collation match Receive packet number 4	If receiving is included in the communication type of the protocol that has been executed fourth, the receive packet number successful in collation match is stored. If the communication type is "Send only", "0" is stored. If an error occurs during execution of the fourth protocol, "0" is stored. If the number of protocols executed is less than 4, "0" is stored.	0, 1 to 16	System
+14	Collation match Receive packet number 5	If receiving is included in the communication type of the protocol that has been executed fifth, the receive packet number successful in collation match is stored. If the communication type is "Send only", "0" is stored. If an error occurs during execution of the fifth protocol, "0" is stored. If the number of protocols executed is less than 5, "0" is stored.	0, 1 to 16	System
+15	Collation match Receive packet number 6	If receiving is included in the communication type of the protocol that has been executed sixth, the receive packet number successful in collation match is stored. If the communication type is "Send only", "0" is stored. If an error occurs during execution of the sixth protocol, "0" is stored. If the number of protocols executed is less than 6, "0" is stored.	0, 1 to 16	System
+16	Collation match Receive packet number 7	If receiving is included in the communication type of the protocol that has been executed seventh, the receive packet number successful in collation match is stored. If the communication type is "Send only", "0" is stored. If an error occurs during execution of the seventh protocol, "0" is stored. If the number of protocols executed is less than 7, "0" is stored.	0, 1 to 16	System

Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+17	Collation match Receive packet number 8	If receiving is included in the communication type of the protocol that has been executed eighth, the receive packet number successful in collation match is stored. If the communication type is "Send only", "0" is stored. If an error occurs during execution of the eighth protocol, "0" is stored. If the number of protocols executed is less than 8, "0" is stored.	0, 1 to 16	System

Processing details

- This instruction executes the protocol registered using the engineering tool. Using the connection specified by (s1), the instruction executes the protocol in accordance with the control data stored in the device specified by (s3) and later.
- The instruction continuously executes as many protocols as specified by (s2) (a maximum of 8 protocols) at one time.
- The number of executed protocols is stored in the device specified by (s3)+0.
- The protocol execution status can be checked with the predefined protocol support function execution status check area (Un\G350 to Un\G669). (User's manual for the controller used)
- The execution status and the completion status of this instruction can be checked with the completion device (d) specified by the setup data and the completion status indication device (d)+1.

• Completion device (d)

This device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing.

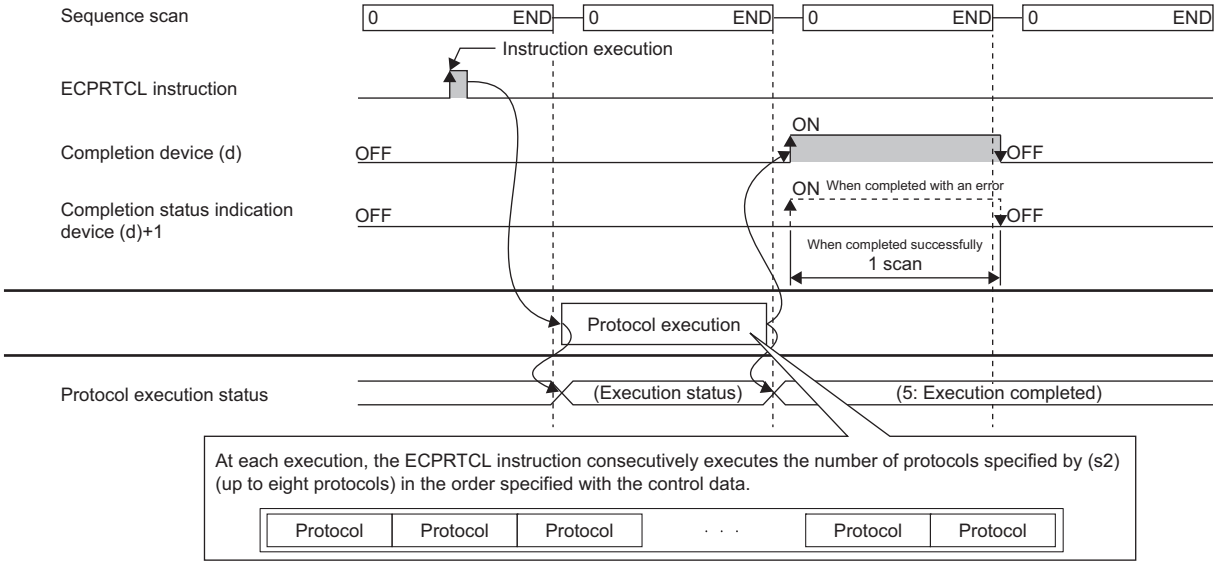
• Completion status indication device (d)+1

This device turns on or off depending on the completion status of this instruction.

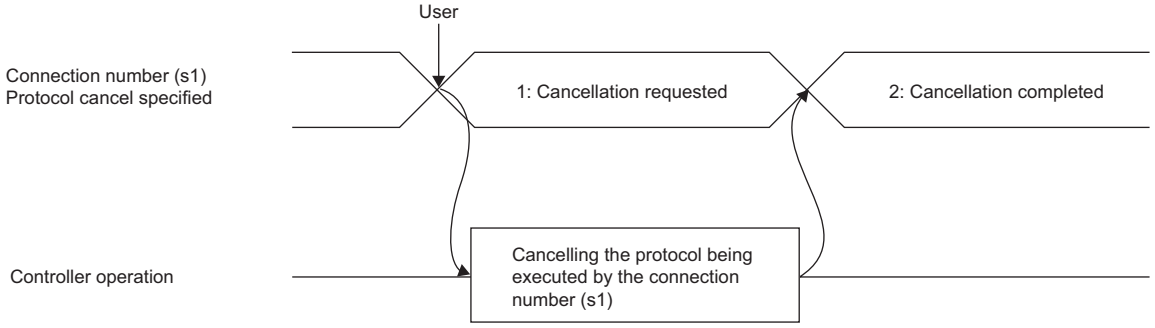
Completed successfully: The device remains off.

Completed with an error: The device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing. In addition, an error code is stored in the device specified by (s3)+1.

- The following figure shows the execution timing of this instruction.



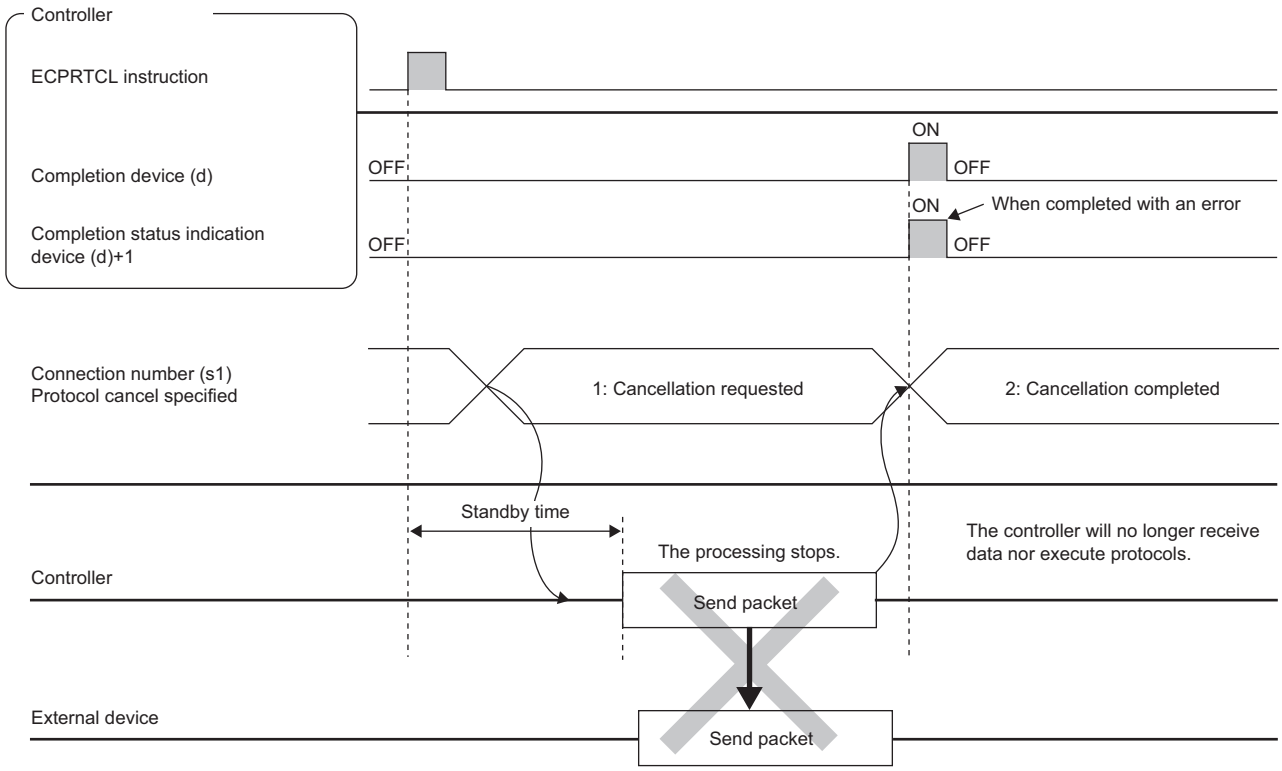
- Protocol execution can be canceled by setting a protocol cancel request. The protocol cancel request is specified in the predefined protocol support function execution status check area (Un\G350 to Un\G669). (User's manual for the controller used)



- The following figure shows the protocol cancel operations from time to time.

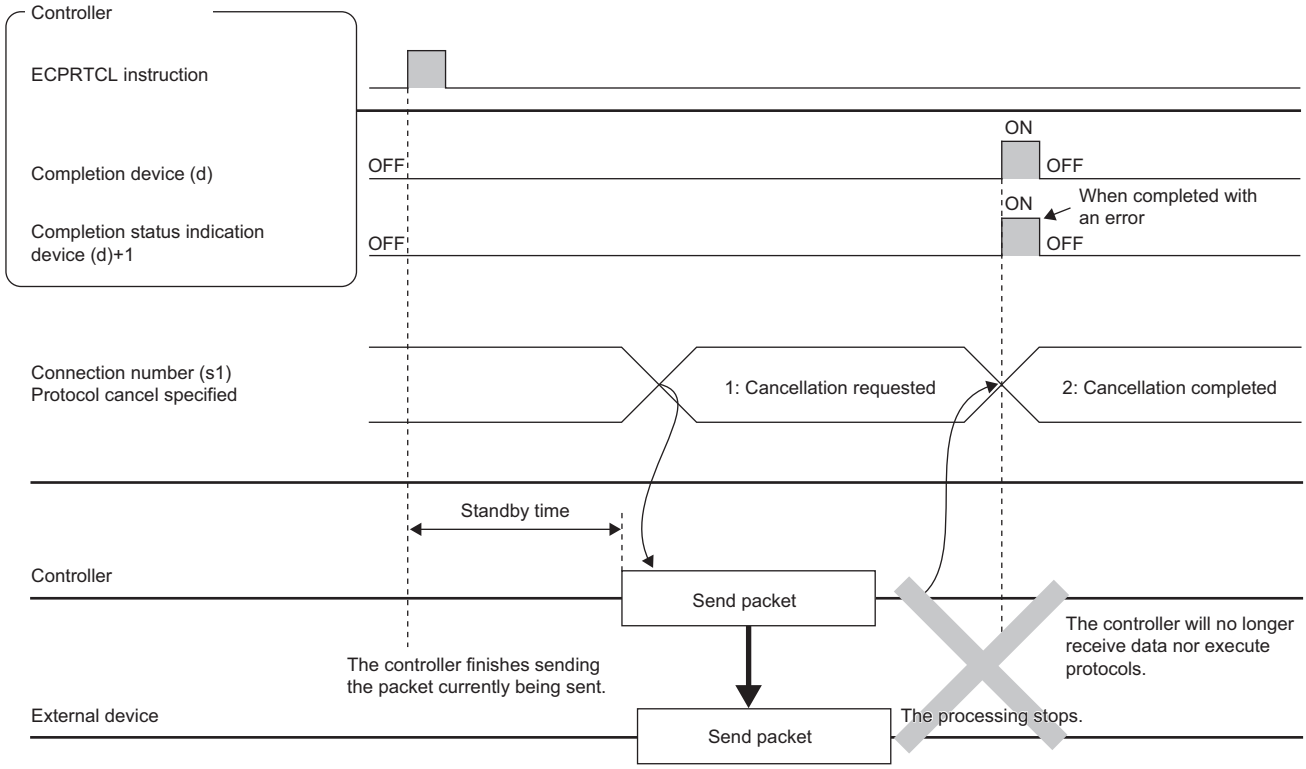
• If a cancel request is issued before transmission

The following figure shows the operation when the protocol execution status is "1: Waiting for transmission".



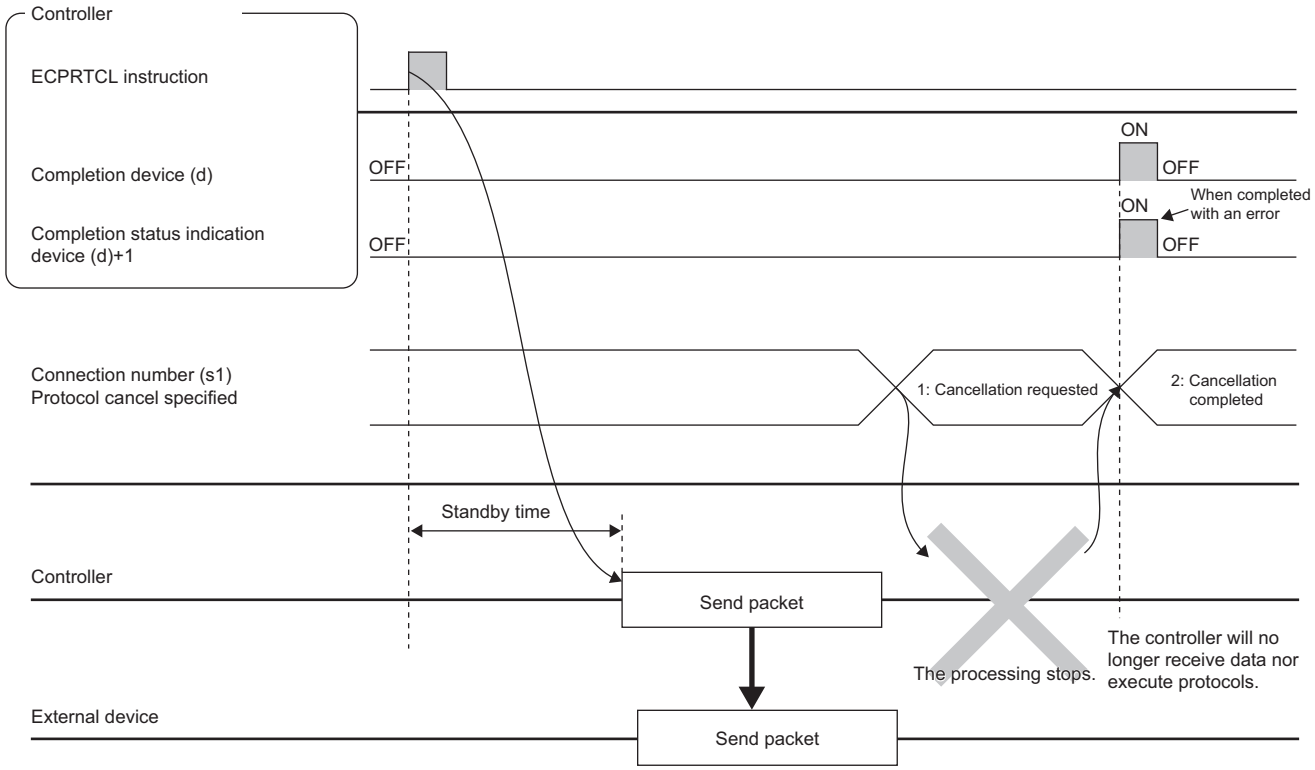
• If a cancel request is issued before completion of transmission

The following figure shows the operation when transmission has not been completed while the protocol execution status is "2: Sending".



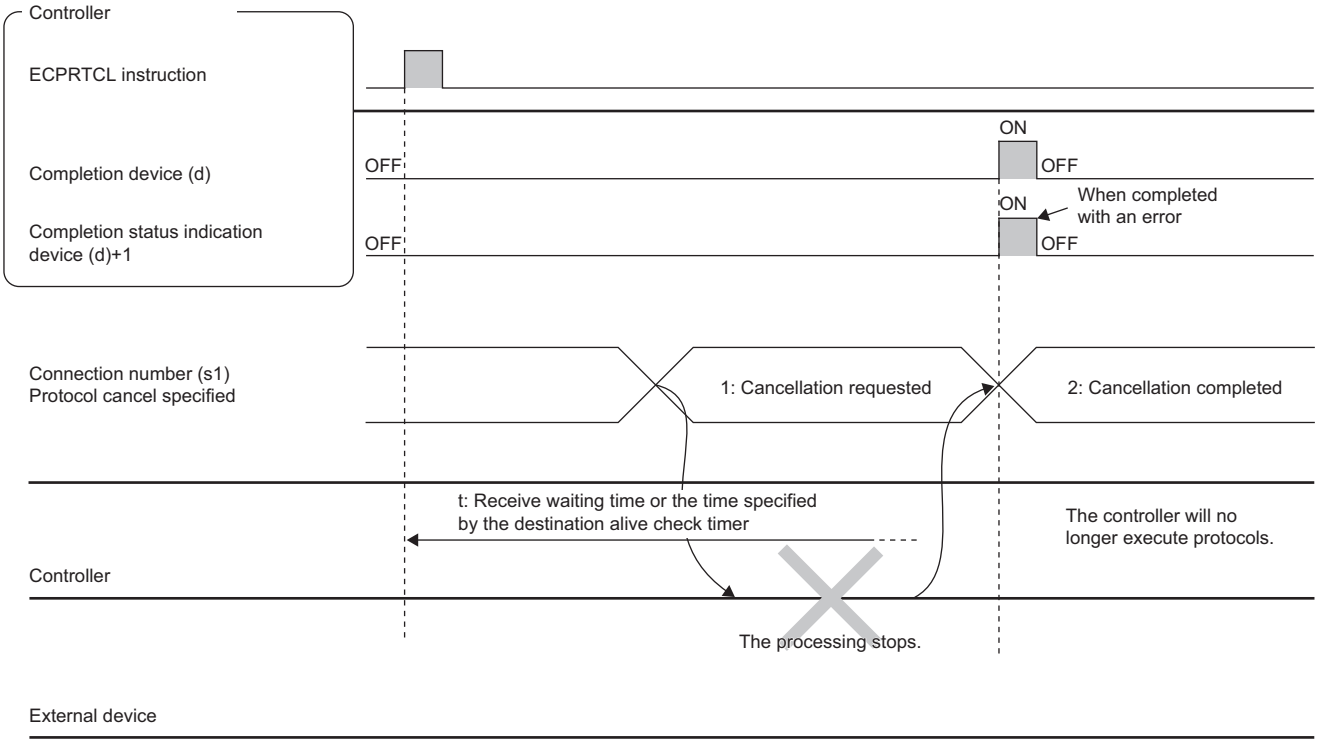
• If a cancel request is issued upon completion of transmission

The following figure shows the operation when transmission has been completed while the protocol execution status is "2: Sending".



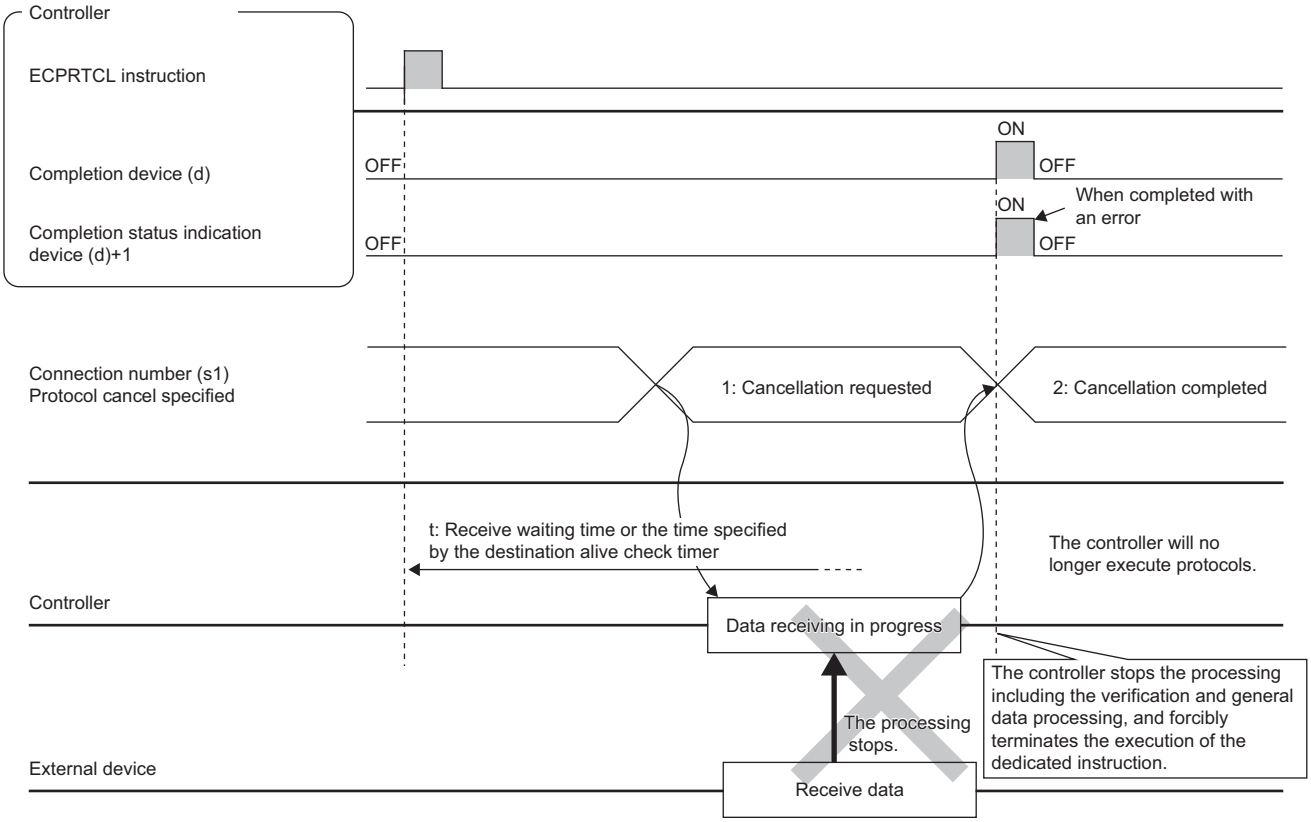
• If a cancel request is issued while waiting for reception

The following figure shows the operation when the protocol execution status is "3: Waiting for data reception".



• If a cancel request is issued during receiving

The following figure shows the operation when the protocol execution status is "4: Receiving".



Operation error

Error code (SD0)	Description
3285H	(s1) is outside the range.

Upon completion with an error, the completion status indication device (d)+1 is turned on and an error code is stored in the completion status (s3)+1.

For the error code stored in the completion status (s3)+1, refer to the following.

User's manual for the controller used

Precautions

- If an error occurs in the mth protocol while multiple protocols are being executed, the instruction does not execute the "m+1"th protocol and after and is completed with an error.
- When a protocol including no-conversion variables is executed, the total data length of the variables used in one packet may exceed 1920 bytes. In this case, the instruction may obtain CPU device values over several scans. Therefore, do not change the CPU device values specified in non-conversion variables from the start of the instruction to the end of execution.
- Protocol cancellation
  - This instruction is completed with an error and stores the protocol cancel request error (C404H) in the device (completion status) specified by (s3)+1.
  - If a cancel request is issued while no protocol is being executed, the controller completes the cancel request without performing any processing.
  - While no communication protocol is used, any cancel request is ignored if issued.
  - When multiple protocols are executed continuously, a cancel request may be issued during execution of the nth protocol. In this case, the controller forcibly terminates the nth protocol and does not execute the subsequent protocols. Protocol number n being executed is stored in the device specified by ((s3)+0), the receive packet number successful in comparison match is stored in the device specified by 1 to (n-1), and the protocol cancel request error (C404H) is stored in the device specified by ((s3)+1).
  - The controller periodically checks for a cancel request. For this reason, it may take time until cancel processing is performed after a cancel request is issued.
- This instruction itself does not open/close a connection and therefore the SOCOPEN instruction/SOCCLOSE instructions need to be used to open/close the connection.

☞ Page 1110 GP.SOCOPEN, SP.SOCOPEN

☞ Page 1115 GP.SOCCLOSE, SP.SOCCLOSE

- If same instructions are executed for the same connection, the subsequent instruction is ignored and is not executed until the preceding instruction is completed.
- If the receive waiting time is set to "0: Infinite wait", this instruction is not completed until the data specified in the protocol setting is received.
- When "Send Only" is selected for the communication type, and data is sent to the external device by using TCP/IP protocol, the execution of this instruction is completed without checking the arrival of that data at the external device, and the completion status ((S3)+1) becomes completed successfully. Select "Send & Receive" for the communication type to check for the arrival of the sent data at the external device, so that a response from the external device is awaited before the instruction is completed.

26.4 SLMP Frame Send Instruction

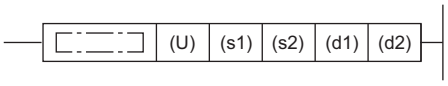
Sending an SLMP frame

G(P).SLMPSND, J(P).SLMPSND, SP.SLMPSND

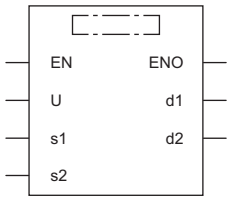
This instruction sends SLMP messages to the SLMP-compatible device.

Point

- The G(P).SLMPSND instruction can be used for both ports for Ethernet and for CC-Link IE TSN.
- The J(P).SLMPSND instruction can be used only for the ports for CC-Link IE TSN.
- The SP.SLMPSND instruction can be used only for the port for Ethernet.

Ladder	ST
	ENO:=G_SLMPSND(EN, U, s1, s2, d1, d2) ENO:=GP_SLMPSND(EN, U, s1, s2, d1, d2) ENO:=J_SLMPSND(EN, U, s1, s2, d1, d2) ENO:=JP_SLMPSND(EN, U, s1, s2, d1, d2) ENO:=SP_SLMPSND(EN, U, s1, s2, d1, d2)

FBD/LD



■Execution condition

Instruction	Execution condition
G.SLMPSND J.SLMPSND	
GP.SLMPSND JP.SLMPSND SP.SLMPSND	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	G(P).SLMPSND	Start I/O number of target (Specifies the first 3 digits of the start I/O number)	Ethernet port: 3E0H CC-Link IE TSN port: 00H to FEH	16-bit unsigned binary	ANY16
	J(P).SLMPSND	Own station network number	1 to 239	16-bit unsigned binary	ANY16
	SP.SLMPSND	Dummy	—	String	ANYSTRING_SINGLE
(s1)	G(P).SLMPSND J(P).SLMPSND	Start device where control data is stored	☞ Page 1149 Control data	Device	ANY16*1
	SP.SLMPSND			Word	
(s2)		Start device where a request frame is stored	☞ Page 1150 Request frame	Word	ANY16*1
(d1)		Start device for storing a response frame	☞ Page 1151 Response frame	Word	ANY16_ARRAY*2
(d2)		Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d2)+1 also turns on.	—	Bit	ANYBIT_ARRAY*2 (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

*2 When specifying setting data by using a label, define an array to secure enough operation area.

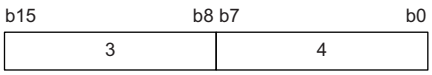
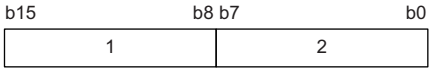
■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others	
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	U	J
(U)	G(P).SLMPSND	—	—	○	—	—	—	—	○	○	—	—	○	—
	J(P).SLMPSND	—	—	—	—	—	—	—	—	—	—	—	—	○
	SP.SLMPSND	—	—	○	—	—	—	—	○	—	—	○	○	—
(s1)		—	—	○	—	—	—	—	○	—	—	—	—	—
(s2)		—	—	○	—	—	—	—	○	—	—	—	—	—
(d1)		—	—	○	—	—	—	—	○	—	—	—	—	—
(d2)		○	—	○*1	—	—	—	—	○	—	—	—	—	—

*1 T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution/error completion type	b15 b7 b0 (3) | 0 | (2) | 0 | (1) (1) Execution type (b0) • 0: Without arrival check (The instruction is regarded as completed when a request message is sent from own station.)*1 • 1: With arrival check (The instruction is regarded as completed when a response message is received from the external device.) (2) Error completion type (b7) Specify whether to set data at completion with an error. • 0: Do not set data at completion with an error in (s1)+13 and later. (Clear (s1)+13 and later.) • 1: Set data at completion with an error in (s1)+13 and later. (3) Instruction type (b15) • 0: Operand (U) is specified as 00H to FEH • 1: Operand (U) is specified as 3E0H	—	User
+1	Completion status	The completion status is stored upon completion of the instruction. 0000H: Completed successfully Other than 0000H: Completed with an error (error code)	—	System
+2	Own station channel	Specify the channel to be used by own station. Since whether or not a serial number*4 is given to the request message depends on the channel, specify the channel as follows according to the application. When using in conjunction with the SLMPSNDC instruction, be careful not to overlap the channels because the channel where serial numbers are given is shared. • 1: No serial number is given • 2 to 9: Serial number is given • 10 to 17: Station number extension frame	• 1 to 17*2	User
+3	IP address of external device (third and fourth octets)	Specify the IP address (third and fourth octets) of external device. b15 b8 b7 b0 3 | 4 3, 4: Indicates the octets of the IP address.	00000001H to FFFFFFFEH (both (s1)+3 and (s1)+4 together) (1 to 4294967294)	User
+4	IP address of external device (first and second octets)	Specify the IP address (first and second octets) of external device. b15 b8 b7 b0 1 | 2 1, 2: Indicates the octets of the IP address.		User
+5	Destination port number	Specify the destination port number.	1 to 65534 (1 to FFFEH)	User
+6	Request destination network number	0000H (fixed)	0000H	User
+7	Request destination station number	00FFH (fixed)	00FFH	User
+8	Request destination module I/O number	Specify a module I/O number of the request destination. • 03FFH: Own station/control CPU • 03E0H: Multiple CPU No.1 • 03E1H: Multiple CPU No.2 • 03E2H: Multiple CPU No.3 • 03E3H: Multiple CPU No.4	03FFH, 03E0H to 03E3H	User
+9	Request destination multidrop station number	Specify a multidrop number of the external device.	0000H to FFFEH*3	User
+10	Number of resends*5	The device becomes effective when the execution type specified by (s1)+0 is "1: With arrival check". ■Before instruction execution Specify the number of times that the request message will be resent if the response message is not received within the monitoring time specified by (s1)+11. • 0 to 15 (times) ■At completion of instruction The number of times that the request message has been resent (result) is stored. • 0 to 15 (times)	0 to 15	User/ system

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+11	Arrival monitoring time	Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent the number of times specified in (s1)+10. • 0: 10s • 1 to 32767: 1 to 32767s	0 to 32767	User
+12	Clock setting flag	The validity status (valid or invalid) of the data in (s)+13 and later is stored. Note that the data in (s1)+13 and later is cleared when the instruction is completed successfully. • 0: Invalid • 1: Valid	—	System
+13	Clock data	Upper 8 bits: Month (01H to 12H) Lower 8 bits: Year (00H to 99H: Lower two digits of the year)	—	System
+14		Upper 8 bits: Hour (00H to 23H) Lower 8 bits: Day (01H to 31H)		
+15		Upper 8 bits: Second (00H to 59H) Lower 8 bits: Minute (00H to 59H)		
+16		Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.))		
+17	IP address of error detected device (third and fourth octets)	The IP address (third and fourth octets) of the station where an error was detected is stored.  3, 4: Indicates the octets of the IP address.	—	System
+18	IP address of error detected device (first and second octets)	The IP address (first and second octets) of the station where an error was detected is stored.  1, 2: Indicates the octets of the IP address.	—	System

*1 If (s1)+0 is set to "0: Without arrival check", receive data is not set. Set 0 in (s1)+0 in the following cases:
· When a command that does not return a response message is used
· When a response message is not referred to

*2 If operand (U) is specified as 3E0H, 1 to 9 can be specified.

*3 If operand (U) is specified as 3E0H, it is fixed to 0000H.

*4 Give the serial numbers when sending several request messages to the same SLMP-compatible device. Serial numbers to be given are automatically numbered by the system.
For the serial number, refer to the following.
SLMP Reference Manual

*5 When the request message send processing is executed, the request message may not be sent if the external device does not exist on the network, but the number of resends at the completion of the instruction is counted. At this time, the number of sent request messages does not match the number of resends at the completion of the instruction.

■Request frame

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Request data length	Specify the data length from the monitoring timer to the request data. (In units of bytes)	1 to 2000	User
+1	Monitoring timer	This timer sets the waiting time for the external device that received a request message to wait for the response after it issued a processing request to the access destination. (Unit: Increments of 250ms) • 0: Infinite wait • 1 to 65535: 1 to 65535 × 250ms	0 to 65535	User
+2 to +□	Request data	The request data of the SLMP message is stored.	—	User

■Response frame

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Response data length	The data length from the end code to the response data is stored. (In units of bytes)	2 to 2000	System
+1	End code	The result of command processing is stored. In normal end, 0 is stored. In abnormal end, an error code set by the external device is stored.	—	System
+2 to +□	Response data*1	Sets execution results for the request data. For details on the request data, refer to the following. 📖SLMP Reference Manual	—	System

*1 The response data is stored in units of bytes sequentially from lower bytes. When an odd number of bytes of response data is received, the last response data is stored in the lower byte of the last data storage area.

Processing details

- This instruction sends the request frame in the device specified by (s2) and later to the external device specified by the external device IP address in the control data. When a response message is received from the external device, it is stored in the device specified by (d1).
- This instruction uses UDP communications. Set the external device to use UDP.
- This instruction communicates in binary code. Match the setting of the external device also with the binary code.
- The execution status and the completion status of this instruction can be checked with the completion device (d2)+0 specified by the setup data and the completion status indication device (d2)+1.

• Completion device (d2)+0

This device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing.

• Completion status indication device (d2)+1

This device turns on or off depending on the completion status of this instruction.

Completed successfully: The device remains off.

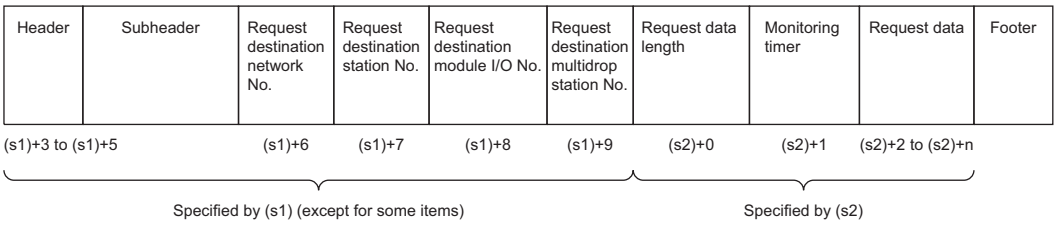
Completed with an error: The device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing.

- When an odd number of bytes of response data is received, invalid data is stored in the upper byte of the device where the last response data is stored.

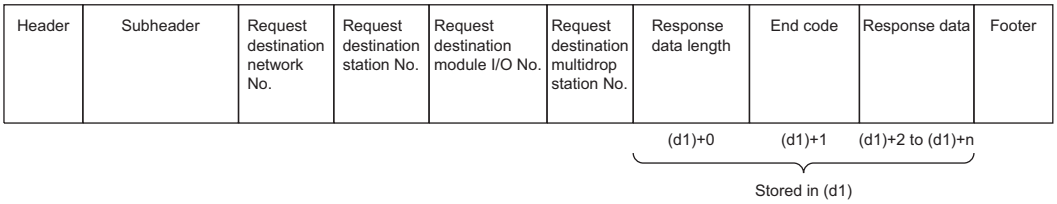
■Message format (3E/4E frame)

The following figures show the request data of 3E/4E frame and the response data in normal/abnormal end.

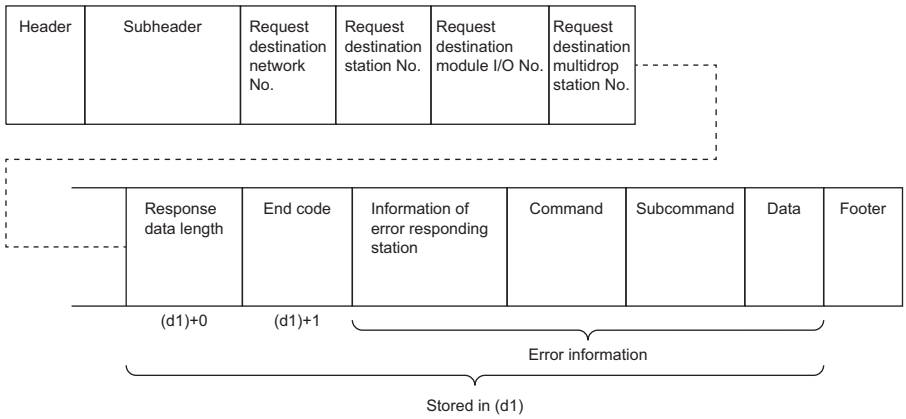
- Request message



- Response message (in normal end)



• Response message (in abnormal end)

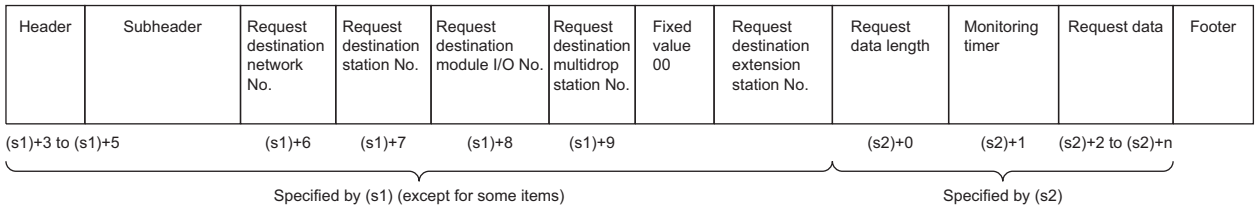


The network number, station number, request destination module I/O number, and multidrop station number are stored to the information of the error responding station.

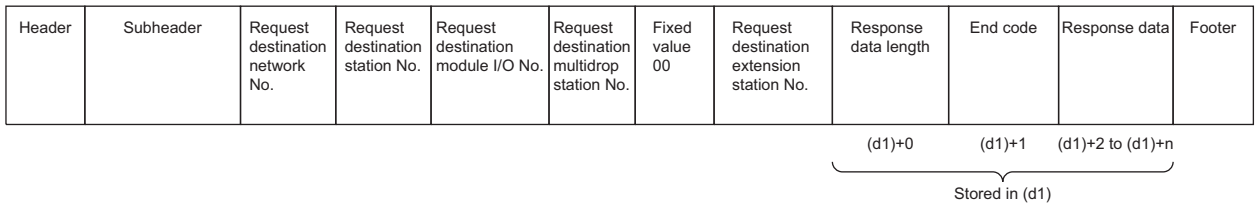
■Message format (station number extension frame)

The following figures show the request data of station number extension frame and the response data in normal/abnormal end.

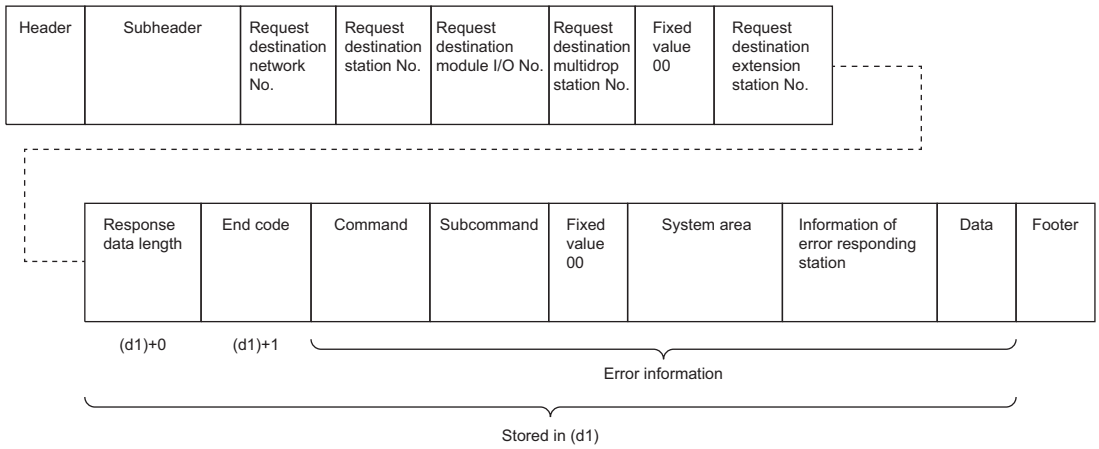
• Request message



• Response message (in normal end)



• Response message (in abnormal end)

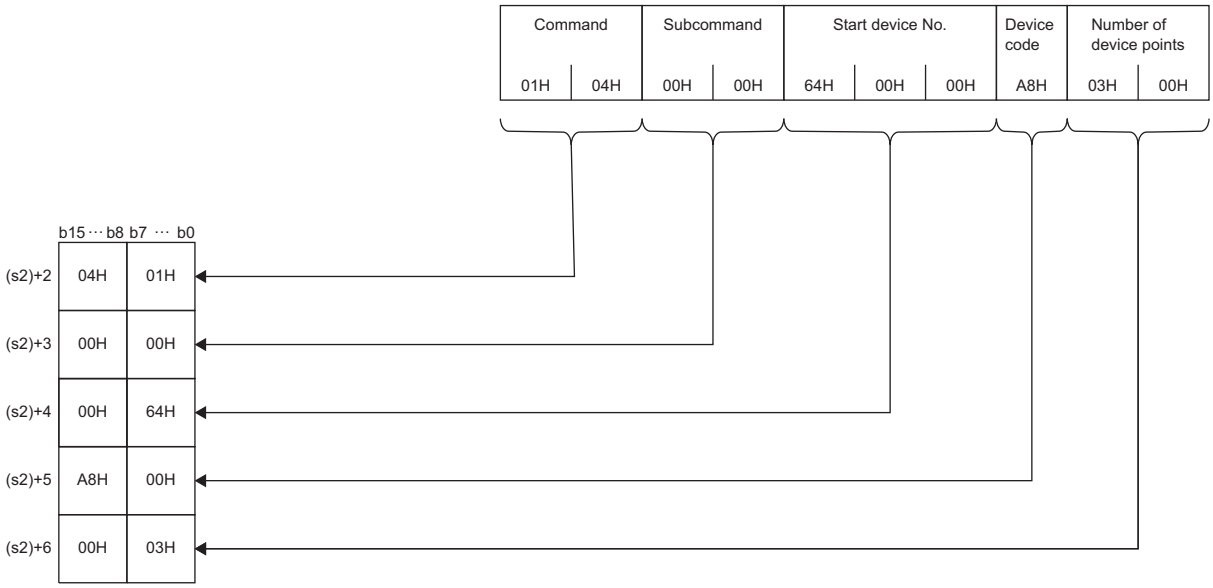


The network number, station number, request destination module I/O number, multidrop station number, 00 (fixed value), and extension station number are stored to the information of the error responding station.

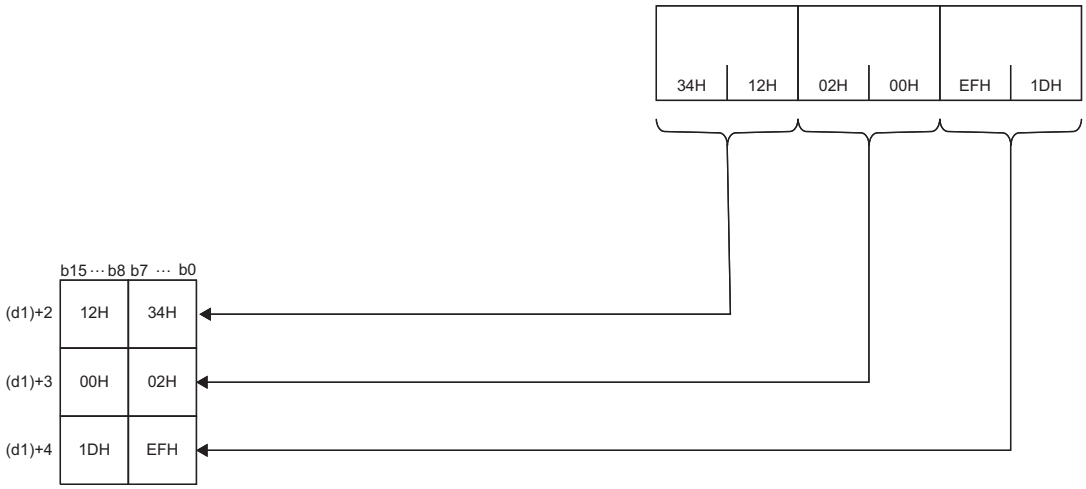
Ex.

When sending "Read (command: 0401H)" (reading in units of words) which reads the value in D100 to D102

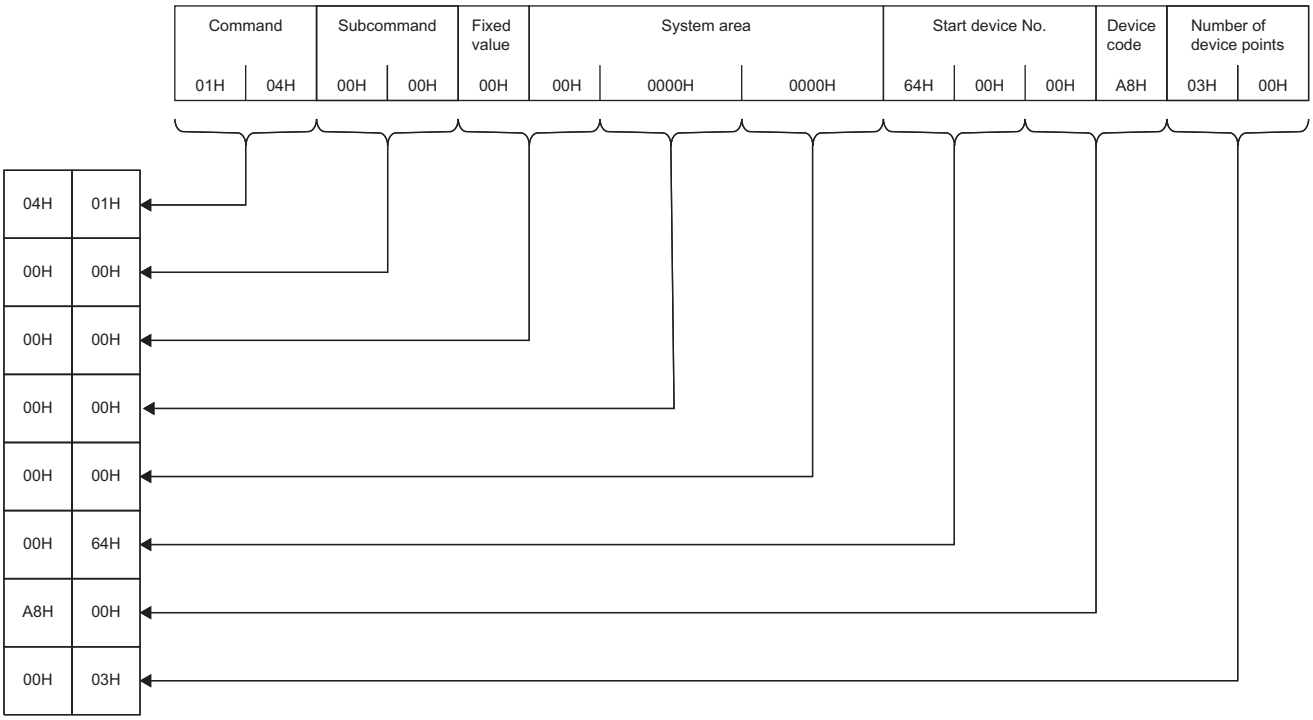
- The following example shows that request data is stored in (s2)+2 and later (3E/4E frame).



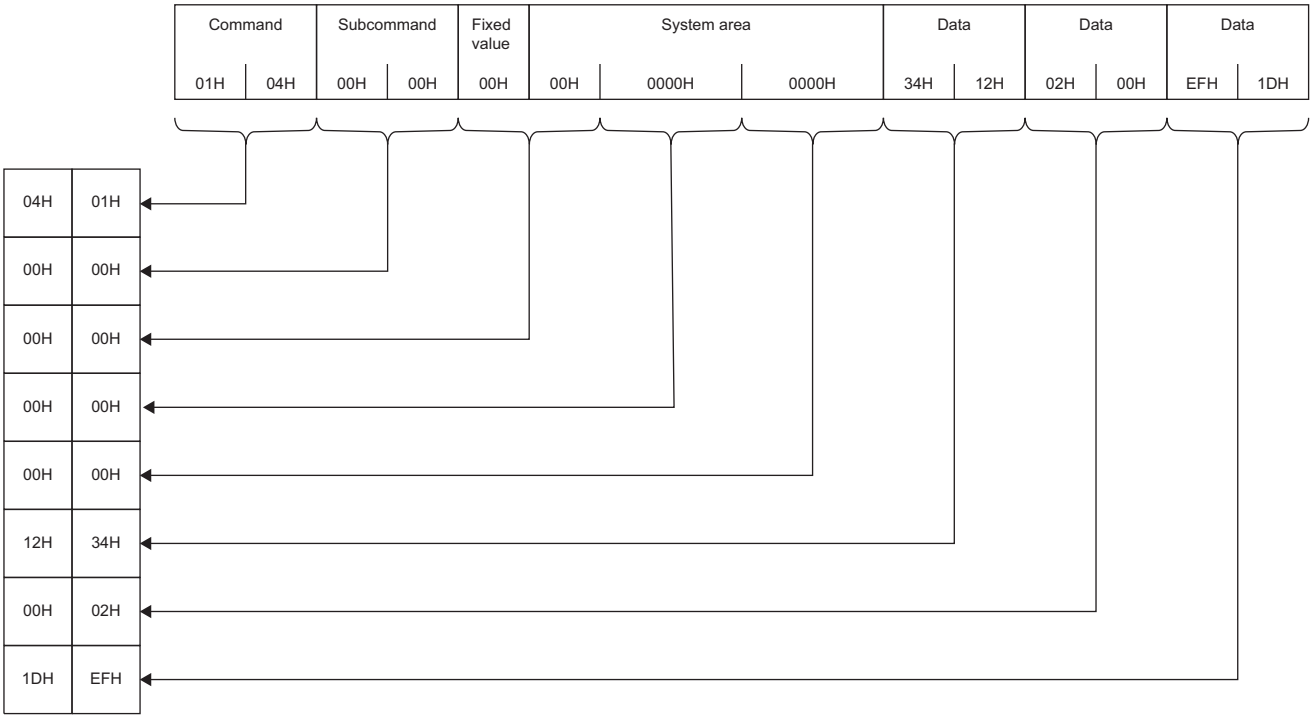
- The following example shows that response data is stored in (d1)+2 and later (3E/4E frame).



• The following example shows that request data is stored in (s2)+2 and later (station number extension frame).

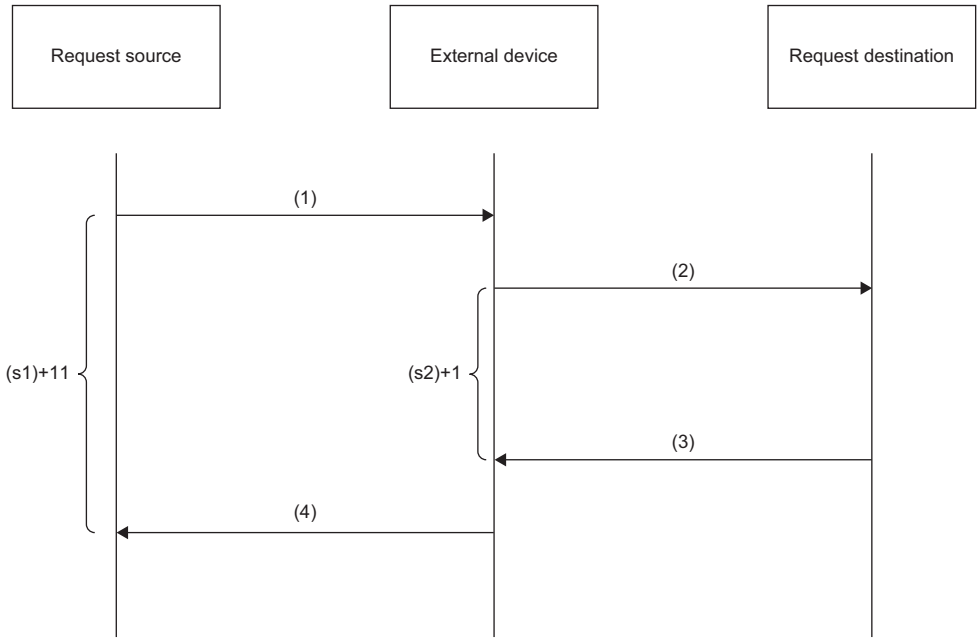


• The following example shows that response data is stored in (d1)+2 and later (station number extension frame).



Precautions

- When executing multiple SLMPSND instructions concurrently, be careful not to overlap the channels of the SLMPSND instructions. During the execution of the SLMPSND instruction, if another SLMPSND instruction specifying the same channel is executed, the later instruction is ignored. Upon completion of the first instruction, if the execution conditions for the later instruction are met, the later instruction will be requested consecutively. While the request for the later instruction is being ignored, SM699 (Dedicated instruction skip flag) turns on.
- Specify the arrival monitoring time ((s1)+11) of the control data and monitoring timer ((s2)+1) of the request frame so that the arrival monitoring time ≥ monitoring timer.



(1) Request message
(2) Processing request from external device to request destination
(3) Processing response from request destination to external device
(4) Response message

Point

This instruction is successfully completed even if the target device returns an abnormal response. When this instruction is completed successfully, the response is whether normal or abnormal can be identified by the end code of the response frame. When an abnormal response is returned, check the manual of the SLMP-compatible device being used and take corrective action.

Operation error

Error code (SD0)	Description
3285H	The value set to (s1)+2 as own station channel is out of the range.
	The value set to (s2)+0 as the request data length is 0 or exceeds 2000.

Upon completion with an error, the completion status indication device (d2)+1 is turned on and an error code is stored in the completion status (s1)+1. For the error code stored in the completion status (s1)+1, refer to the following.

User's manual for the controller used

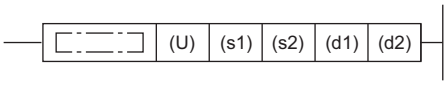
Sending an encrypted SLMP frame

G(P).SLMPSNDC

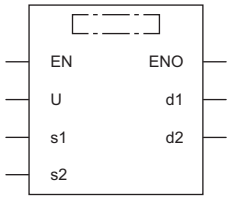
Sends encrypted SLMP messages to the SLMP-compatible device.

Point

This instruction can be used only for the port for Ethernet.

Ladder	ST
	ENO:=G_SLMPSNDC(EN, U, s1, s2, d1, d2) ENO:=GP_SLMPSNDC(EN, U, s1, s2, d1, d2)

FBD/LD



■Execution condition

Instruction	Execution condition
G.SLMPSNDC	
GP.SLMPSNDC	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Start I/O number of target (Specifies the first 3 digits of the start I/O number)	3E0H	16-bit unsigned binary	ANY16
(s1)	Start device where control data is stored	☞ Page 1149 Control data	Device name	ANY16*1
(s2)	Start device where a request frame is stored	☞ Page 1150 Request frame	Device name	ANY16*1
(d1)	Start device for storing a response frame	☞ Page 1151 Response frame	Device name	ANY16_ARRAY*2
(d2)	Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d2)+1 also turns on.	—	Bit	ANYBIT_ARRAY*2 (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

*2 When specifying setting data by using a label, define an array to secure enough operation area.

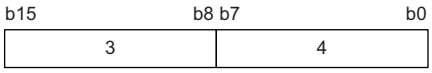
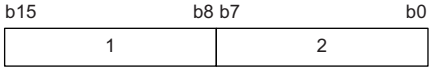
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	○	—	—	—	—	○	○	—	—	○
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○*1	—	—	—	—	○	—	—	—	—

*1 T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution/error completion type	b15 b7 b0 1 | 0 | (2) | 0 | (1) (1) Execution type (b0) • 0: Without arrival check (The instruction is regarded as completed when a request message is sent from own station.)*1 • 1: With arrival check (The instruction is regarded as completed when a response message is received from the external device.) (2) Error completion type (b7) Specify whether to set data at completion with an error. • 0: Do not set data at completion with an error in (s1)+13 and later. (Clear (s1)+13 and later.) • 1: Set data at completion with an error in (s1)+13 and later.	—	User
+1	Completion status	The completion status is stored upon completion of the instruction. 0: Completed successfully Other than 0: Completed with an error (error code)	—	System
+2	Own station channel	Specify the channel to be used by own station. Since whether or not a serial number*2 is given to the request message depends on the channel, specify the channel as follows according to the application. • 1: No serial number is given • 2 to 9: Serial number is given	1 to 9	User
+3	Own station port number	Specify the port number of the own station. When setting a channel where a serial number is given, be careful to specify the own station port number so that it does not overlap. When using in conjunction with the SLMPSND instruction, be careful not to overlap the channels because the channel where serial numbers are given is shared.	1 to 4999, 5010 to 65534 (1H to 1387H, 1392H to FFFEH)	User
+4	IP address of external device (third and fourth octets)	Specify the IP address (third and fourth octets) of external device. b15 b8 b7 b0 3 | 4 3, 4: Indicates the octets of the IP address.	00000001H to FFFFFFFEH (both (s1)+3 and (s1)+4 together) (1 to 4294967294)	User
+5	IP address of external device (first and second octets)	Specify the IP address (first and second octets) of external device. b15 b8 b7 b0 1 | 2 1, 2: Indicates the octets of the IP address.		User
+6	Destination port number	Specify the destination port number.	1 to 65534 (1 to FFFEH)	User
+7	Request destination network number	0000H (fixed)	0000H	User
+8	Request destination station number	00FFH (fixed)	00FFH	User

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+9	Request destination module I/O number	Specify a module I/O number of the request destination. • 03FFH: Own station/control CPU • 03E0H: Multiple CPU No.1 • 03E1H: Multiple CPU No.2 • 03E2H: Multiple CPU No.3 • 03E3H: Multiple CPU No.4	03FFH, 03E0H to 03E3H	User
+10	Request destination multidrop station number	0000H (fixed)	0000H	User
+11	Number of resends*3*4	The device becomes effective when the execution type specified by (s1)+0 is "1: With arrival check". ■Before instruction execution Specify the number of times that the request message will be resent if the response message is not received within the monitoring time specified by (s1)+11. • 0 to 15 (times) ■At completion of instruction The number of times that the request message has been resent (result) is stored. • 0 to 15 (times)	0 to 15	User/ system
+12	Arrival monitoring time	Specify the monitoring time until completion of processing. If processing is not completed within the monitoring time, the request is resent the number of times specified in (s1)+10. • 0: 10s • 1 to 32767: 1 to 32767s	0 to 32767	User
+13	Clock setting flag	The validity status (valid or invalid) of the data in (s)+13 and later is stored. Note that the data in (s1)+13 and later is cleared when the instruction is completed successfully. • 0: Invalid • 1: Valid	—	System
+14	Clock data	Upper 8 bits: Month (01H to 12H) Lower 8 bits: Year (00H to 99H: Lower two digits of the year)	—	System
+15		Upper 8 bits: Hour (00H to 23H) Lower 8 bits: Day (01H to 31H)		
+16		Upper 8 bits: Second (00H to 59H) Lower 8 bits: Minute (00H to 59H)		
+17		Upper 8 bits: Year (00H to 99H: Upper two digits of the year) Lower 8 bits: Day of the week (00H (Sun.) to 06H (Sat.))		
+18	IP address of error detected device (third and fourth octets)	The IP address (third and fourth octets) of the station where an error was detected is stored.  3, 4: Indicates the octets of the IP address.	—	System
+19	IP address of error detected device (first and second octets)	The IP address (first and second octets) of the station where an error was detected is stored.  1, 2: Indicates the octets of the IP address.	—	System

*1 If (s1)+0 is set to "0: Without arrival check", receive data is not set. Set (s1)+0 to "Without arrival check" when using a command that does not return a response message. (For example, the command that does not return a response message includes the remote reset command.)
When a command that return a response message is used, "With arrival check" must be set. If "Without arrival check" is set, an error may occur in the external device.

*2 Give the serial numbers when sending several request messages to the same SLMP-compatible device. Serial numbers to be given are automatically numbered by the system.
For the serial number, refer to the following.
SLMP Reference Manual

*3 This item does not apply to the number of resends of handshakes such as Client Hello that are sent when a DTLS connection is established with the destination device.

*4 When the request message send processing is executed, the request message may not be sent if the external device does not exist on the network, but the number of resends at the completion of the instruction is counted. At this time, the number of sent request messages does not match the number of resends at the completion of the instruction.

■Request frame

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Request data length	Specify the data length from the monitoring timer to the request data. (In units of bytes)	1 to 2000	User
+1	Monitoring timer	This timer sets the waiting time for the external device that received a request message to wait for the response after it issued a processing request to the access destination. (Unit: Increments of 250ms) • 0: Infinite wait • 1 to 65535: 1 to 65535 × 250ms	0 to 65535	User
+2 to +□	Request data	The request data of the SLMP message is stored.	—	User

■Response frame

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Response data length	The data length from the end code to the response data is stored. (In units of bytes)	2 to 2000	System
+1	End code	The result of command processing is stored. In normal end, 0 is stored. In abnormal end, an error code set by the external device is stored.	—	System
+2 to +□	Response data*1	Sets execution results for the request data. (Some commands do not return response data.)	—	System

*1 The response data is stored in units of bytes sequentially from lower bytes. When an odd number of bytes of response data is received, the last response data is stored in the lower byte of the last data storage area.

Processing details

- This instruction sends the request frame in the device specified by (s2) and later to the external device specified by the external device IP address in the control data. When a response message is received from the external device, it is stored in the device specified by (d1).
- This instruction uses DTLS communications. Set the external device to use DTLS.
- This instruction communicates in binary code. Match the setting of the external device also with the binary code.
- The execution status and the completion status of this instruction can be checked with the completion device (d2)+0 specified by the setup data and the completion status indication device (d2)+1.

• Completion device (d2)+0

This device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing.

• Completion status indication device (d2)+1

This device turns on or off depending on the completion status of this instruction.

Completed successfully: The device remains off.

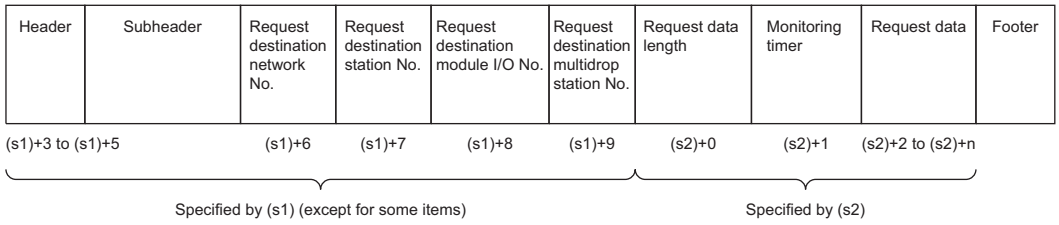
Completed with an error: The device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing.

- When an odd number of bytes of response data is received, invalid data is stored in the upper byte of the device where the last response data is stored.

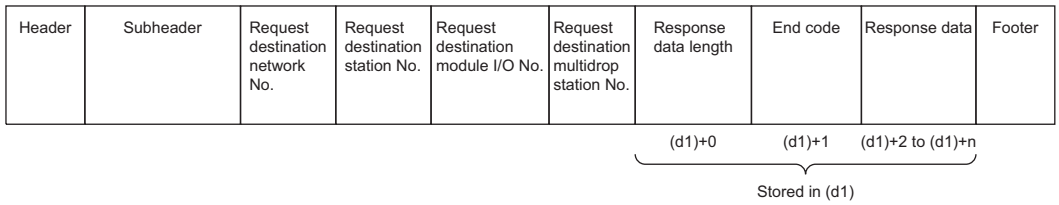
■Message format (3E/4E frame)

The following figures show the request data of 3E/4E frame and the response data in normal/abnormal end.

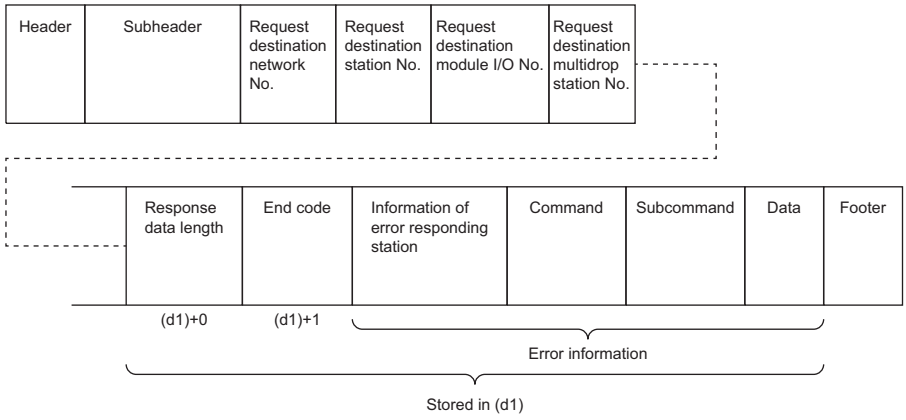
- Request message



- Response message (in normal end)



- Response message (in abnormal end)

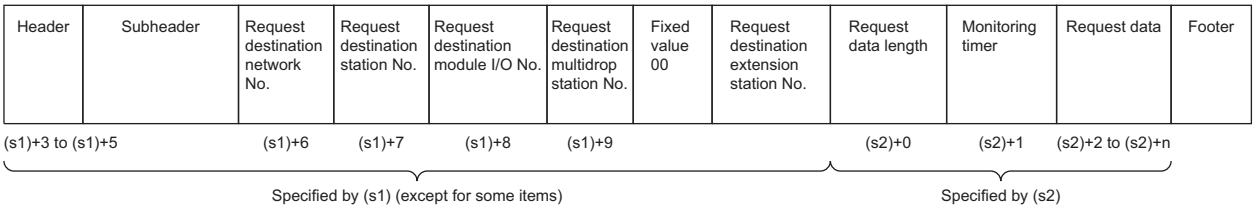


The network number, station number, request destination module I/O number, and multidrop station number are stored to the information of the error responding station.

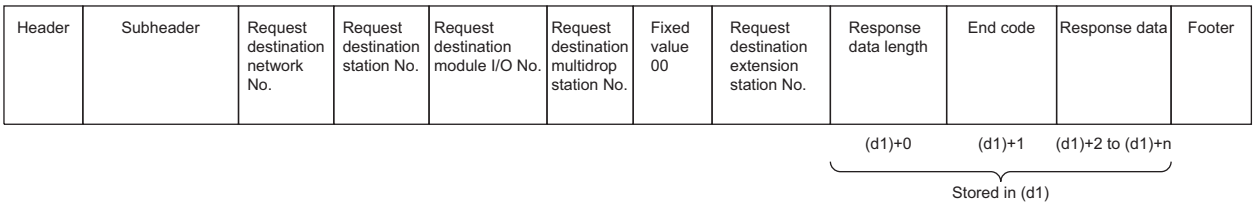
■Message format (station number extension frame)

The following figures show the request data of station number extension frame and the response data in normal/abnormal end.

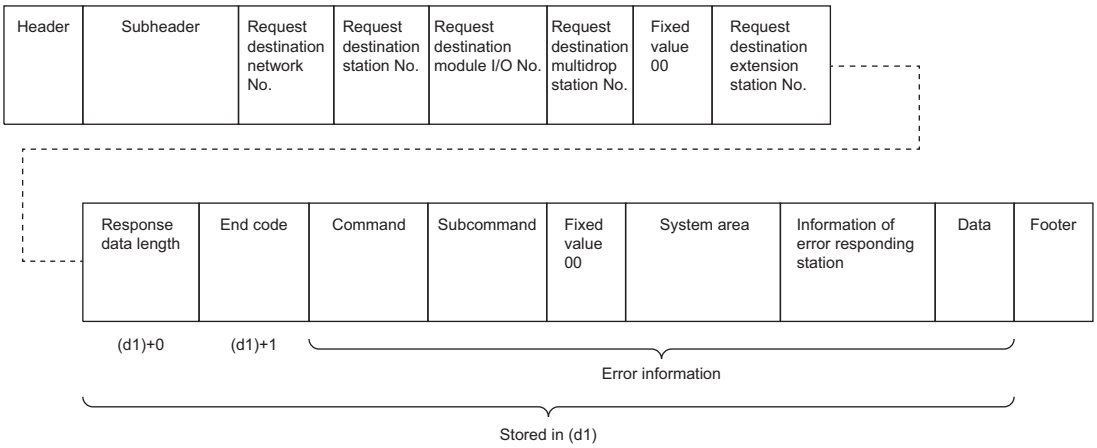
• Request message



• Response message (in normal end)



• Response message (in abnormal end)

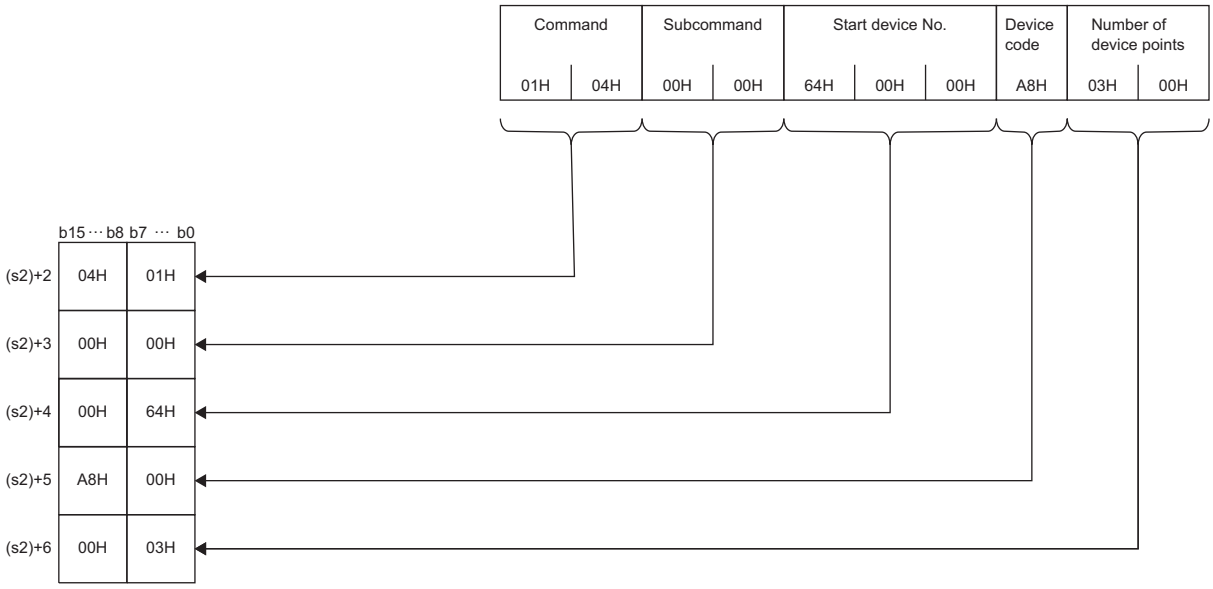


The network number, station number, request destination module I/O number, multidrop station number, 00 (fixed value), and extension station number are stored to the information of the error responding station.

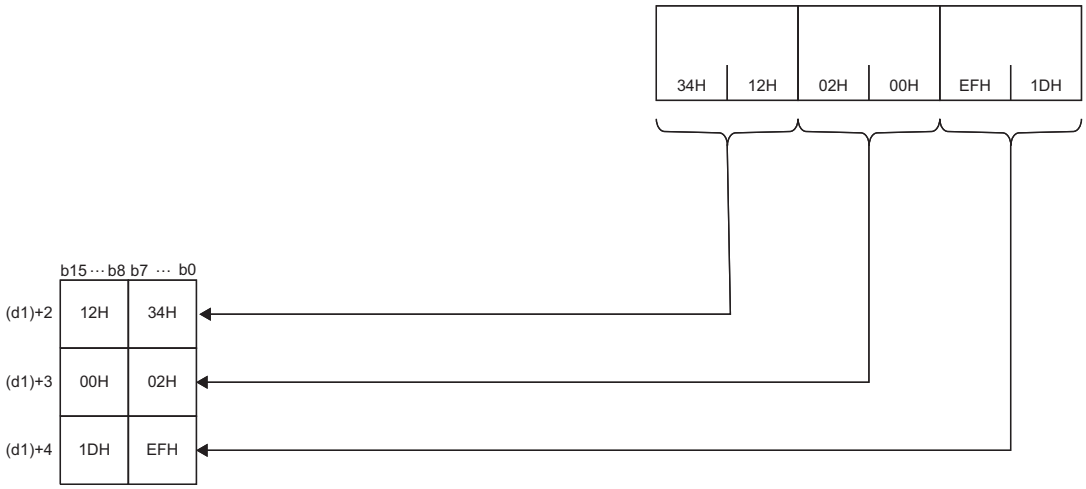
Ex.

When sending "Read (command: 0401H)" (reading in units of words) which reads the value in D100 to D102

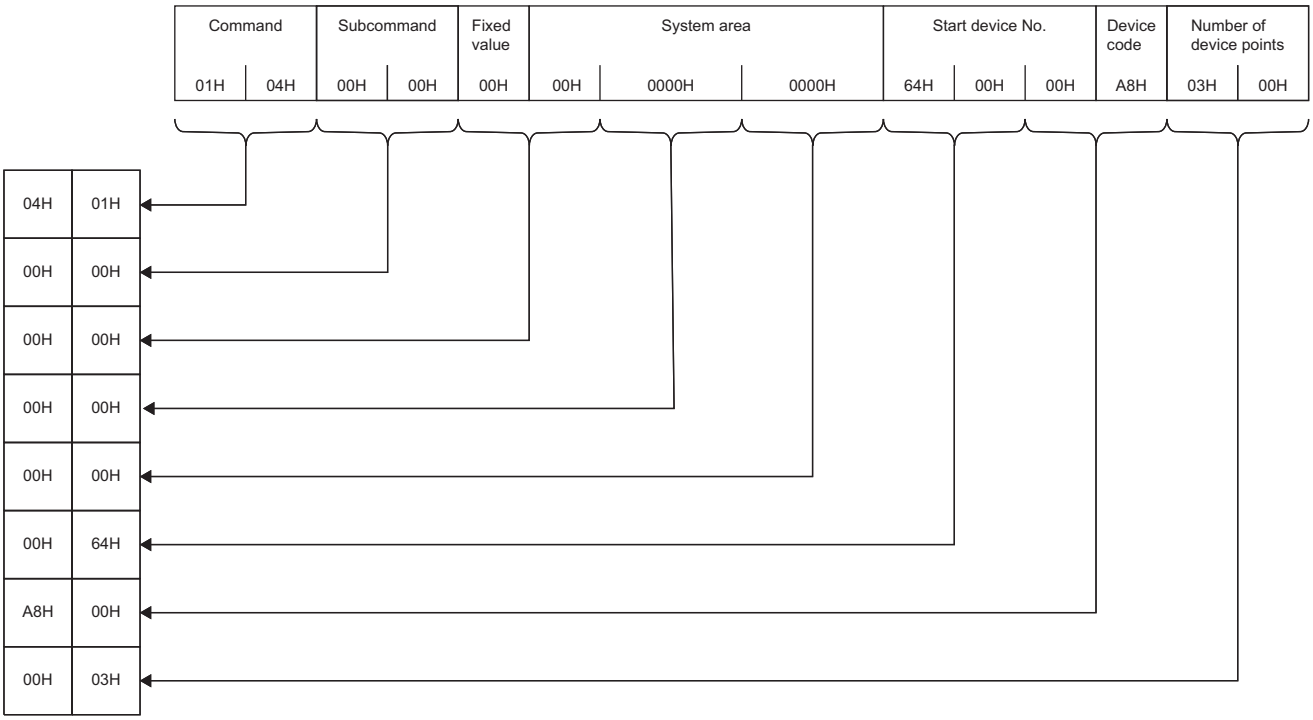
- The following example shows that request data is stored in (s2)+2 and later (3E/4E frame).



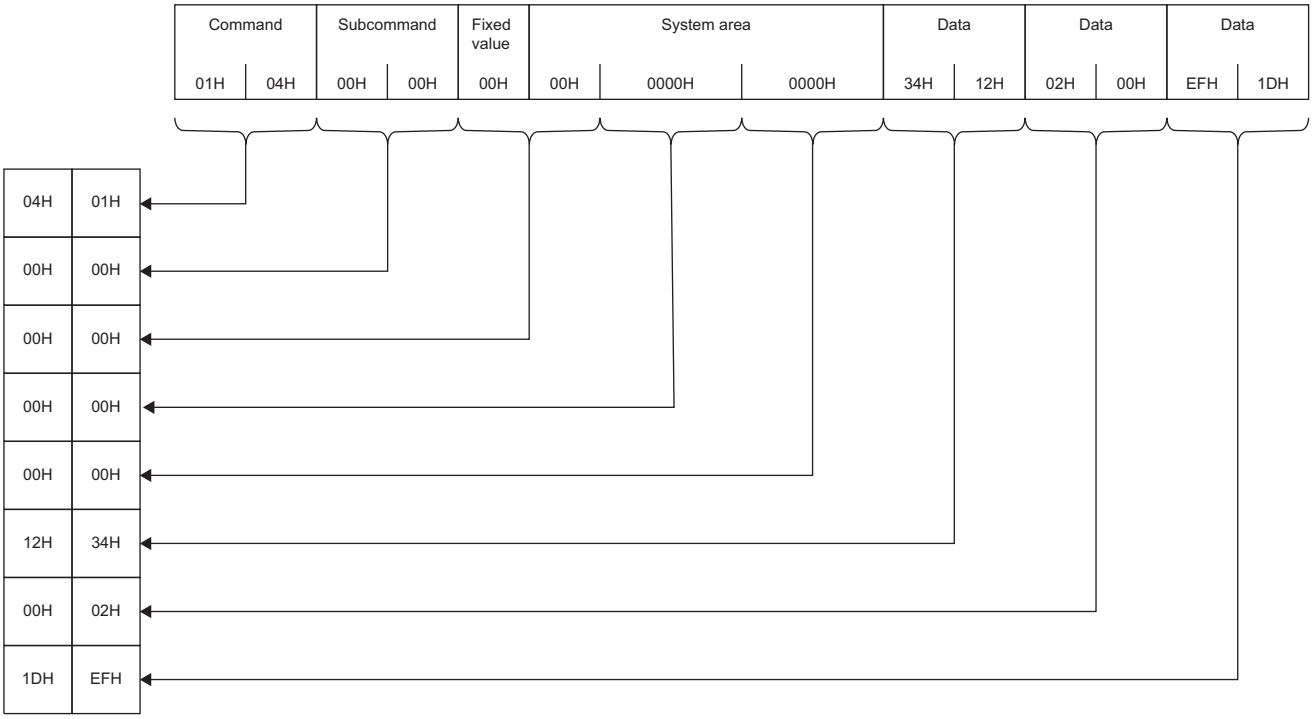
- The following example shows that response data is stored in (d1)+2 and later (3E/4E frame).



• The following example shows that request data is stored in (s2)+2 and later (station number extension frame).

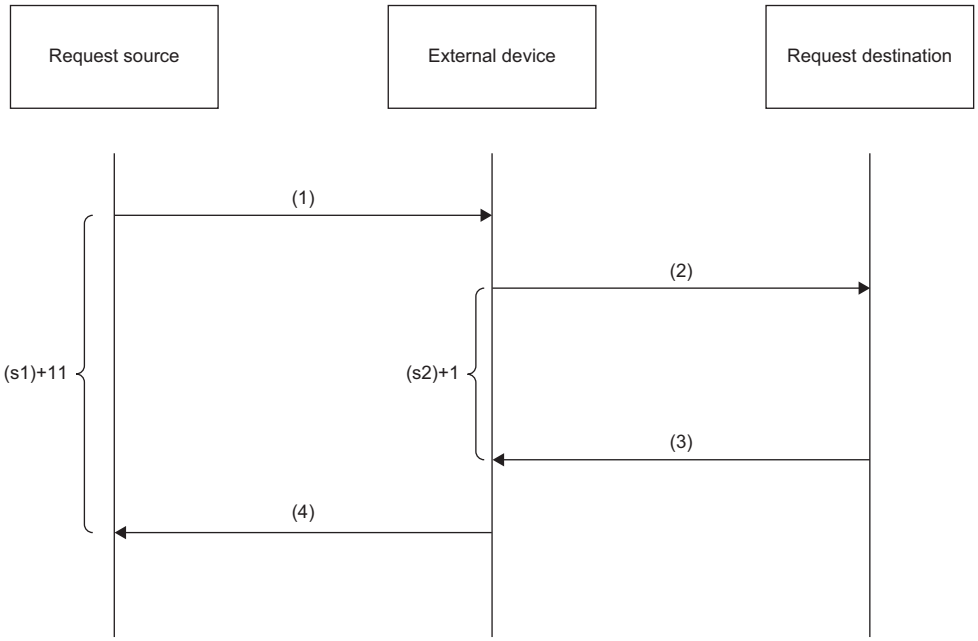


• The following example shows that response data is stored in (d1)+2 and later (station number extension frame).



Precautions

- When executing some of this instructions concurrently, be careful to set the channels of this instruction not to overlap. During the execution of this instruction, if another instance of this instruction specifying the same channel is executed, the later instruction is ignored. Upon completion of the first instance of this instruction, if the execution conditions for the later instance of this instruction are met, the later instruction will be requested consecutively. While the request for the later instruction is being ignored, SM699 (Dedicated instruction skip flag) turns on.
- Specify the arrival monitoring time ((s1)+11) of the control data and monitoring timer ((s2)+1) of the request frame so that the arrival monitoring time ≥ monitoring timer.



(1) Request message
(2) Processing request from external device to request destination
(3) Processing response from request destination to external device
(4) Response message

Point

This instruction is successfully completed even if the target device returns an abnormal response. When this instruction is completed successfully, the response is whether normal or abnormal can be identified by the end code of the response frame. When an abnormal response is returned, check the manual of the SLMP-compatible device being used and take corrective action.

Operation error

Error code (SD0)	Description
1193H	Different channels where serial numbers are given, having the same own station port number
3285H	The value set to (s1)+2 as own station channel is out of the range, 1 to 9.
	The value set to (s2)+0 as the request data length is 0 or exceeds 2000.

Upon completion with an error, the completion status indication device (d2)+1 is turned on and an error code is stored in the completion status (s1)+1. For the error code stored in the completion status (s1)+1, refer to the following.

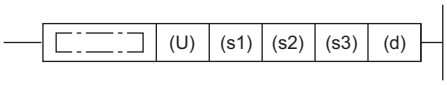
User's manual for the controller used

26.5 File Transfer Function Instructions

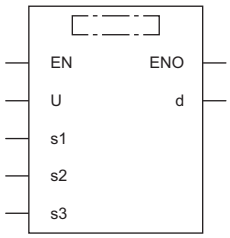
Sending FTP client files

GP.FTPCPUT, SP.FTPPUT

Sends files in the controller, which are specified by (s2), to the folder path of the FTP server, which is specified by (s3).

Ladder	ST
	ENO:=GP_FTPCPUT(EN,U,s1,s2,s3,d); ENO:=SP_FTPPUT(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.FTPCPUT SP.FTPPUT	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.FTPCPUT	Start I/O number of target (Specifies the first 3 digits of the start I/O number)	3E0H	16-bit unsigned binary	ANY16
	SP.FTPPUT	Dummy	—	String	ANYSTRING_SINGLE
(s1)	GP.FTPCPUT	Start device where control data is stored	Refer to the control data.	Device	ANY16
	SP.FTPPUT			Word	ANY16_ARRAY (Number of elements: 4)
(s2)		Name of files stored in the controller (transfer source)*1	—	Unicode string*2	ANYSTRING_DOUBLE
(s3)		Folder path of the FTP server (transfer destination)*1	—	Unicode string*2	ANYSTRING_DOUBLE
(d)		Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 Unicode string or the start device where the Unicode string is stored

*2 Even though the data type is Unicode string, only one-byte alphanumeric characters, symbols, and kana characters; and two-byte characters (Shift JIS codes) can be used. Unsupported characters are treated as "_".

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others (U)
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	GP.FTPCPUT	—	—	○	—	—	—	—	○	○	—	—	○
	SP.FTPPUT	—	—	○	—	—	—	—	○	—	—	○	○
(s1)		—	—	○	—	—	—	—	○	—	—	—	—
(s2)		—	—	○	—	—	—	—	○	—	—	○	—
(s3)		—	—	○	—	—	—	—	○	—	—	○	—
(d)		○	—	○*1	—	—	—	—	○	—	—	—	—

*1 T, ST, C, and FD cannot be used.

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Application setting area	b15 … b3 b2 b1 b0 0 (2) (1) 0 (1) Transfer completion file delete setting (b2) Specify whether to delete transfer completion files or not. • 0: Do not delete • 1: Delete (2) Temporary file create setting (b3) Specify whether to create a temporary file (transfer source file name including extension.TMP) during the file transfer processing or not. Setting this bit to 0 prevents files from becoming undefined in the transfer destination when a cable is disconnected or power is shut off during the file transfer processing. • 0: Create • 1: Do not create	Refer to the "Description" column.	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+2	Total number of files to be transferred	The total number of files to be transferred by this instruction is stored.	—	System
+3	Number of transferred files	The number of transferred files is stored.	—	System

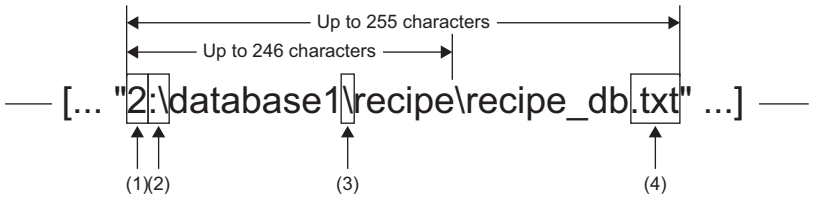
Processing details

- Sends files in the controller, which are specified by (s2), to the folder path of the FTP server, which is specified by (s3). The controller opens a connection with the FTP server set in the module parameters ("FTP Client Settings") at execution of this instruction, and closes a connection after sending files. For details on the parameter setting, refer to the following.

User's manual for the controller used

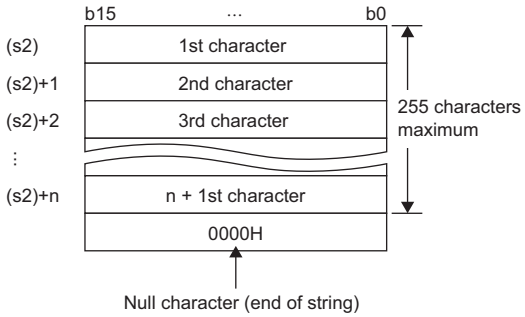
- The total number of files to be transferred by this instruction is stored in (s1)+2, and the number of transferred files is stored in (s1)+3. However, if the FTP server cannot be connected at execution of the instruction, the number of transferred files may be 0.

- Specify the transfer source drive number (2 to 4, or 6) of the controller, the folder path where the files are stored, and the file name (including a period and an extension) in (s2) in Unicode string in 64 characters or less. The maximum number of characters used in a file path is 255. The maximum number of characters used in a path, excluding the file name, is 246 (not including a delimiter). Use one-byte '\' or '/' as a delimiter to specify the boundaries between the elements in a file path. *1



(1) Drive numbers that can be specified are 2 to 4, or 6. (User's manual for the controller used)
(2) Use one-byte ':\' or ':/' as a delimiter of the drive number.
(3) Use one-byte '\' or '/' as a delimiter of the folder path and file.
(4) The specified file name should include an extension.

*1 A relative path cannot be specified for the FTP client path.



- Wild card characters (*, ?) can be used to specify the file name and extension of (s2).

Symbol	Description
*	An asterisk '*' is replaced with any character or string (including none) in a file name.
?	A question mark '?' is replaced with a character (excluding none) in a file name. ('?' can be used multiple times.)

A period is a wild card target character only when "?" is used.

Using wild card characters in the following ways results in an error.

- Two or more asterisks '*' are used in a file name (before the period) or an extension. (Example: *abc*.txt)
- An asterisk '*' and a question mark '?' are used in a file name (before the period) or an extension. (Example: *ab?. txt)

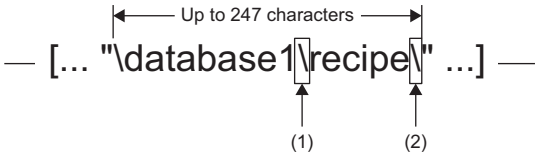
When any wild card character is used, the number of files that can be transferred is determined by the total size of the file names of the specified files. The specified files can be transferred when the number of these files and the total size of the file names satisfy the following condition. If a file transfer function instruction is executed without satisfying the following condition, the instruction completes with an error.

(Fi + NM) + 1 < 65536 [bytes]

N: Total number of files that match the wild card specification
Fi: Total size of the file names that match the wild card specification
M: Specific information size (Fixed value: 6 bytes)

- If an error occurs in any one of the files to be transferred during execution of this instruction, the transfer processing will be stopped upon detection of the error, and the rest of the target files will not be transferred.
- Untransferable files will not be transferred even though the wild card specification conditions are satisfied.
- If the number of characters in the file path which includes a file name and an extension exceeds 255, files will not be transferred even though the wild card specification conditions are satisfied.
- If the number of characters in the file name which includes an extension exceeds 64, files will not be transferred even though the wild card specification conditions are satisfied.

- Specify the folder path of the transfer destination FTP server in (s3) in Unicode string. The specified folder path shall be a relative path or an absolute path from home directory of the FTP server. Use one-byte '\' or '/' as a delimiter to specify the boundary of the folder path. *1 The maximum number of characters used in the folder path is 255. The maximum number of characters used in a path, excluding the file name, is 246 (not including a delimiter). Note that the total number of characters in a folder path (including the delimiter at the end) and the file name specified in (s2) must be within the maximum path length supported by the transfer destination FTP server. The delimiter at the end of a string can be omitted. When omitted, '\' is assumed to be set at the end. If a nonexistent folder path is specified, a folder is automatically created by the system at execution of the instruction, and then the processing is performed.*2



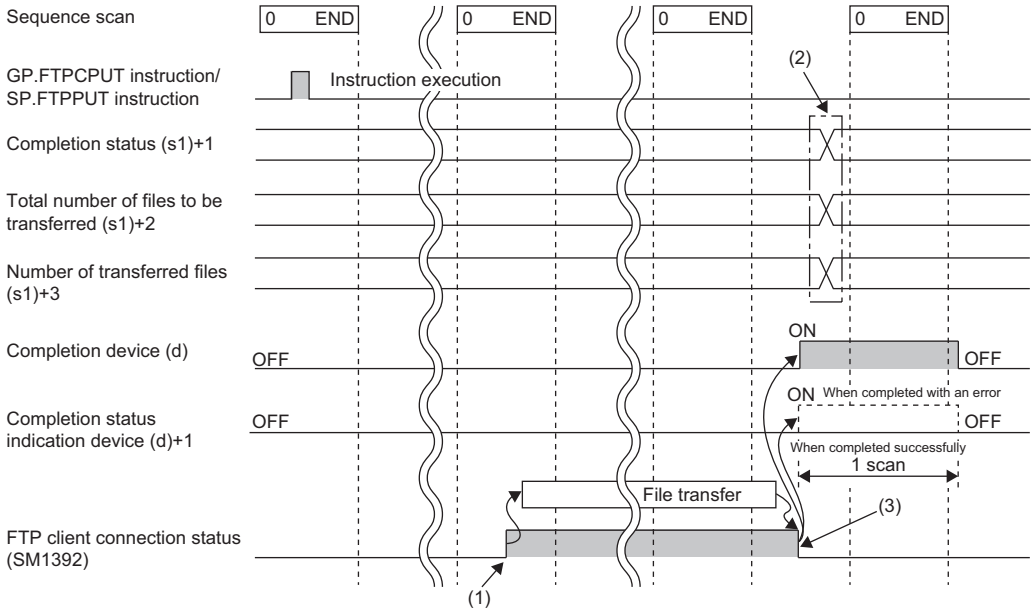
(1) Use one-byte '\' or '/' as a delimiter to specify the boundary of the folder path.*3
(2) The delimiter at the end of string can be omitted.

*2 FTP server drive cannot be specified.
*3 Note that '\' cannot be used as a delimiter for some FTP servers.

- If a NULL character is specified by (s3) or only "0000H" is specified for the device, the controller directly accesses under the home directory of the FTP server. For details, follow the FTP server specifications.
- If a file with the same name exists in the transfer destination, the file will be overwritten.
- The maximum size of a file that can be sent is 4294967295 bytes (4G bytes - 1 byte).
- The execution status and the completion status of this instruction can be checked with the completion device (d) specified by the setup data and the completion status indication device (d)+1.

• Completion device (d)
This device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing.
• Completion status indication device (d)+1
This device turns on or off depending on the completion status of this instruction.
Completed successfully: The device remains off.
Completed with an error: The device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing. In addition, an error code is stored in the device specified by (s1)+1.

• The following figure shows the execution timing of this instruction.



(1) SM1392 turns on during the END processing after the controller is connected to the FTP server.
(2) Values are stored upon completion of the instruction.
(3) When all files have been transferred, SM1392 turns off.

- SM1392 (FTP client connection status) is on while the controller is connected to the FTP server, and SM1392 turns off when disconnected.
- SM753 (File being accessed) turns on while this instruction is being executed. While SM753 is on, this instruction cannot be executed. If executed, no processing is performed.
- During the execution of this instruction, GP.FTPCGET instruction, or SP.FTPGET instruction, if another instance of this instruction is executed, the later instruction is ignored. Upon completion of the first instruction, if the execution conditions for the later instruction are met, the later instruction will be requested consecutively. While the later instruction is being ignored, SM699 (Dedicated instruction skip flag) turns on.
- Regardless of the setting of bit3 (temporary file creation setting during file transfer) of (s1) + 0, if there is no free space in the transfer destination, the instruction completes with an error.
- Even though the operating status of the controller is switched RUN to STOP during the file transfer processing, the controller continues the processing until completed.

Precautions

- If a cable is disconnected, power is shut off, or the controller is reset during the file transfer processing, delete unnecessary files (such as a temporary file and undefined files) on the FTP server as needed. Then, transfer files again. In addition, note that the data being written may be lost if a power off occurs during the file transfer.
- When b2 (Transfer completion file delete setting) of (s1)+0 is set to 1 (Delete), note the following.

Item	Description
When files in the controller are transferred	Files required for the controller to operate are also deleted. If deleted, operations of the controller cannot be guaranteed.
When wild card characters are used to specify a file name	Required files may be deleted unintentionally.

- When transfer source file access is restricted by the file password function, this instruction completes with an error. Note, however, that this instruction can be executed if b2 (Transfer completion file delete setting) of (s1)+0 is set to 0 (Do not delete) and the file password setting type is "Write Protection".
- When bit3 (temporary file creation setting during file transfer) of (s1)+0 is set to 0 (Create), a temporary file of up to 68 characters (transfer source file name including period and extension.TMP) will be created in the transfer destination. Therefore, set the transfer destination folder path so that the total number of characters in the folder path and the temporary file does not exceed 255. Ensure that the number of characters in the file name does not exceed 64.
- Do not use any unsupported characters. If an unsupported character is included in the file name or the folder name specified by (s2) and (s3), the character is converted to "_" and processed. When wild card characters are used and an unsupported character is included in the name of files stored in the transfer source controller, the character is converted to "_" and processed. For this reason, the corresponding file is transferred in the same way as files having the same file name or folder name after character conversion are transferred.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (0000H) in the assignment range of the relevant device/label for the device number specified by (s2), (s3), and later.
3285H	The number of characters in the string specified by (s2) exceeds 255.
	The number of characters in the path specified by (s2) or (s3), excluding the file name, exceeds 246 (not including a delimiter).
	The total number of characters in the strings specified by (s2) (only the file name part, excluding the drive number and folder path) and (s3) exceeds 255.*1*2
	The drive number specified by (s2) is out of range.
	The number of characters in the file name specified by (s2) exceeds 64.*2
3296H	A file name is not specified by (s2).
	The file name that cannot be transferred is specified by (s2).
	The delimiter used to separate the drive number in (s2) is neither ':\' nor ':/'.
	Two or more asterisks '*' are used in the file name (before the period) or the extension specified by (s2).
	An asterisk '*' and a question mark '?' are used in the file name (before the period) or the extension specified by (s2).
	Wild card characters, '*' and '?', are used in the string specified by (s3).
329BH	This instruction was executed without setting FTP client parameters.

*1 When the delimiter at the end in a folder path is abbreviated by (s3), the maximum number of characters is 254.

*2 When the temporary file creation setting for file transfer is set to "0: Create", the file name will include an extension of four characters ".TMP".

Upon completion with an error, the completion status indication device (d)+1 is turned on and an error code is stored in the completion status (s1)+1.

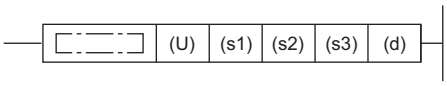
For the error code stored in the completion status (s1)+1, refer to the following.

User's manual for the controller used

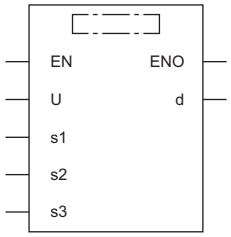
Retrieving FTP client files

GP.FTPCGET, SP.FTPGET

Retrieves files on the FTP server, which are specified by (s2), to the folder path of the controller, which is specified by (s3).

Ladder	ST
	ENO:=GP_FTPCGET(EN,U,s1,s2,s3,d); ENO:=SP_FTPGET(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
GP.FTPCGET SP.FTPGET	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(U)	GP.FTPCGET	Start I/O number of target (Specifies the first 3 digits of the start I/O number)	3E0H	16-bit unsigned binary	ANY16
	SP.FTPGET	Dummy	—	String	ANYSTRING_SINGLE
(s1)	GP.FTPCGET	Start device where control data is stored	Refer to the control data.	Device	ANY16
	SP.FTPGET			Word	ANY16_ARRAY (Number of elements: 4)
(s2)		Name of files stored in the FTP server (transfer source)*1	—	Unicode string*2	ANYSTRING_DOUBLE
(s3)		Folder path of the controller (transfer destination)*1	—	Unicode string*2	ANYSTRING_DOUBLE
(d)		Device that turns on for one scan upon completion of the instruction When the instruction completes with an error, (d)+1 also turns on.	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
EN		Execution condition	—	Bit	BOOL
ENO		Execution result	—	Bit	BOOL

*1 Unicode string or the start device where the Unicode string is stored

*2 Even though the data type is Unicode string, only one-byte alphanumeric characters, symbols, and kana characters; and two-byte characters (Shift JIS codes) can be used. Unsupported characters are treated as "_".

■Applicable devices

Operand		Bit		Word			Double word		Indirect specification	Constant			Others (U)
		X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	GP.FTPCGET	—	—	○	—	—	—	—	○	○	—	—	○
	SP.FTPGET	—	—	○	—	—	—	—	○	—	—	○	○
(s1)		—	—	○	—	—	—	—	○	—	—	—	—
(s2)		—	—	○	—	—	—	—	○	—	—	○	—
(s3)		—	—	○	—	—	—	—	○	—	—	○	—
(d)		○	—	○*1	—	—	—	—	○	—	—	—	—

*1 T, ST, C, and FD cannot be used.

■Control data

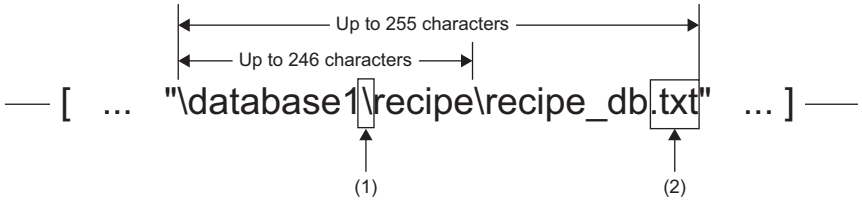
Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Application setting area	b15 … b3 b2 b1 b0 0 (2) (1) 0 (1) Transfer completion file delete setting (b2) Specify whether to delete transfer completion files or not. • 0: Do not delete • 1: Delete (2) Temporary file create setting (b3) Specify whether to create a temporary file (transfer source file name including extension.TMP) during the file transfer processing or not. Setting this bit to 0 prevents files from becoming undefined in the transfer destination when a cable is disconnected or power is shut off during the file transfer processing. • 0: Create • 1: Do not create	Refer to the "Description" column.	User
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+2	Total number of files to be transferred	The total number of files to be transferred by this instruction is stored.	—	System
+3	Number of transferred files	The number of transferred files is stored.	—	System

Processing details

- Retrieves files on the FTP server, which are specified by (s2), to the folder path of the controller, which is specified by (s3). The controller opens a connection with the FTP server set in the module parameters ("FTP Client Settings") at execution of this instruction, and closes a connection after retrieving files. For details on the parameter setting, refer to the following.

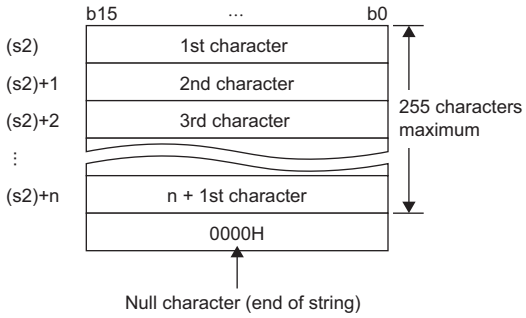
User's manual for the controller used

- The total number of files to be transferred by this instruction is stored in (s1)+2, and the number of transferred files is stored in (s1)+3. However, if the SD memory card is set as the destination at execution of the instruction and the SD memory card is removed during file transfer, the number of transferred files may be 0.
- Specify the folder path where the transfer source files are stored on the FTP server, and the file names (including a period and an extension) in (s2) in Unicode string in 64 characters or less. The maximum number of characters used in a file path is 255. The maximum number of characters used in a path, excluding the file name, is 246 (not including a delimiter). The specified file path shall be a relative path or an absolute path from home directory of the FTP server. Use one-byte '\' or '/' as a delimiter to specify the boundaries between the folder path and the file name. *1*2



(1) Use one-byte '\' or '/' as a delimiter of the folder path or file.*2
(2) The specified file name should include an extension.

*1 FTP server drive cannot be specified.
*2 Note that '\' cannot be used as a delimiter for some FTP servers.



- Wild card characters (*, ?) can be used to specify the file name and extension of (s2).

Symbol	Description
*	An asterisk '*' is replaced with any character or string (including none) in a file name.
?	A question mark '?' is replaced with a character (excluding none) in a file name. ('?' can be used multiple times.)

A period is a wild card target character only when "?" is used.

Using wild card characters in the following ways results in an error.

- Two or more asterisks '*' are used in a file name (before the period) or an extension. (Example: *abc*.txt)
- An asterisk '*' and a question mark '?' are used in a file name (before the period) or an extension. (Example: *ab?. txt)

Files in the subfolder cannot be transferred.

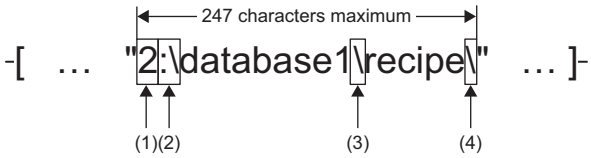
When any wild card character is used, files to be retrieved differ depending on the FTP server environment. Check the operating specifications of the FTP server.

When any wild card character is used, the number of files that can be transferred is determined by the total size of the file names of the specified files. The specified files can be transferred when the number of these files and the total size of the file names satisfy the following condition. If a file transfer function instruction is executed without satisfying the following condition, the instruction completes with an error.

(Fi + NM) + 1 < 65536 [bytes]

N: Total number of files that match the wild card specification
Fi: Total size of the file names that match the wild card specification
M: Specific information size (Fixed value: 6 bytes)

- If an error occurs in any one of the files to be retrieved during execution of this instruction, the transfer processing will be stopped upon detection of the error, and the rest of the target files will not be transferred.
- Untransferable files will not be transferred even though the wild card specification conditions are satisfied.
- If the number of characters in the file path which includes a file name and an extension exceeds 255, files will not be transferred even though the wild card specification conditions are satisfied.
- If the number of characters in the file name which includes an extension exceeds 64, files will not be transferred even though the wild card specification conditions are satisfied.
- If only a file name is specified by (s2), the controller directly accesses under the home directory of the FTP server. When specifying only a file name, start with a delimiter. For details, follow the FTP server specifications.
- Specify the folder path of the transfer destination of the controller in (s3) in Unicode string. Use one-byte '\' or '/' as a delimiter to specify the boundary of the folder path. The maximum number of characters in the folder path is 247 (246 when a delimiter at the end of the string is omitted). Note that the total number of characters in a folder path (including the delimiter at the end) and the file name specified in (s2) must be within the maximum path length (255 characters) supported by the controller. The delimiter at the end of a string can be omitted. When omitted, '\' is assumed to be set at the end. If a nonexistent folder path is specified, a folder is automatically created by the system at execution of the instruction, and then the processing is performed. *3



(1) Drive numbers that can be specified are 2 to 4. (User's manual for the controller used)
(2) Use one-byte ':\' or ':/' as a delimiter of the drive number.
(3) Use one-byte '\' or '/' as a delimiter to specify the boundary of the folder path.
(4) The delimiter at the end of string can be omitted.

*3 A relative path cannot be specified for the FTP client path.

- If a NULL character is specified by (s3) or only "0000H" is specified for the device, the controller directly accesses under the home directory (4:\) of the FTP server.
- If a file with the same name exists in the transfer destination, the file will be overwritten.
- The maximum size of a file that can be retrieved is 4294967295 bytes (4G bytes - 1 byte).
- The execution status and the completion status of this instruction can be checked with the completion device (d) specified by the setup data and the completion status indication device (d)+1.

• Completion device (d)

This device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing.

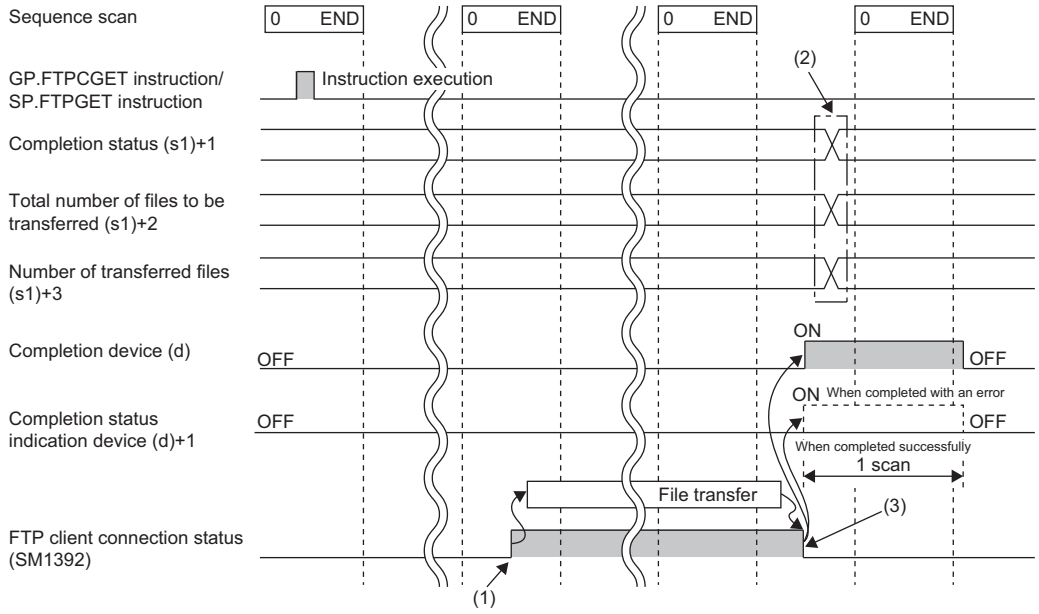
• Completion status indication device (d)+1

This device turns on or off depending on the completion status of this instruction.

Completed successfully: The device remains off.

Completed with an error: The device turns on during END processing of the scan where this instruction completes, and turns off during the next END processing. In addition, an error code is stored in the device specified by (s1)+1.

- The following figure shows the execution timing of this instruction.



(1) SM1392 turns on during the END processing after the controller is connected to the FTP server.
(2) Values are stored upon completion of the instruction.
(3) When all files have been transferred, SM1392 turns off.

- SM1392 (FTP client connection status) is on while the controller is connected to the FTP server, and SM1392 turns off when disconnected.
- SM753 (File being accessed) turns on while this instruction is being executed. While SM753 is on, this instruction cannot be executed. If executed, no processing is performed.
- During the execution of the GP.FTPCPUT instruction, SP.FTPPUT instruction, or this instruction, if another instance of this instruction is executed, the later instruction is ignored. Upon completion of the first instruction, if the execution conditions for the later instruction are met, the later instruction will be requested consecutively. While the later instruction is being ignored, SM699 (Dedicated instruction skip flag) turns on.
- Regardless of the setting of bit3 (temporary file creation setting during file transfer) of (s1) + 0, if there is no free space in the transfer destination, the instruction completes with an error.
- Even though the operating status of the controller is switched RUN to STOP during the file transfer processing, the controller continues the processing until completed.

Precautions

- If a cable is disconnected, power is shut off, or the controller is reset during the file transfer processing, transfer files again. In those cases, unnecessary files will be deleted in the following ways. In addition, note that the data being written may be lost if a power off occurs during the file transfer.

Unnecessary file	Delete operation
Temporary file (transfer source file name including extension.TMP)	A temporary file left in the FTP client will be automatically deleted at the start of next file transfer processing to the same folder.
Undefined transfer target file	An undefined transfer target file left in the FTP client will be overwritten during next file transfer processing to the same file.

- When a file with the same name has already existed in the transfer destination and its access is restricted by the file password function, this instruction completes with an error.
- When b3 (Temporary file create setting) of (s1)+0 is set to 0 (Create), set the maximum number of characters in the folder path of the controller to 243 (242 when a delimiter at the end of the string is omitted). Since a temporary file of up to 68 characters (transfer source file name including period and extension.TMP) is created in the transfer destination, set the transfer destination folder path so that the total number of characters in the folder path and the temporary file does not exceed the maximum path length (255 characters) supported by the controller. Ensure that the number of characters in the file name does not exceed 64.
- Do not use any unsupported characters. If an unsupported character is included in the file name or the folder name specified by (s2) and (s3), the character is converted to "_" and processed. For this reason, the corresponding file is transferred in the same way as files having the same file name or folder name after character conversion are transferred. When wild card characters are used and an unsupported character is included in the name of files stored in the transfer source FTP server, files cannot be retrieved properly. (How to treat unsupported characters depends on the specifications of the FTP server.)

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (0000H) in the assignment range of the relevant device/label for the device number specified by (s2), (s3), and later.
3285H	The number of characters in the string specified by (s2) exceeds 255.
	The number of characters in the path specified by (s2) or (s3), excluding the file name, exceeds 246 (not including a delimiter).
	The total number of characters in the strings specified by (s3) and (s2) (only the file name part) exceeds 255.*1*2
	The drive number specified by (s3) is out of range.
	The number of characters in the file name specified by (s2) exceeds 64.*2
3296H	A file name is not specified by (s2).
	The file name that cannot be transferred is specified by (s2).
	The delimiter used to separate the drive number in (s3) is neither ':\' nor ':/'.
	Two or more asterisks '*' are used in the file name (before the period) or the extension specified by (s2).
	An asterisk '*' and a question mark '?' are used in the file name (before the period) or the extension specified by (s2).
	Wild card characters, '*' and '?', are used in the string specified by (s3).
329BH	This instruction was executed without setting FTP client parameters.

*1 When the delimiter at the end in a folder path is abbreviated by (s2), the maximum number of characters is 254.
*2 When the temporary file creation setting for file transfer is set to "0: Create", the file name will include an extension of four characters ".TMP".

Upon completion with an error, the completion status indication device (d)+1 is turned on and an error code is stored in the completion status (s1)+1.

For the error code stored in the completion status (s1)+1, refer to the following.

User's manual for the controller used

PART 6 STANDARD FUNCTIONS

This part consists of the following chapters.

27 TYPE CONVERSION FUNCTIONS

28 SINGLE VARIABLE FUNCTIONS

29 ARITHMETIC OPERATION FUNCTIONS

30 BIT SHIFT FUNCTIONS

31 BOOLEAN FUNCTIONS

32 SELECTION FUNCTIONS

33 COMPARISON FUNCTIONS

34 STRING FUNCTIONS

35 TIME DATA TYPE FUNCTIONS

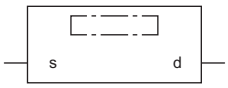
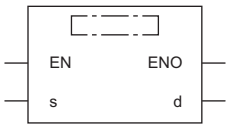
27 TYPE CONVERSION FUNCTIONS

27.1 Converting BOOL to WORD

BOOL_TO_WORD(_E)

These functions convert a value from BOOL data type to WORD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=BOOL_TO_WORD(s);
		[With EN/ENO]
		d:=BOOL_TO_WORD_E(EN,ENO,s);

Setting data

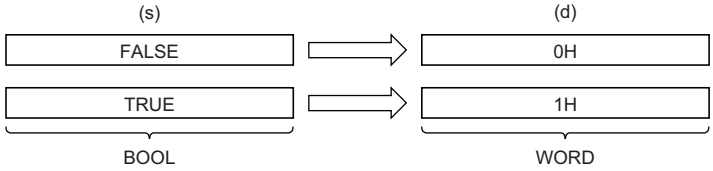
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	WORD

Processing details

■Operation processing

- These functions convert the value input to (s) from BOOL data type to WORD data type, and output the converted value from (d).
- When the input value is FALSE, 0H (WORD data type) is output.
- When the input value is TRUE, 1H (WORD data type) is output.



- Input a BOOL data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

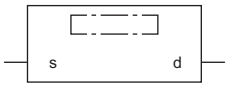
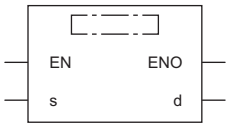
27.2 Converting BOOL to DWORD

BOOL_TO_DWORD(_E)

These functions convert a value from BOOL data type to DWORD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=BOOL_TO_DWORD(s);
		[With EN/ENO]
		d:=BOOL_TO_DWORD_E(EN,ENO,s);

Setting data

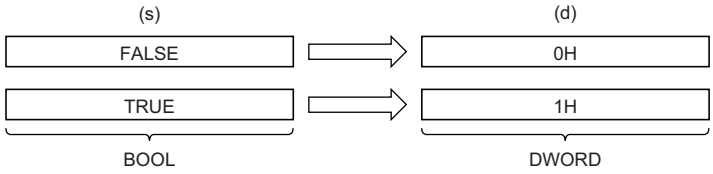
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DWORD

Processing details

■Operation processing

- These functions convert the value input to (s) from BOOL data type to DWORD data type, and output the converted value from (d).
- When the input value is FALSE, 0H (DWORD data type) is output.
- When the input value is TRUE, 1H (DWORD data type) is output.



- Input a BOOL data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

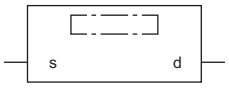
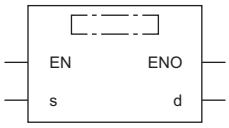
27.3 Converting BOOL to INT

BOOL_TO_INT(_E)

These functions convert a value from BOOL data type to INT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=BOOL_TO_INT(s);
		[With EN/ENO]
		d:=BOOL_TO_INT_E(EN,ENO,s);

Setting data

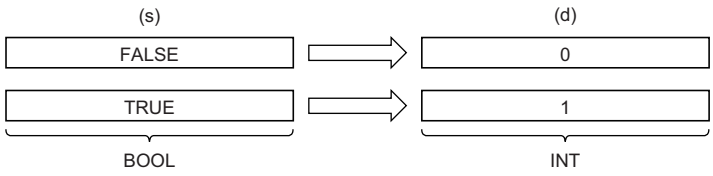
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert the value input to (s) from BOOL data type to INT data type, and output the converted value from (d).
- When the input value is FALSE, 0 (INT data type) is output.
- When the input value is TRUE, 1 (INT data type) is output.



- Input a BOOL data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

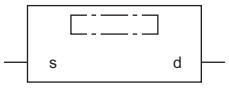
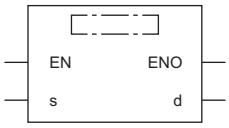
27.4 Converting BOOL to DINT

BOOL_TO_DINT(_E)

These instructions convert a value from BOOL data type to DINT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=BOOL_TO_DINT(s);
		[With EN/ENO]
		d:=BOOL_TO_DINT_E(EN,ENO,s);

Setting data

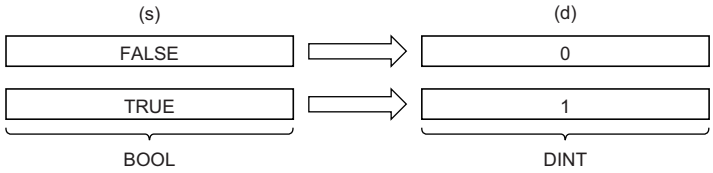
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert the value input to (s) from BOOL data type to DINT data type, and output the converted value from (d).
- When the input value is FALSE, 0 (DINT data type) is output.
- When the input value is TRUE, 1 (DINT data type) is output.



- Input a BOOL data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

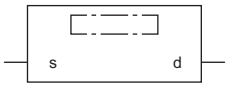
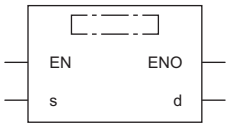
27.5 Converting BOOL to TIME

BOOL_TO_TIME(_E)

These functions convert a value from BOOL data type to TIME data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=BOOL_TO_TIME(s);
		[With EN/ENO]
		d:=BOOL_TO_TIME_E(EN,ENO,s);

Setting data

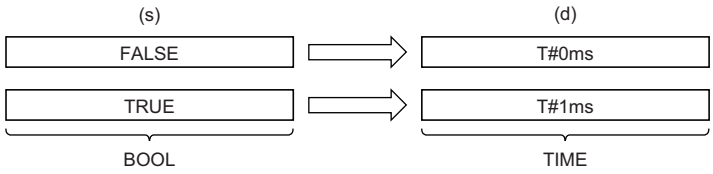
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	TIME

Processing details

■Operation processing

- These functions convert the value input to (s) from BOOL data type to TIME data type, and output the converted value from (d).
- When the input value is FALSE, 0 (TIME data type) is output.
- When the value is TRUE, 1 (TIME data type) is output.



- Input a BOOL data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

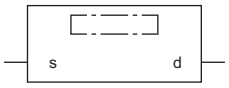
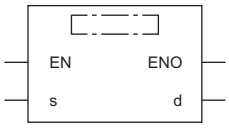
27.6 Converting BOOL to STRING

BOOL_TO_STRING(_E)

These functions convert a value from BOOL data type to STRING data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=BOOL_TO_STRING(s);
		[With EN/ENO]
		d:=BOOL_TO_STRING_E(EN,ENO,s);

Setting data

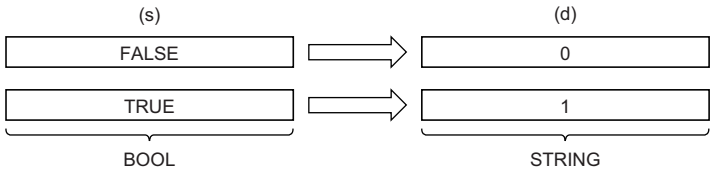
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	STRING

Processing details

■Operation processing

- These functions convert the value input to (s) from BOOL data type to STRING data type, and output the converted value from (d).
- When the input value is FALSE, 0 (STRING data type) is output.
- When the input value is TRUE, 1 (STRING data type) is output.



- Input a BOOL data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

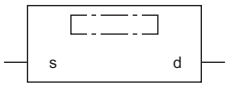
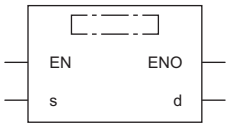
27.7 Converting WORD to BOOL

WORD_TO_BOOL(_E)

These functions convert a value from WORD data type to BOOL data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=WORD_TO_BOOL(s);
		[With EN/ENO]
		d:=WORD_TO_BOOL_E(EN,ENO,s);

Setting data

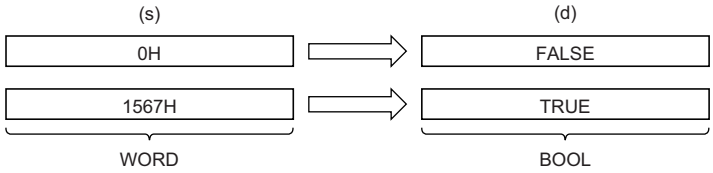
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	WORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	BOOL

Processing details

■Operation processing

- These functions convert the value input to (s) from WORD data type to BOOL data type, and output the converted value from (d).
- When the input value is 0H, FALSE is output.
- When the input value is other than 0H, TRUE is output.



- Input a WORD data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

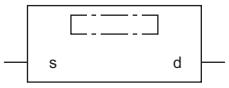
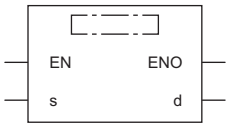
27.8 Converting WORD to DWORD

WORD_TO_DWORD(_E)

These functions convert a value from WORD data type to DWORD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=WORD_TO_DWORD(s);
		[With EN/ENO]
		d:=WORD_TO_DWORD_E(EN,ENO,s);

Setting data

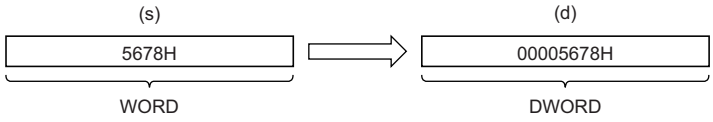
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	WORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DWORD

Processing details

■Operation processing

- These functions convert the value input to (s) from WORD data type to DWORD data type, and output the converted value from (d).
- After the data type is converted, the upper 16 bits are filled with 0s.



- Input a WORD data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

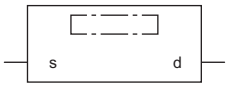
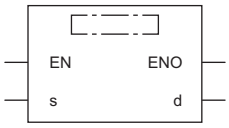
27.9 Converting WORD to INT

WORD_TO_INT(_E)

These functions convert a value from WORD data type to INT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=WORD_TO_INT(s);
		[With EN/ENO]
		d:=WORD_TO_INT_E(EN,ENO,s);

Setting data

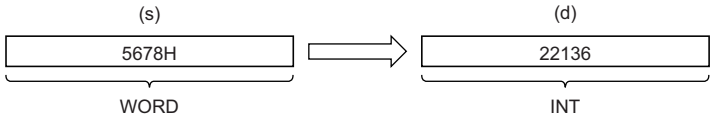
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	WORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert the value input to (s) from WORD data type to INT data type, and output the converted value from (d).



- Input a WORD data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

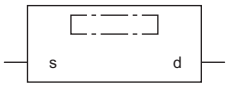
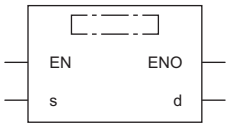
There is no operation error.

27.10 Converting WORD to DINT

WORD_TO_DINT(_E)

These functions convert a value from WORD data type to DINT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=WORD_TO_DINT(s);
		[With EN/ENO]
		d:=WORD_TO_DINT_E(EN,ENO,s);

Setting data

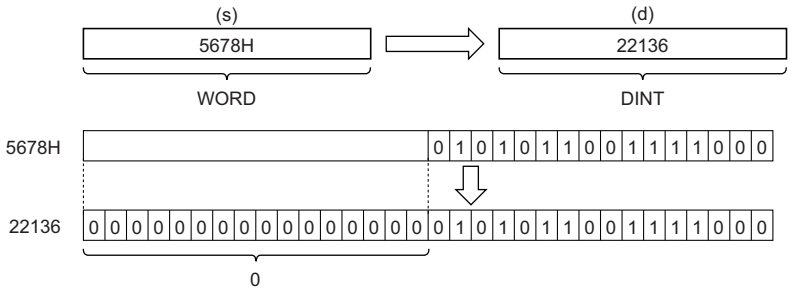
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	WORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert the value input to (s) from WORD data type to DINT data type, and output the converted value from (d).
- After the data type is converted, the upper 16 bits are filled with 0s.



- Input a WORD data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

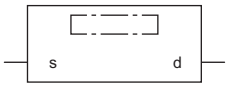
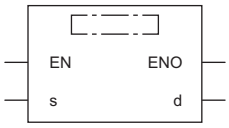
27.11 Converting WORD to TIME

WORD_TO_TIME(_E)

These functions convert a value from WORD data type to TIME data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=WORD_TO_TIME(s);
		[With EN/ENO]
		d:=WORD_TO_TIME_E(EN,ENO,s);

Setting data

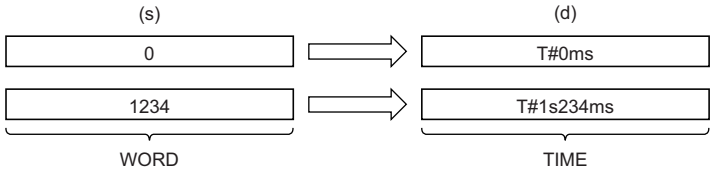
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	WORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

Processing details

■Operation processing

- These functions convert the value input to (s) from WORD data type to TIME data type, and output the converted value from (d).



- Input a WORD data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

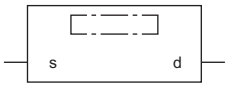
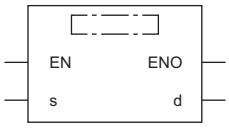
27.12 Converting WORD to STRING

WORD_TO_STRING(_E)

These functions convert a value from WORD data type to STRING data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=WORD_TO_STRING(s);
		[With EN/ENO]
		d:=WORD_TO_STRING_E(EN,ENO,s);

Setting data

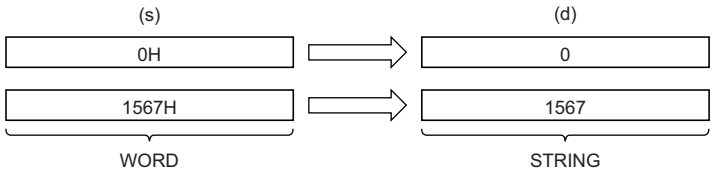
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	WORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	STRING(4)

Processing details

■Operation processing

- These functions convert the value input to (s) from WORD data type to STRING data type, and output the converted value from (d).



- Input a WORD data type value to (s).
- When SM701 (Number of output characters selection) is off, 00H is stored at the end of the string.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

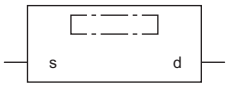
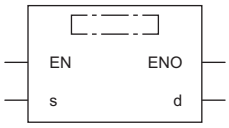
27.13 Converting DWORD to BOOL

DWORD_TO_BOOL(_E)

These functions convert a value from DWORD data type to BOOL data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=DWORD_TO_BOOL(s);
		[With EN/ENO]
		d:=DWORD_TO_BOOL_E(EN,ENO,s);

Setting data

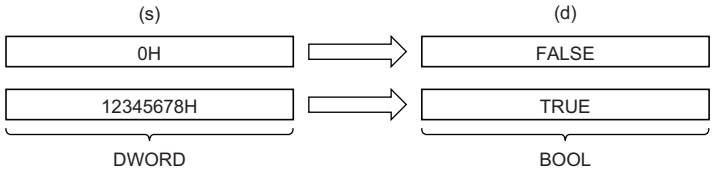
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DWORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	BOOL

Processing details

■Operation processing

- These functions convert the value input to (s) from DWORD data type to BOOL data type, and output the converted value from (d).
- When the input value is 0H, FALSE is output.
- When the input value is other than 0H, TRUE is output.



- Input a DWORD data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

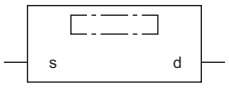
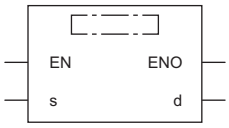
27.14 Converting DWORD to WORD

DWORD_TO_WORD(_E)

These functions convert a value from DWORD data type to WORD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=DWORD_TO_WORD(s);
		[With EN/ENO]
		d:=DWORD_TO_WORD_E(EN,ENO,s);

Setting data

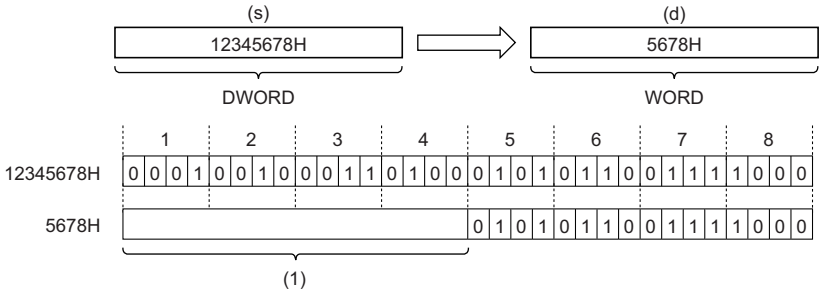
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DWORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	WORD

Processing details

■Operation processing

- These functions convert the value input to (s) from DWORD data type to WORD data type, and output the converted value from (d).
- The upper 16-bit data of the input value (DWORD data type) are discarded. (Refer to (1) in the figure below.)



- Input a DWORD data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Point

When the DWORD_TO_WORD(_E) function is executed, the upper 16-bit data of the input value (DWORD data type) are discarded.

Operation error

There is no operation error.

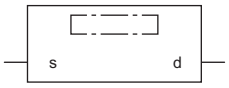
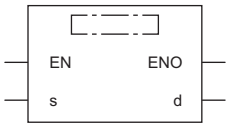
27.15 Converting DWORD to INT

DWORD_TO_INT(_E)

These functions convert a value from DWORD data type to INT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=DWORD_TO_INT(s);
		[With EN/ENO]
		d:=DWORD_TO_INT_E(EN,ENO,s);

Setting data

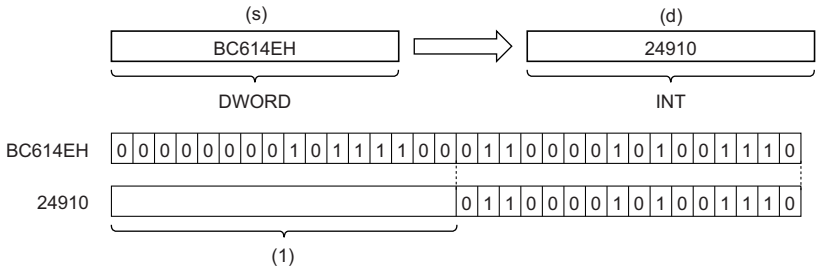
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DWORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert the value input to (s) from DWORD data type to INT data type, and output the converted value from (d).
- The upper 16-bit data of the input value (DWORD data type) are discarded. (Refer to (1) in the figure below.)



- Input a DWORD data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Point

When the DWORD_TO_INT(_E) function is executed, the upper 16-bit data of the input value (DWORD data type) are discarded.

Operation error

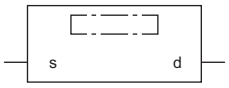
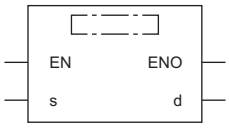
There is no operation error.

27.16 Converting DWORD to DINT

DWORD_TO_DINT(_E)

These functions convert a value from DWORD data type to DINT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DWORD_TO_DINT(s);
		[With EN/ENO]
		d:=DWORD_TO_DINT_E(EN,ENO,s);

Setting data

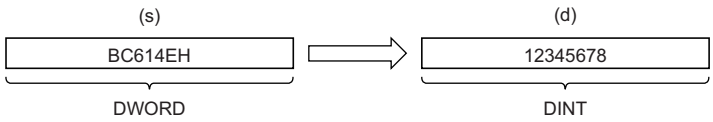
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DWORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert the value input to (s) from DWORD data type to DINT data type, and output the converted value from (d).



- Input a DWORD data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

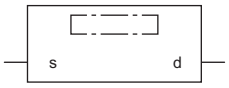
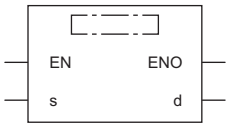
27.17 Converting DWORD to TIME

DWORD_TO_TIME(_E)

These functions convert a value from DWORD data type to TIME data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=DWORD_TO_TIME(s);
		[With EN/ENO]
		d:=DWORD_TO_TIME_E(EN,ENO,s);

Setting data

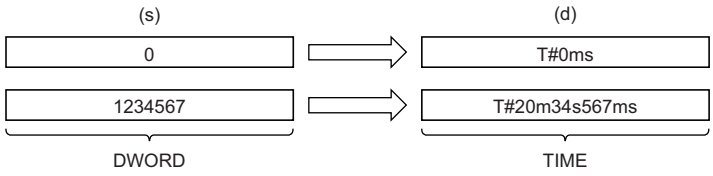
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DWORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

Processing details

■Operation processing

- These functions convert the value input to (s) from DWORD data type to TIME data type, and output the converted value from (d).



- Input a DWORD data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

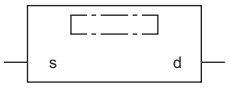
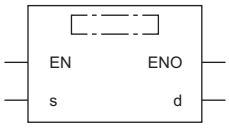
27.18 Converting DWORD to STRING

DWORD_TO_STRING(_E)

These functions convert a value from DWORD data type to STRING data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=DWORD_TO_STRING(s);
		[With EN/ENO]
		d:=DWORD_TO_STRING_E(EN,ENO,s);

Setting data

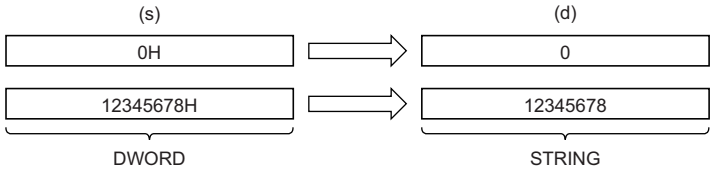
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DWORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	STRING(8)

Processing details

■Operation processing

- These functions convert the value input to (s) from DWORD data type to STRING data type, and output the converted value from (d).



- Input a DWORD data type value to (s).
- When SM701 (Number of output characters selection) is off, 00H is stored at the end of the string.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

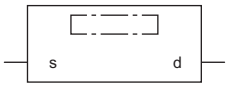
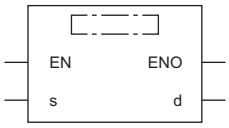
27.19 Converting INT to BOOL

INT_TO_BOOL(_E)

These functions convert a value from INT data type to BOOL data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=INT_TO_BOOL(s);
		[With EN/ENO]
		d:=INT_TO_BOOL_E(EN,ENO,s);

Setting data

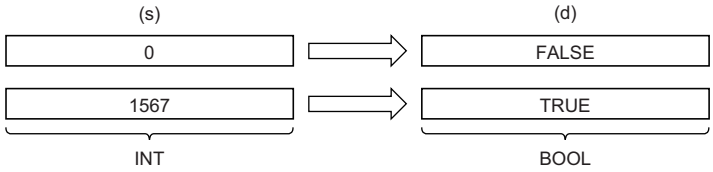
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	BOOL

Processing details

■Operation processing

- These functions convert the value input to (s) from INT data type to BOOL data type, and output the converted value from (d).
- When the value 0 is input, FALSE is output.
- When the value other than 0 is input, TRUE is output.



- Input an INT data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

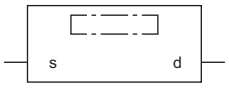
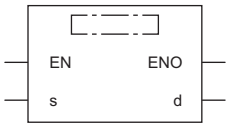
27.20 Converting INT to WORD

INT_TO_WORD(_E)

These functions convert a value from INT data type to WORD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=INT_TO_WORD(s);
		[With EN/ENO]
		d:=INT_TO_WORD_E(EN,ENO,s);

Setting data

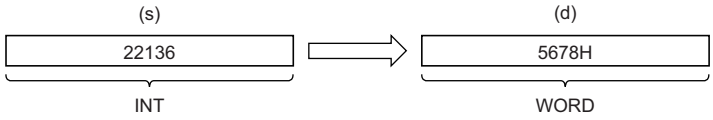
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	WORD

Processing details

■Operation processing

- These functions convert the value input to (s) from INT data type to WORD data type, and output the converted value from (d).



- Input an INT data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

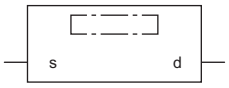
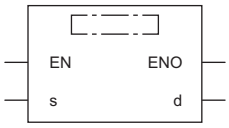
27.21 Converting INT to DWORD

INT_TO_DWORD(_E)

These functions convert a value from INT data type to DWORD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=INT_TO_DWORD(s);
		[With EN/ENO]
		d:=INT_TO_DWORD_E(EN,ENO,s);

Setting data

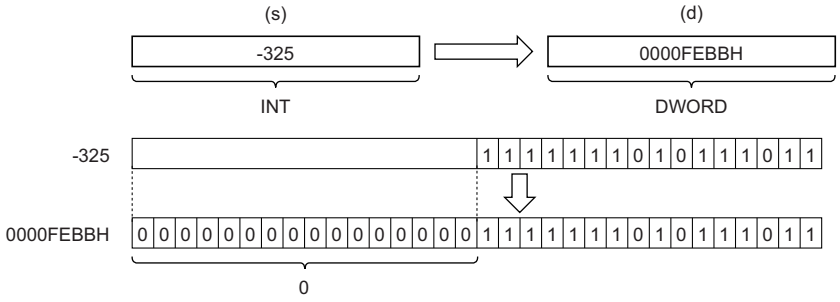
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DWORD

Processing details

■Operation processing

- These functions convert the value input to (s) from INT data type to DWORD data type, and output the converted value from (d).
- After the data type is converted, the upper 16 bits are filled with 0s.



- Input an INT data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

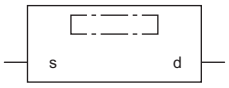
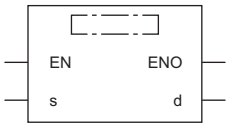
27.22 Converting INT to DINT

INT_TO_DINT(_E)

These functions convert a value from INT data type to DINT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=INT_TO_DINT(s);
		[With EN/ENO]
		d:=INT_TO_DINT_E(EN,ENO,s);

Setting data

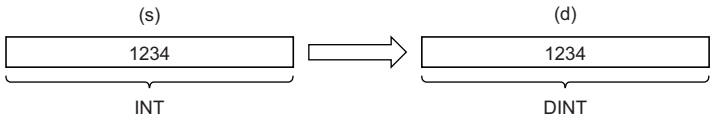
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert the value input to (s) from INT data type to DINT data type, and output the converted value from (d).



- Input an INT data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

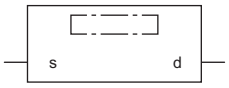
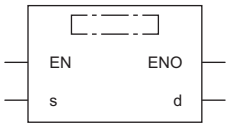
27.23 Converting INT to BCD

INT_TO_BCD(_E)

These functions convert a value from INT data type to BCD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=INT_TO_BCD(s);
		[With EN/ENO]
		d:=INT_TO_BCD_E(EN,ENO,s);

Setting data

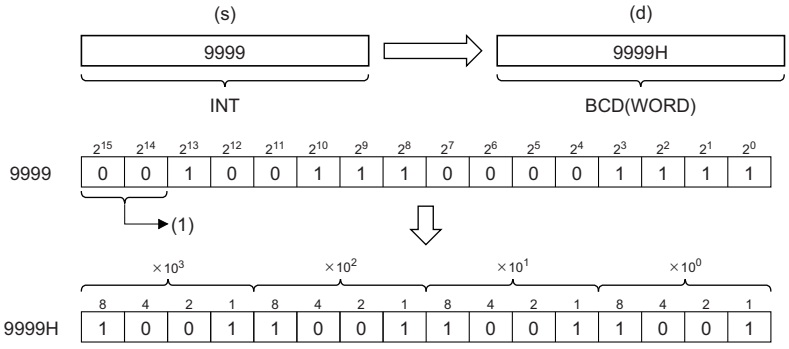
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	WORD

Processing details

■Operation processing

- These functions convert the value input to (s) from INT data type to BCD (WORD) data type, and output the converted value from (d).



(1) Set 0s.

- Input an INT data type value to (s) within the range of 0 to 9999.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3281H	Data input to (s) is out of the range, 0 to 9999.

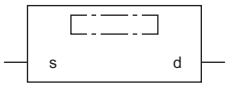
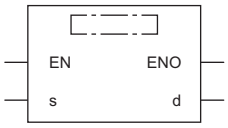
27.24 Converting INT to REAL

INT_TO_REAL(_E)

These functions convert a value from INT data type to REAL data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=INT_TO_REAL(s);
		[With EN/ENO]
		d:=INT_TO_REAL_E(EN,ENO,s);

Setting data

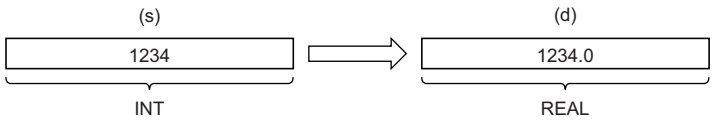
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	REAL

Processing details

■Operation processing

- These functions convert the value input to (s) from INT data type to REAL data type, and output the converted value from (d).



- Input an INT data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

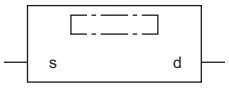
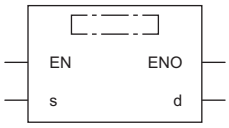
27.25 Converting INT to LREAL

INT_TO_LREAL(_E)

These functions convert a value from INT data type to LREAL data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=INT_TO_LREAL(s);
		[With EN/ENO]
		d:=INT_TO_LREAL_E(EN,ENO,s);

Setting data

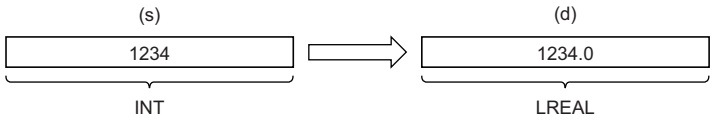
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	LREAL

Processing details

■Operation processing

- These functions convert the value input to (s) from INT data type to LREAL data type, and output the converted value from (d).



- Input an INT data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

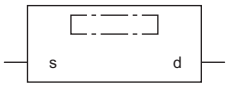
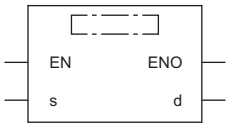
27.26 Converting INT to TIME

INT_TO_TIME(_E)

These functions convert a value from INT data type to TIME data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=INT_TO_TIME(s);
		[With EN/ENO]
		d:=INT_TO_TIME_E(EN,ENO,s);

Setting data

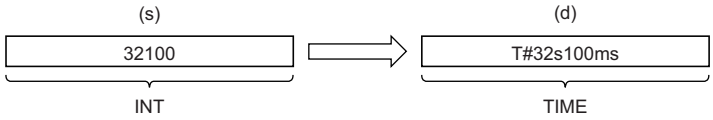
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

Processing details

■Operation processing

- These functions convert the value input to (s) from INT data type to TIME data type, and output the converted value from (d).



- Input an INT data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

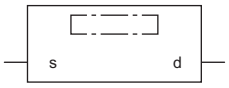
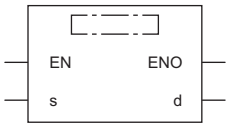
27.27 Converting INT to STRING

INT_TO_STRING(_E)

These functions convert a value from INT data type to STRING data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=INT_TO_STRING(s);
		[With EN/ENO]
		d:=INT_TO_STRING_E(EN,ENO,s);

Setting data

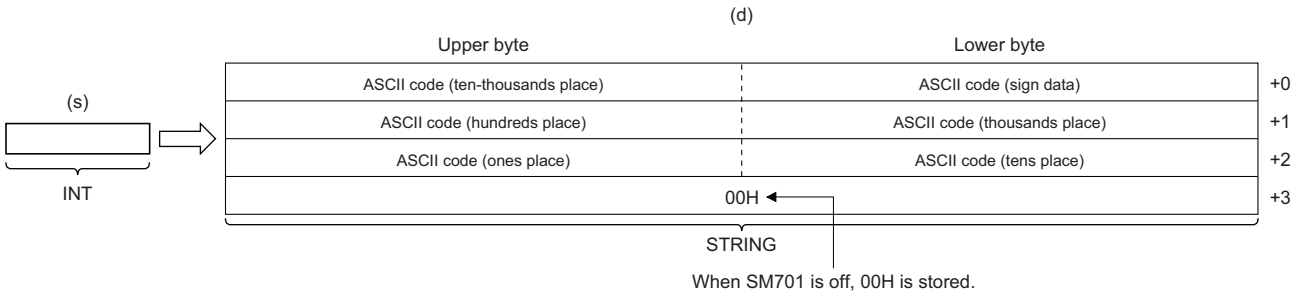
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	STRING(6)

Processing details

■Operation processing

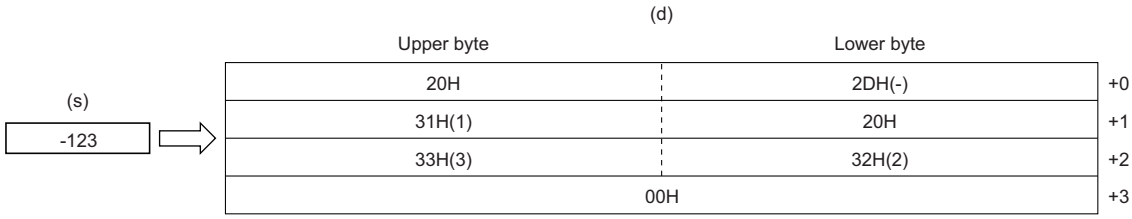
- These functions convert the value input to (s) from INT data type to STRING data type, and output the converted value from (d).



- Input an INT data type value to (s).
- As sign data, 20H (space) is stored if the input value is positive, and 2DH (-) is stored if the value is negative.
- If the number of digits in the input value is less than the number of significant digits, 20H (space) is stored for the upper digit(s).

Ex.

When the value -123 is input



- When SM701 (Number of output characters selection) is off, 00H is stored at the end of the string (4th word).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

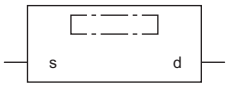
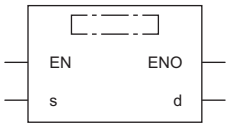
27.28 Converting DINT to BOOL

DINT_TO_BOOL(_E)

These functions convert a value from DINT data type to BOOL data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=DINT_TO_BOOL(s);
		[With EN/ENO]
		d:=DINT_TO_BOOL_E(EN,ENO,s);

Setting data

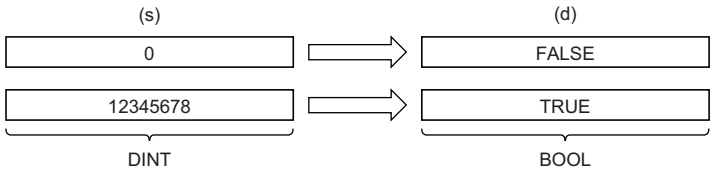
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	BOOL

Processing details

■Operation processing

- These functions convert the value input to (s) from DINT data type to BOOL data type, and output the converted value from (d).
- When the value 0 is input, FALSE is output.
- When the value other than 0 is input, TRUE is output.



- Input a DINT data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

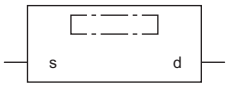
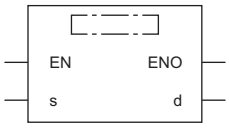
There is no operation error.

27.29 Converting DINT to WORD

DINT_TO_WORD(_E)

These functions convert a value from DINT data type to WORD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=DINT_TO_WORD(s);
		[With EN/ENO]
		d:=DINT_TO_WORD_E(EN,ENO,s);

Setting data

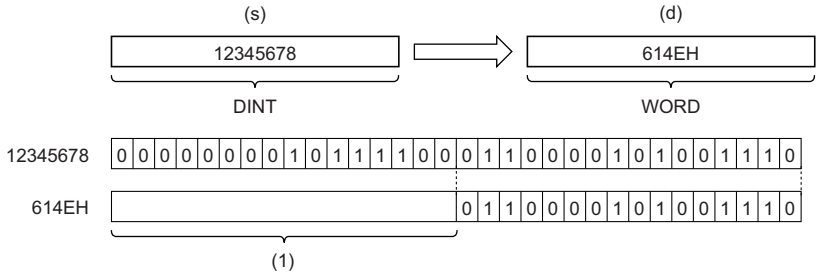
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	WORD

Processing details

■Operation processing

- These functions convert the value input to (s) from DINT data type to WORD data type, and output the converted value from (d).
- The upper 16-bit data of the input value (DINT data type) are discarded. (Refer to (1) in the figure below.)



- Input a DINT data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Point

When the DINT_TO_WORD(_E) function is executed, the upper 16-bit data of the input value (DINT data type) are discarded.

Operation error

There is no operation error.

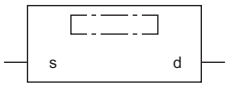
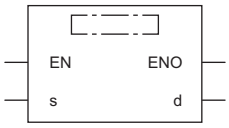
27.30 Converting DINT to DWORD

DINT_TO_DWORD(_E)

These functions convert a value from DINT data type to DWORD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=DINT_TO_DWORD(s);
		[With EN/ENO]
		d:=DINT_TO_DWORD_E(EN,ENO,s);

Setting data

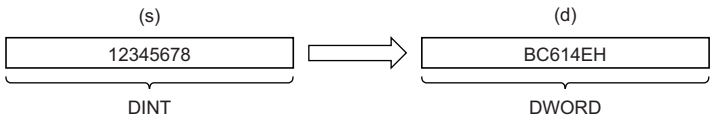
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DWORD

Processing details

■Operation processing

- These functions convert the value input to (s) from DINT data type to DWORD data type, and output the converted value from (d).



- Input a DINT data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

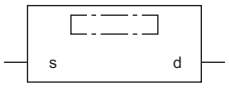
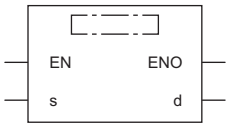
27.31 Converting DINT to INT

DINT_TO_INT(_E)

These functions convert a value from DINT data type to INT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=DINT_TO_INT(s);
		[With EN/ENO]
		d:=DINT_TO_INT_E(EN,ENO,s);

Setting data

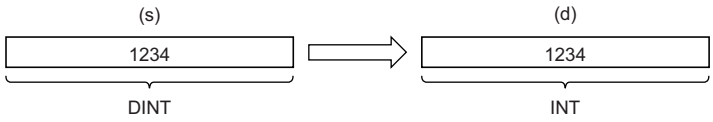
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert the value input to (s) from DINT data type to INT data type, and output the converted value from (d).



- Input a DINT data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3281H	The 32-bit signed binary data input to (s) is out of the range, -32768 to 32767.

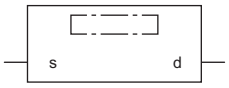
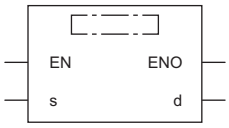
27.32 Converting DINT to BCD

DINT_TO_BCD(_E)

These functions convert a value from DINT data type to BCD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=DINT_TO_BCD(s);
		[With EN/ENO]
		d:=DINT_TO_BCD_E(EN,ENO,s);

Setting data

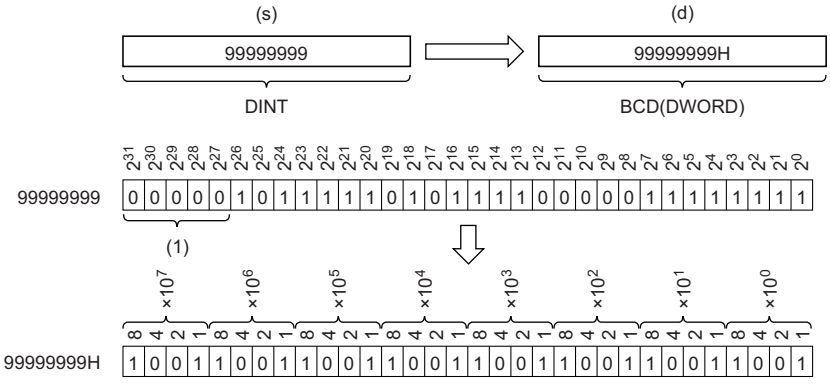
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

Processing details

■Operation processing

- These functions convert the value input to (s) from DINT data type to BCD (DWORD) data type, and output the converted value from (d).



(1) Set 0s.

- Input a DINT data type value to (s). When (d) is of WORD date type, the input value range is 0 to 9999. When (d) is of DWORD date type, the input value range is 0 to 99999999.
- WORD or DWORD data type can be specified for (d). BOOL data type cannot be specified.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

• When (d) is of WORD data type

Error code (SD0)	Description
3281H	The 32-bit signed binary data input to (s) is out of the range, -32768 to 32767.
	Data input to (s) is out of the range, 0 to 9999.

• When (d) is of DWORD data type

Error code (SD0)	Description
3281H	Data in the device specified by (s) is out of the range, 0 to 99999999.

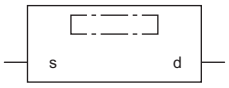
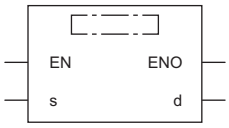
27.33 Converting DINT to REAL

DINT_TO_REAL(_E)

These functions convert a value from DINT data type to REAL data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=DINT_TO_REAL(s);
		[With EN/ENO]
		d:=DINT_TO_REAL_E(EN,ENO,s);

Setting data

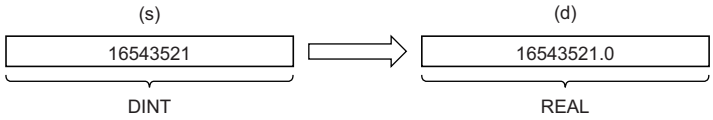
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	REAL

Processing details

■Operation processing

- These functions convert the value input to (s) from DINT data type to REAL data type, and output the converted value from (d).



- Input a DINT data type value to (s).
- The number of significant digits is about seven because a REAL data type value is processed in 32-bit single precision.
- If the integer value exceeds the range of -16777216 to 16777215, a rounding error occurs in the converted value.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

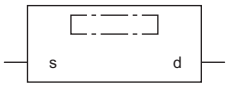
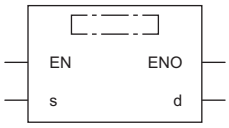
27.34 Converting DINT to LREAL

DINT_TO_LREAL(_E)

These functions convert a value from DINT data type to LREAL data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=DINT_TO_LREAL(s);
		[With EN/ENO]
		d:=DINT_TO_LREAL_E(EN,ENO,s);

Setting data

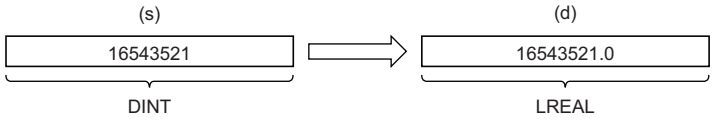
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	LREAL

Processing details

■Operation processing

- These functions convert the value input to (s) from DINT data type to LREAL data type, and output the converted value from (d).



- Input a DINT data type value to (s).
- The number of significant digits is about 15 because an LREAL data type value is processed in 64-bit double precision.
- If the integer value exceeds the range of -2147483648 to 2147483647, a rounding error occurs in the converted value.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

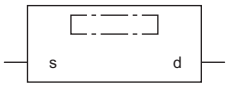
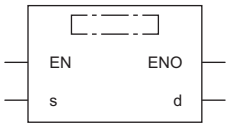
27.35 Converting DINT to TIME

DINT_TO_TIME(_E)

These functions convert a value from DINT data type to TIME data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=DINT_TO_TIME(s);
		[With EN/ENO]
		d:=DINT_TO_TIME_E(EN,ENO,s);

Setting data

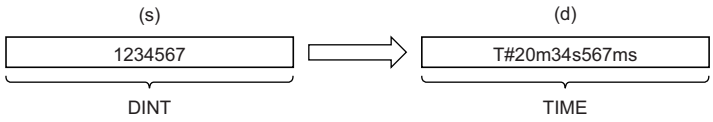
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

Processing details

■Operation processing

- These functions convert the value input to (s) from DINT data type to TIME data type, and output the converted value from (d).



- Input a DINT data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

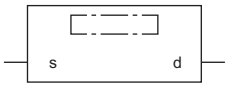
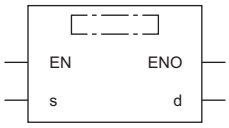
There is no operation error.

27.36 Converting DINT to STRING

DINT_TO_STRING(_E)

These functions convert a value from DINT data type to STRING data type.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DINT_TO_STRING(s);
		[With EN/ENO]
		d:=DINT_TO_STRING_E(EN,ENO,s);

Setting data

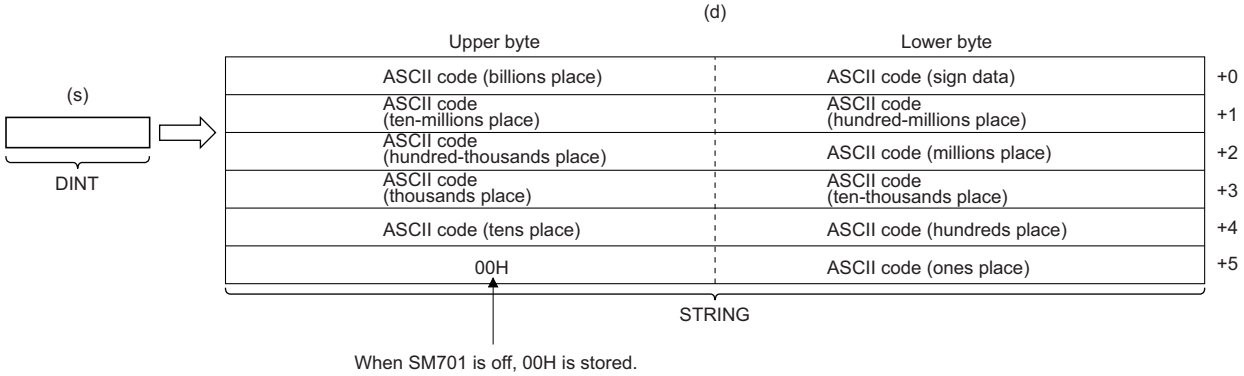
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	DINT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	STRING(11)

Processing details

■Operation processing

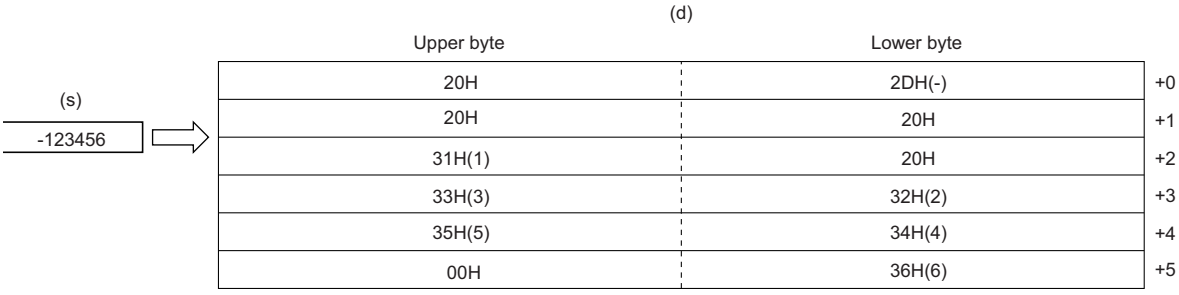
- These functions convert the value input to (s) from DINT data type to STRING data type, and output the converted value from (d).



- Input a DINT data type value to (s).
- As sign data, 20H (space) is stored if the input value is positive, and 2DH (-) is stored if the value is negative.
- If the number of digits in the input value is less than the number of significant digits, 20H (space) is stored for the upper digit(s).

Ex.

When the value -123456 is input



- When SM701 (Number of output characters selection) is off, 00H is stored at the end of the string (upper bytes of the 6th word).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

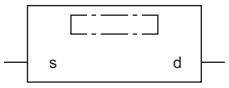
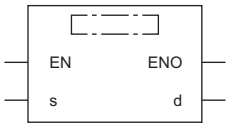
27.37 Converting BCD to INT

BCD_TO_INT(_E)

These functions convert a value from BCD data type to INT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=BCD_TO_INT(s);
		[With EN/ENO]
		d:=BCD_TO_INT_E(EN,ENO,s);

Setting data

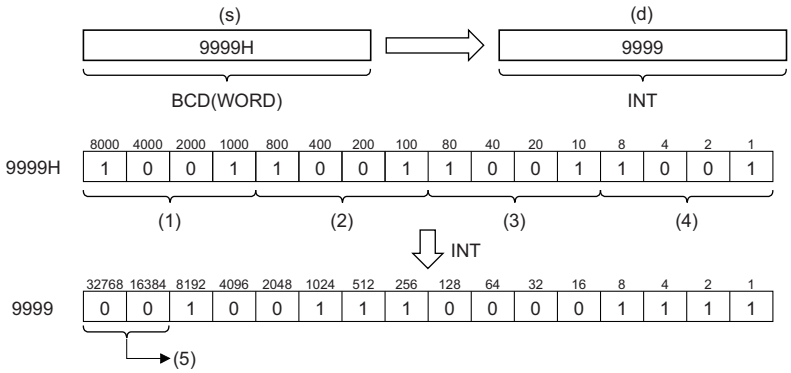
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	WORD
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert the value input to (s) from BCD (WORD) data type to INT data type, and output the converted value from (d).



(1) Thousands place
(2) Hundreds place
(3) Tens place
(4) Ones place
(5) Filled with 0s.

- Input a WORD data type value to (s) within the range of 0H to 9999H (range of each digit: 0 to 9).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3281H	A value other than 0 to 9 exists at any digit of the value input to (s).

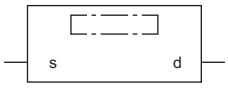
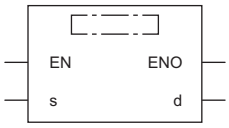
- Turning on SM754 can prevent the above error from being issued. If the specified value is out of the valid range, the BCD_TO_INT(_E) function is not executed regardless of the status (on/off) of SM754.

27.38 Converting BCD to DINT

BCD_TO_DINT(_E)

These functions convert a value from BCD data type to DINT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=BCD_TO_DINT(s);
		[With EN/ENO]
		d:=BCD_TO_DINT_E(EN,ENO,s);

Setting data

■Description, type, data type

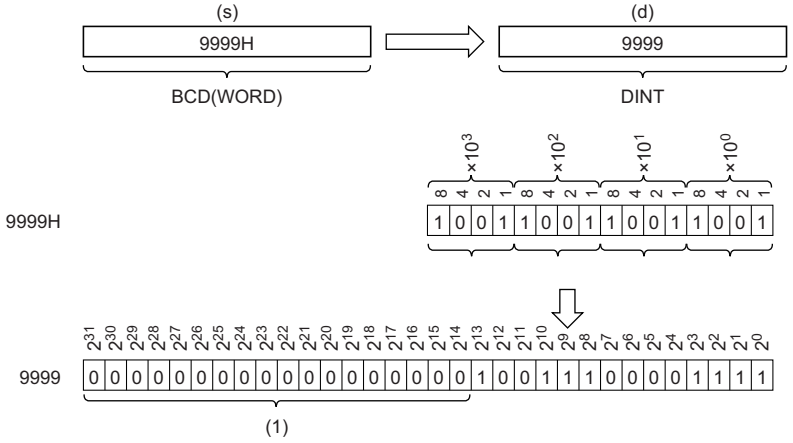
Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

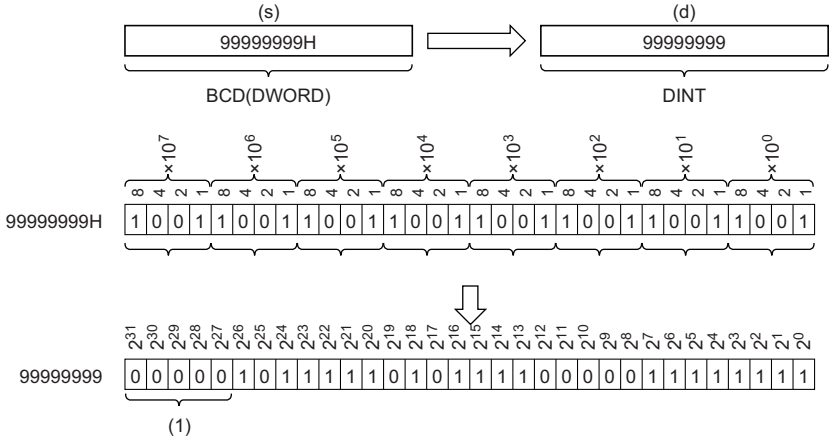
- These functions convert the value input to (s) from BCD (WORD or DWORD) data type to DINT data type, and output the converted value from (d).

• When (s) is of WORD data type



(1) Filled with 0s.

• When (s) is of DWORD data type



(1) Filled with 0s.

- Input a WORD data type value within the range of 0H to 9999H (range of each digit: 0 to 9) or a DWORD date type value within the range of 0H to 99999999H (range of each digit: 0 to 9) to (s).
- WORD or DWORD data type can be specified for (s). BOOL data type cannot be specified.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3281H	A value other than 0 to 9 exists at any digit of the value input to (s).

- Turning on SM754 can prevent the above error from being issued. If the specified value is out of the valid range, the BCD_TO_DINT(_E) function is not executed regardless of the status (on/off) of SM754.

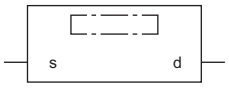
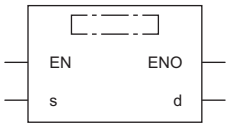
27.39 Converting BCD to STRING

BCD_TO_STRING(_E)

These functions convert a value from BCD data type to STRING data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=BCD_TO_STRING(s);
		[With EN/ENO]
		d:=BCD_TO_STRING_E(EN,ENO,s);

Setting data

■Description, type, data type

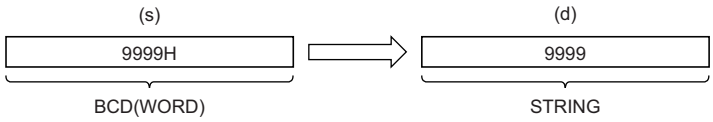
Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	STRING(8)

Processing details

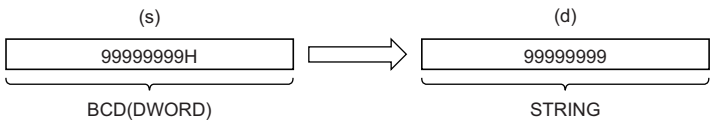
■Operation processing

- These functions convert the value input to (s) from BCD (WORD or DWORD) data type to STRING data type, and output the converted value from (d).

• When (s) is of WORD data type



• When (s) is of DWORD data type



- WORD or DWORD data type can be specified for (s). BOOL data type cannot be specified.
- When SM701 (Number of output characters selection) is off, 00H is stored at the end of the string.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s) is of WORD data type

Error code (SD0)	Description
3281H	Data input to (s) is out of the range, 0 to 9999.

- When (s) is of DWORD data type

Error code (SD0)	Description
3281H	Data in the device specified by (s) is out of the range, 0 to 99999999.

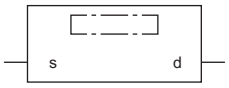
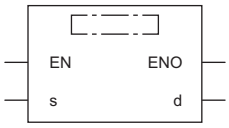
27.40 Converting REAL to INT

REAL_TO_INT(_E)

These functions convert a value from REAL data type to INT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=REAL_TO_INT(s);
		[With EN/ENO]
		d:=REAL_TO_INT_E(EN,ENO,s);

Setting data

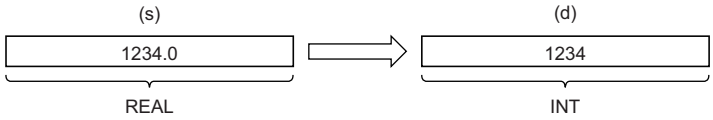
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert the value input to (s) from REAL data type to INT data type, and output the converted value from (d).



- Input a REAL data type value to (s) within the range of -32768 to 32767.
- After conversion, the first digit after the decimal point of the input value (REAL data type) is rounded off.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3281H	The single-precision real number input to (s) is out of the range, -32768 to 32767.
3282H	• An unusual number is set to (s). • The single-precision real number input to (s) is not within the following range: $-2^{128}<(s)\leq-2^{-126}$, 0, $2^{-126}\leq(s)<2^{128}$ (E-3.40282347+38 to E-1.17549435-38, 0, E1.17549435-38 to E3.40282347+38) • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

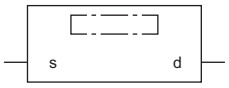
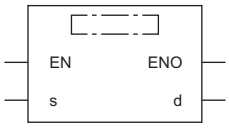
27.41 Converting REAL to DINT

REAL_TO_DINT(_E)

These functions convert a value from REAL data type to DINT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=REAL_TO_DINT(s);
		[With EN/ENO]
		d:=REAL_TO_DINT_E(EN,ENO,s);

Setting data

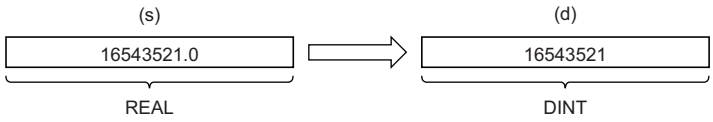
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert the value input to (s) from REAL data type to DINT data type, and output the converted value from (d).



- Input a REAL data type value to (s) within the range of -2147483648 to 2147483647.
- After conversion, the first digit after the decimal point of the input value (REAL data type) is rounded off.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3281H	The single-precision real number input to (s) is out of the range, -2147483648 to 2147483647.
3282H	• An unusual number is set to (s). • The single-precision real number input to (s) is not within the following range: $-2^{128}<(s)\leq-2^{-126}$, 0, $2^{-126}\leq(s)<2^{128}$ (E-3.40282347+38 to E-1.17549435-38, 0, E1.17549435-38 to E3.40282347+38) • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

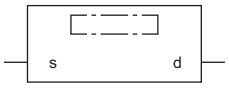
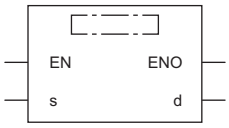
27.42 Converting REAL to LREAL

REAL_TO_LREAL(_E)

These functions convert a value from REAL data type to LREAL data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=REAL_TO_LREAL(s);
		[With EN/ENO]
		d:=REAL_TO_LREAL_E(EN,ENO,s);

Setting data

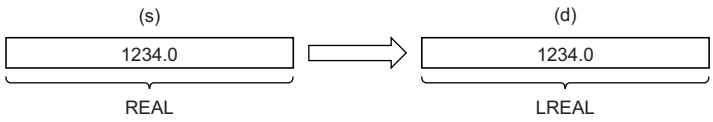
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	LREAL

Processing details

■Operation processing

- These functions convert the value input to (s) from REAL data type to LREAL data type, and output the converted value from (d).



- Input a REAL data type value to (s).
- The number of significant digits is about seven because a REAL data type value is processed in 32-bit single precision.
- If the integer value exceeds the range of -16777216 to 16777215, a rounding error occurs in the converted value.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3282H	The data input to (s) is -0 or out of the following range. $-2^{128}<(s)\leq-2^{-126}$, 0, $2^{-126}\leq(s)<2^{128}$ (E-3.40282347+38 to E-1.17549435-38, 0, E1.17549435-38 to E3.40282347+38)

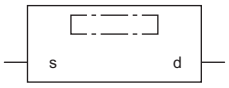
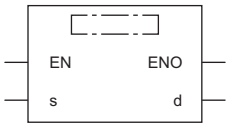
27.43 Converting REAL to STRING

REAL_TO_STRING(_E)

These functions converts a REAL data type value to STRING data type (exponential form).

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=REAL_TO_STRING(s);
		[With EN/ENO]
		d:=REAL_TO_STRING_E(EN,ENO,s);

Setting data

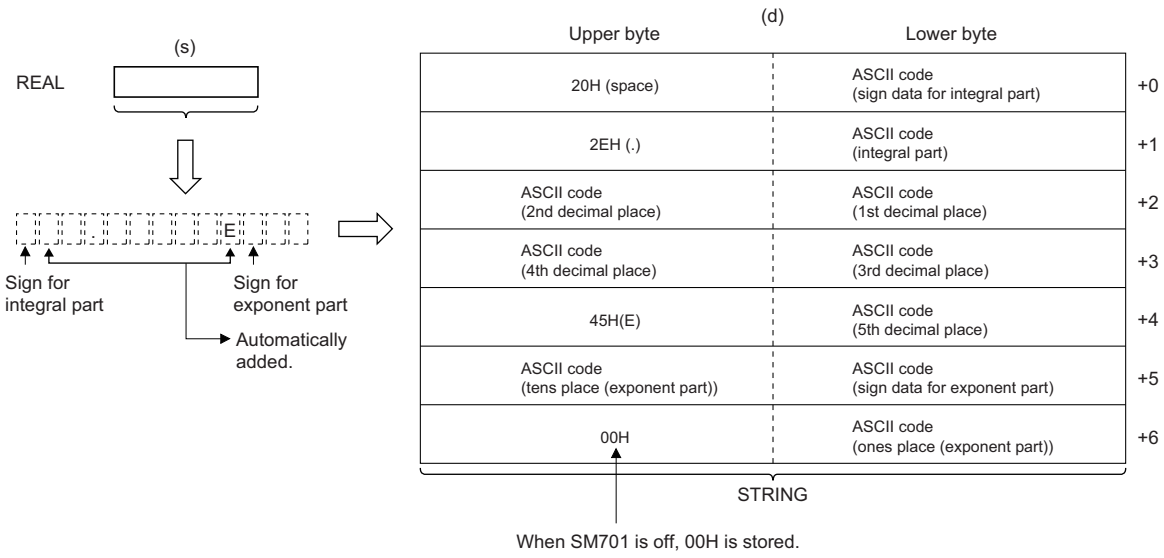
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	STRING(13)

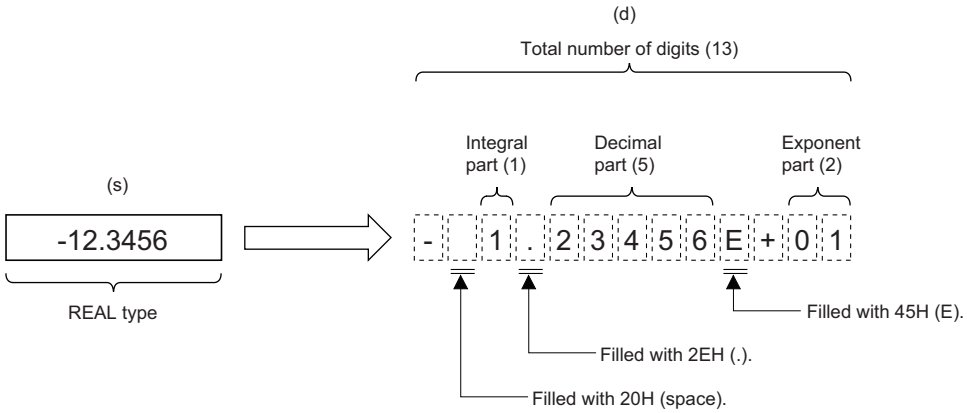
Processing details

■Operation processing

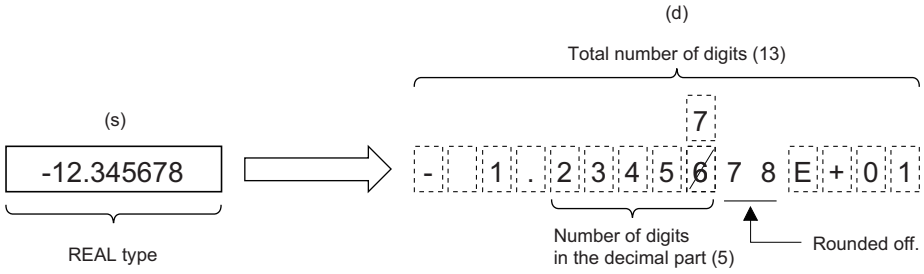
- These functions convert the value input to (s) from REAL data type to STRING data type (exponential form), and output the converted value from (d).



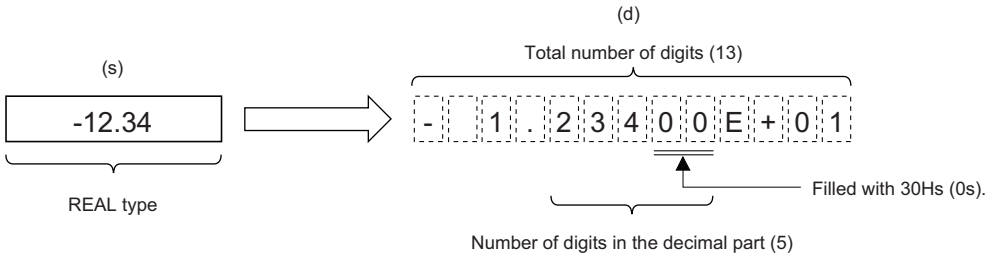
- Input a REAL data type value to (s).
- The converted string data is output from (d) as follows.
  - The number of digits for an integral part, decimal part, and exponent is fixed, integral part: one digit; decimal part: five digits; exponent: two digits.
  - As the second byte, 20H (space) is stored; as the fourth byte, 2EH (.) is stored; and as the 10th byte, 45H (E) is stored automatically.



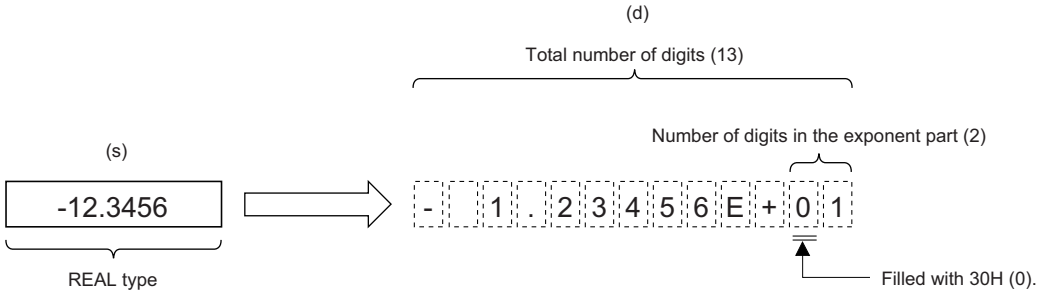
- As sign data (for integral part), 20H (space) is stored if the input value is positive, and 2DH (-) is stored if the input value is negative.
- The sixth and later digits of the decimal part are rounded off.



- If the number of digits in the input value is less than the number of significant digits, 30H (0) is stored in the decimal part.



- As sign data (for exponent), 2BH (+) is stored if the input value is positive, and 2DH (-) is stored if the input value is negative.
- When the exponent is one digit, 30H (0) is stored in the tens place of the exponent.



- The NULL code (00H) is automatically stored at the end (i.e. seventh word) of the converted string.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3282H	• The value input to (s) is out of the following range: $-2^{128}<(s)\leq-2^{-126}$, 0, $2^{-126}\leq(s)<2^{128}$ (E-3.40282347+38 to E-1.17549435-38, 0, E1.17549435-38 to E3.40282347+38) • The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3286H	The entire string after conversion cannot be stored in the label or device area (between the specified device number and the last device number) specified by (d). (The number of required points is insufficient.)

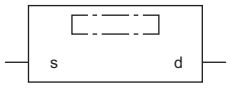
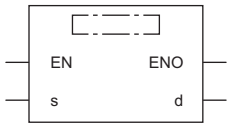
27.44 Converting LREAL to INT

LREAL_TO_INT(_E)

These functions convert a value from LREAL data type to INT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=LREAL_TO_INT(s);
		[With EN/ENO]
		d:=LREAL_TO_INT_E(EN,ENO,s);

Setting data

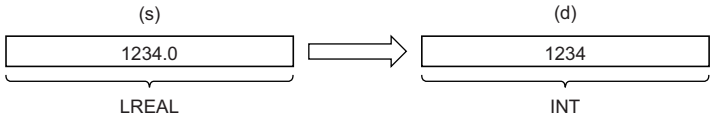
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	LREAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert the value input to (s) from LREAL data type to INT data type, and output the converted value from (d).



- Input an LREAL data type value to (s).
- After conversion, the first digit after the decimal point of the input value (LREAL data type) is rounded off.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

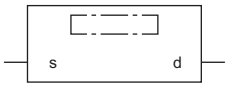
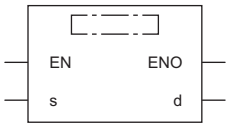
Error code (SD0)	Description
3281H	The data input to (s) is other than -32768 to 32767.
3282H	• The data input to (s) is -0 or out of the following range. $-2^{1024}<(s), (d)\leq -2^{-1022}$, 0, $2^{-1022}\leq(s), (d)<2^{1024}$ (E-1.7976931348623157+308 to E-2.2250738585072014-308, 0, E2.2250738585072014-308 to E1.7976931348623157+308) • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

27.45 Converting LREAL to DINT

LREAL_TO_DINT(_E)

These functions convert a value from LREAL data type to DINT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=LREAL_TO_DINT(s);
		[With EN/ENO]
		d:=LREAL_TO_DINT_E(EN,ENO,s);

Setting data

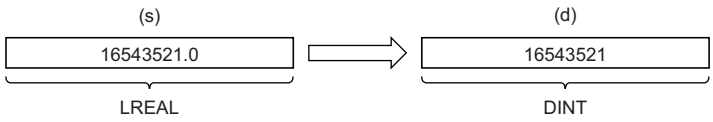
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	LREAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert the value input to (s) from LREAL data type to DINT data type, and output the converted value from (d).



- Input an LREAL data type value to (s).
- After conversion, the first digit after the decimal point of the input value (LREAL data type) is rounded off.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3281H	The data input to (s) is other than -2147483648 to 2147483647.
3282H	• The data input to (s) is -0 or out of the following range. -2^{1024}<(s), (d)≤-2^{-1022}, 0, 2^{-1022}≤(s), (d)<2^{1024} (E-1.7976931348623157+308 to E-2.2250738585072014-308, 0, E2.2250738585072014-308 to E1.7976931348623157+308) • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

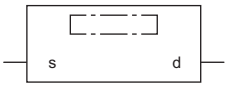
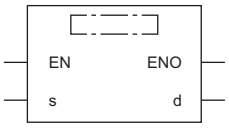
27.46 Converting LREAL to REAL

LREAL_TO_REAL(_E)

These functions convert a value from LREAL data type to REAL data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=LREAL_TO_REAL(s);
		[With EN/ENO]
		d:=LREAL_TO_REAL_E(EN,ENO,s);

Setting data

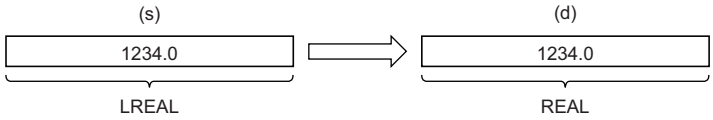
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	LREAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	REAL

Processing details

■Operation processing

- These functions convert the value input to (s) from LREAL data type to REAL data type, and output the converted value from (d).



- Input an LREAL data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3282H	The data input to (s) is -0 or out of the following range. $-2^{1024}<(s), (d)\leq-2^{-1022}$, 0, $2^{-1022}\leq(s), (d)<2^{1024}$ (E-1.7976931348623157+308 to E-2.2250738585072014-308, 0, E2.2250738585072014-308 to E1.7976931348623157+308)
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

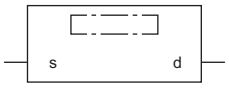
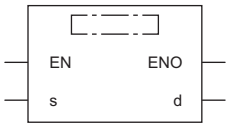
27.47 Converting TIME to BOOL

TIME_TO_BOOL(_E)

These functions convert a value from TIME data type to BOOL data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=TIME_TO_BOOL(s);
		[With EN/ENO]
		d:=TIME_TO_BOOL_E(EN,ENO,s);

Setting data

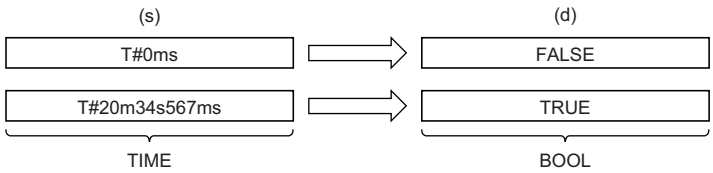
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	BOOL

Processing details

■Operation processing

- These functions convert the value input to (s) from TIME data type to BOOL data type, and output the converted value from (d).



■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

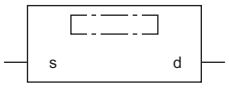
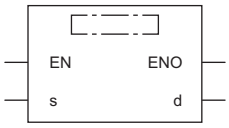
27.48 Converting TIME to WORD

TIME_TO_WORD(_E)

These functions convert a value from TIME data type to WORD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=TIME_TO_WORD(s);
		[With EN/ENO]
		d:=TIME_TO_WORD_E(EN,ENO,s);

Setting data

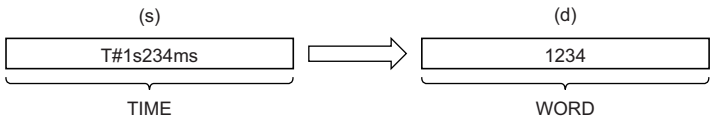
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	WORD

Processing details

■Operation processing

- These functions convert the value input to (s) from TIME data type to WORD data type, and output the converted value from (d).



- Input a TIME data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

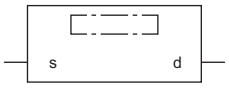
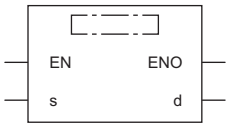
27.49 Converting TIME to DWORD

TIME_TO_DWORD(_E)

These functions convert a value from TIME data type to DWORD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=TIME_TO_DWORD(s);
		[With EN/ENO]
		d:=TIME_TO_DWORD_E(EN,ENO,s);

Setting data

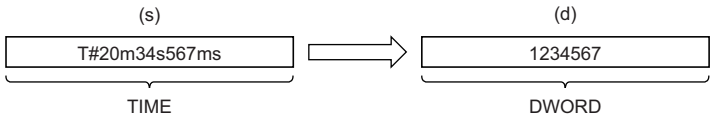
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DWORD

Processing details

■Operation processing

- These functions convert the value input to (s) from TIME data type to DWORD data type, and output the converted value from (d).



- Input a TIME data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

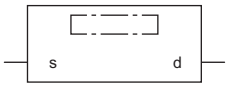
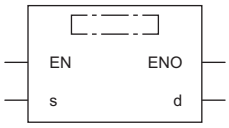
27.50 Converting TIME to INT

TIME_TO_INT(_E)

These functions convert a value from TIME data type to INT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=TIME_TO_INT(s);
		[With EN/ENO]
		d:=TIME_TO_INT_E(EN,ENO,s);

Setting data

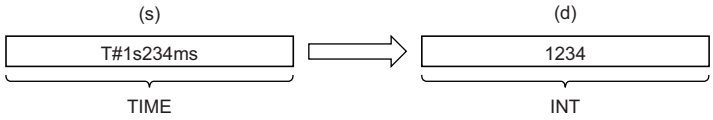
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert the value input to (s) from TIME data type to INT data type, and output the converted value from (d).



- Input a TIME data type value to (s).
- The upper 16-bit data of the input value (TIME data type) are discarded.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

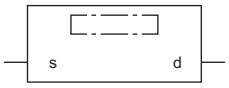
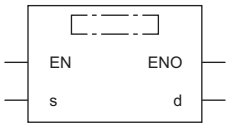
27.51 Converting TIME to DINT

TIME_TO_DINT(_E)

These functions convert a value from TIME data type to DINT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=TIME_TO_DINT(s);
		[With EN/ENO]
		d:=TIME_TO_DINT_E(EN,ENO,s);

Setting data

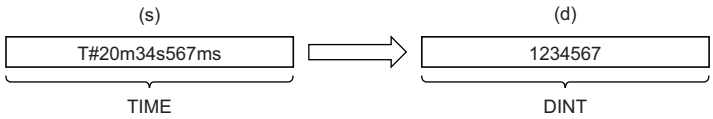
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert the value input to (s) from TIME data type to DINT data type, and output the converted value from (d).



- Input a TIME data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

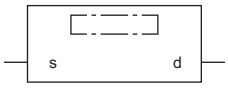
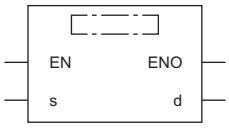
There is no operation error.

27.52 Converting TIME to STRING

TIME_TO_STRING(_E)

These functions convert a value from TIME data type to STRING data type.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=TIME_TO_STRING(s);
		[With EN/ENO]
		d:=TIME_TO_STRING_E(EN,ENO,s);

Setting data

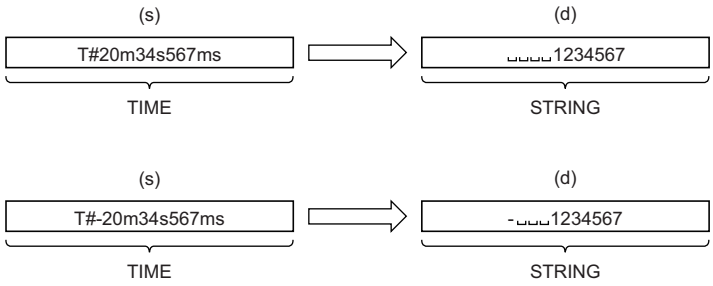
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	STRING STRING(11)

Processing details

■Operation processing

- These functions convert the value input to (s) from TIME data type to STRING data type, and output the converted value from (d).



- Input a TIME data type value to (s).
- When SM701 (Number of output characters selection) is off, 00H is stored at the end of the string.
- The operation result will be as follows.
  - As the first character, 20H (space) is stored if the output value is positive, and 2DH (-) is stored if the output value is negative.
  - At the left of the number of significant digits, 20H (space) is stored.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

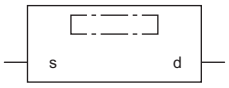
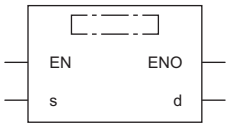
27.53 Converting STRING to BOOL

STRING_TO_BOOL(_E)

These functions convert a value from STRING data type to BOOL data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=STRING_TO_BOOL(s);
		[With EN/ENO]
		d:=STRING_TO_BOOL_E(EN,ENO,s);

Setting data

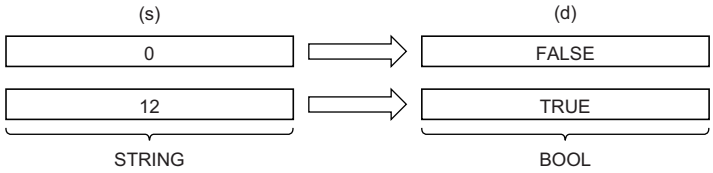
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(1)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	BOOL

Processing details

■Operation processing

- These functions convert the value input to (s) from STRING data type (decimal form/exponential form) to BOOL data type, and output the converted value from (d).



■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

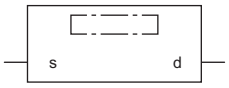
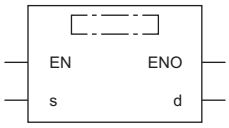
27.54 Converting STRING to WORD

STRING_TO_WORD(_E)

These functions convert a value from STRING data type to WORD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=STRING_TO_WORD(s);
		[With EN/ENO]
		d:=STRING_TO_WORD_E(EN,ENO,s);

Setting data

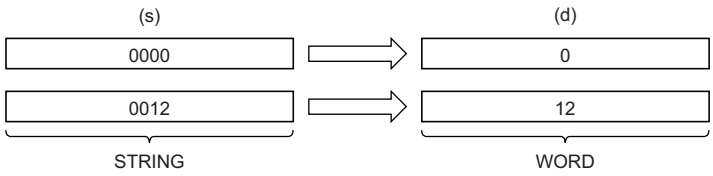
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(4)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	WORD

Processing details

■Operation processing

- These functions convert the value input to (s) from STRING data type to WORD data type, and output the converted value from (d).



- Input a STRING data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3281H	An ASCII code other than 30H to 39H and 41H to 46H is input.

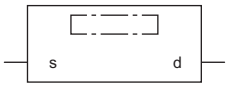
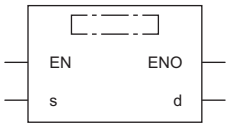
27.55 Converting STRING to DWORD

STRING_TO_DWORD(_E)

These functions convert a value from STRING data type to DWORD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=STRING_TO_DWORD(s);
		[With EN/ENO]
		d:=STRING_TO_DWORD_E(EN,ENO,s);

Setting data

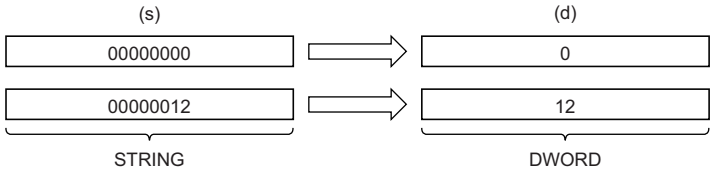
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(8)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DWORD

Processing details

■Operation processing

- These functions convert the value input to (s) from STRING data type to DWORD data type, and output the converted value from (d).



- Input a STRING data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3281H	An ASCII code other than 30H to 39H and 41H to 46H is input.

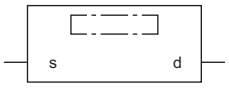
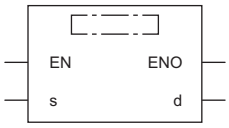
27.56 Converting STRING to INT

STRING_TO_INT(_E)

These functions convert a value from STRING data type to INT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=STRING_TO_INT(s);
		[With EN/ENO]
		d:=STRING_TO_INT_E(EN,ENO,s);

Setting data

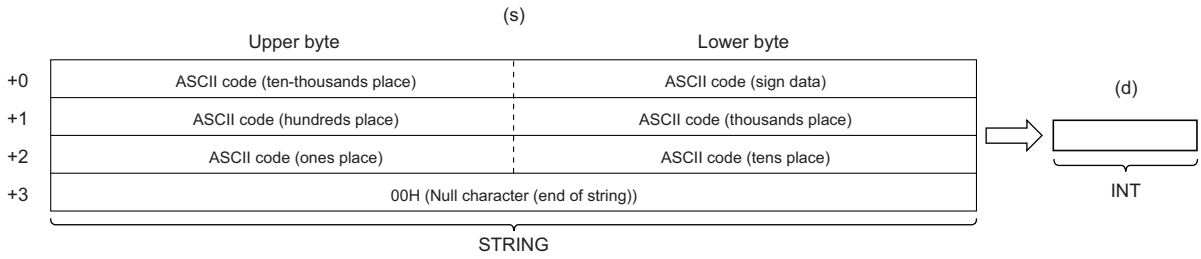
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(6)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions convert the value input to (s) from STRING data type to INT data type, and output the converted value from (d).



- Input a STRING data type value to (s) within the following range.
  - ASCII code: 30H to 39H, 20H, 2DH, and 00H
  - STRING data type value: -32768 to 32767

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3281H	Invalid data that cannot be converted are input in (s). • ASCII code: other than 30H to 39H, 20H, and 00H • STRING data type value: other than -32768 to 32767

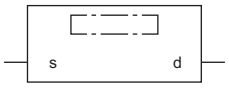
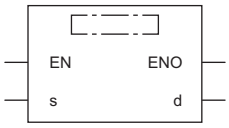
27.57 Converting STRING to DINT

STRING_TO_DINT(_E)

These functions convert a value from STRING data type to DINT data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=STRING_TO_DINT(s);
		[With EN/ENO]
		d:=STRING_TO_DINT_E(EN,ENO,s);

Setting data

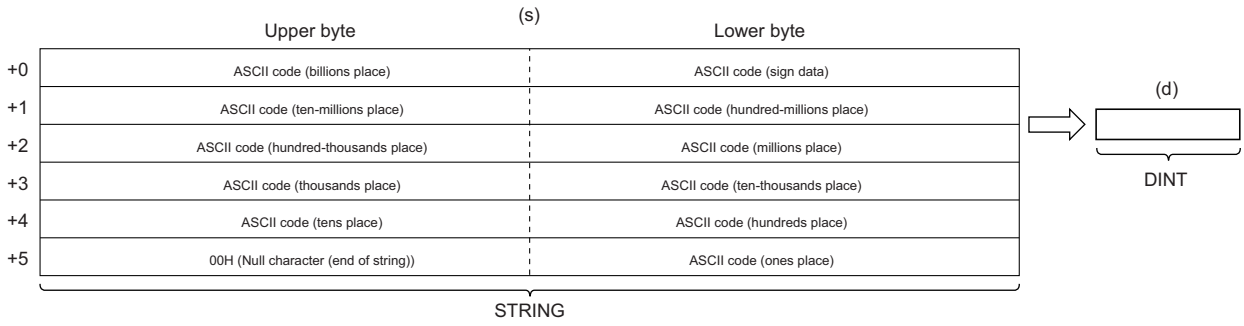
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(11)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

- These functions convert the value input to (s) from STRING data type to DINT data type, and output the converted value from (d).



- Input a STRING data type value to (s) within the following range.
  - ASCII code: 30H to 39H, 20H, 2DH, and 00H
  - STRING data type value: -2147483648 to 2147483647

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

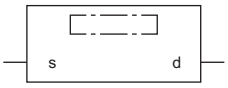
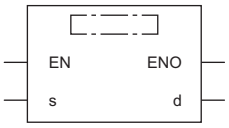
Error code (SD0)	Description
3281H	Invalid data that cannot be converted are input in (s). • ASCII code: other than 30H to 39H, 20H, and 00H • STRING data type value: other than -2147483648 to 2147483647

27.58 Converting STRING to BCD

STRING_TO_BCD(_E)

These functions convert a value from STRING data type to BCD data type.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=STRING_TO_BCD(s);
		[With EN/ENO]
		d:=STRING_TO_BCD_E(EN,ENO,s);

Setting data

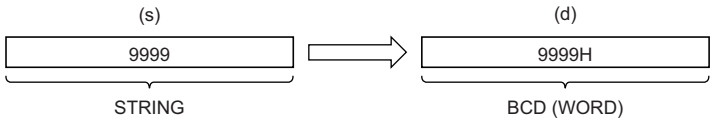
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(8)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

Processing details

■Operation processing

- These functions convert the value input to (s) from STRING data type to BCD (WORD) data type, and output the converted value from (d).



- The ASCII code 20H (space) that exists in the string is ignored.
- The ASCII codes 20H (space) and 30H (0) that exist in the string are counted as one character as well.
- Input a STRING data type value to (s) within the following range.

ASCII code: 30H to 39H, 20H, 00H

- If the string input has less than four characters, the string will be supplemented with zero(s). (Zero(s) is/are added at the end of the string.) For this reason, if a string shorter than four characters is to be converted, input a string padded with 0s (e.g. '0001' for '1').
- If the string length exceeds 4 characters, the four left characters are regarded as the targets to convert.

Input string	Conversion target string	Output (BCD data type)
'1'	'1000'	1000H (4096D)
'12'	'1200'	1200H(4608D)
'123'	'1230'	1230H(4656D)
'1234'	'1234'	1234H(4660D)
'12345'	'1234'	1234H(4660D)

- WORD or DWORD data type can be specified for (d). BOOL data type cannot be specified.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

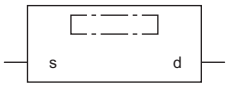
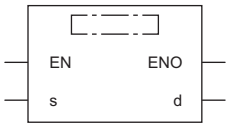
Error code (SD0)	Description
3281H	An ASCII code other than 30H to 39H, 20H, or 00H is input.

27.59 Converting STRING to REAL

STRING_TO_REAL(_E)

These functions convert a value from STRING data type to REAL data type.

Ladder, FBD/LD		ST

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=STRING_TO_REAL(s);
		[With EN/ENO]
		d:=STRING_TO_REAL_E(EN,ENO,s);

Setting data

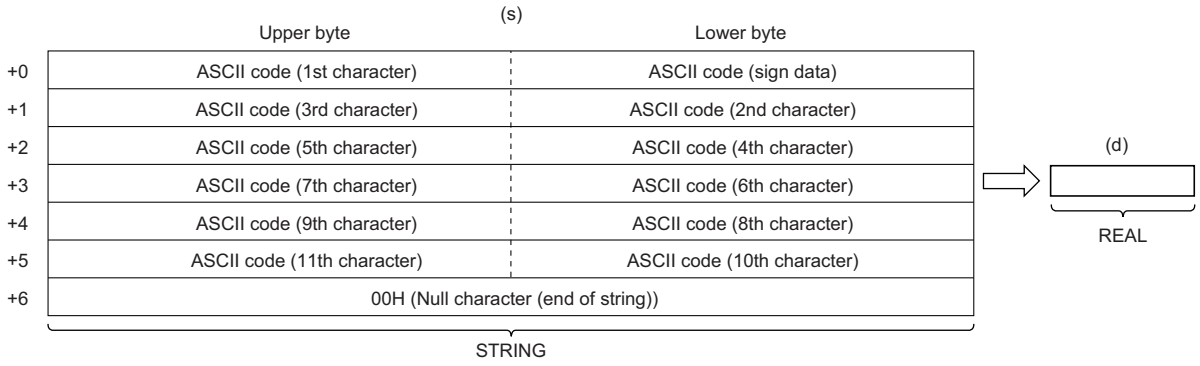
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(24)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	REAL

Processing details

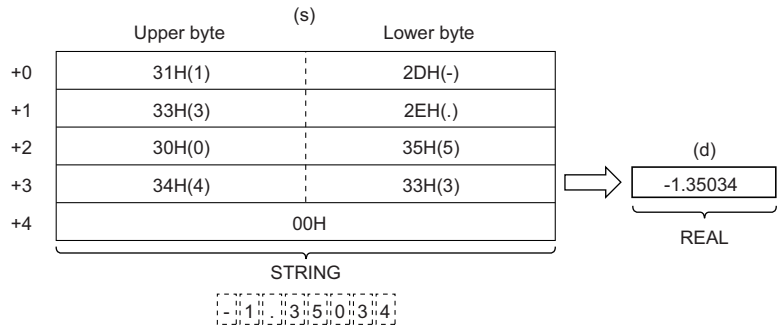
■Operation processing

- These functions convert the value input to (s) from STRING data type (decimal form/exponential form) to REAL data type, and output the converted value from (d).

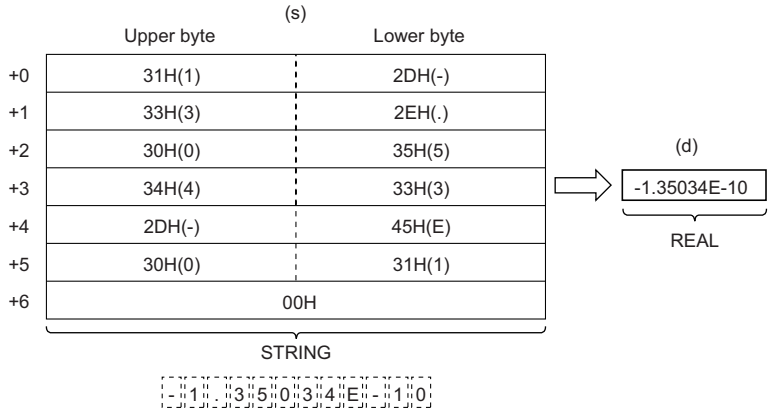


- The STRING data type value both in decimal form and exponential form can be converted.

• When (s) is in decimal form

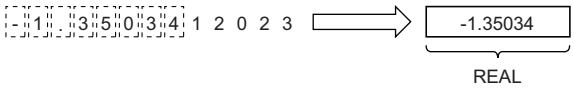


• When (s) is in exponential form

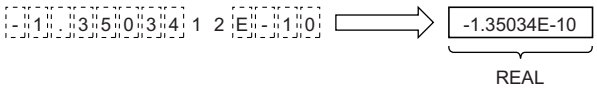


- The number of significant digits of the STRING data type value is six. (The sign, decimal point, and exponent are not included.) The seventh and later digits are rounded down when the data is converted.

• When (s) is in decimal form



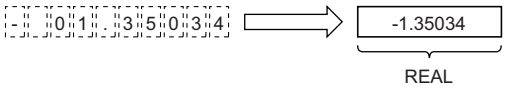
• When (s) is in exponential form



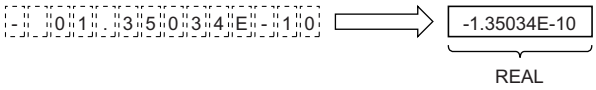
- In decimal form, when 2BH (+) is specified as sign data or the sign data is omitted, the data is converted as a positive value. When 2DH (-) is specified, the data is converted as a negative value.
- In exponential form, when 2BH (+) is specified as sign data for the exponent or the sign data is omitted, the data is converted as a positive value. When 2DH (-) is specified, the data is converted as a negative value.

- The ASCII code 20H (space) or 30H (0) that exists before the first numerical value 0 in the STRING data type value is ignored.

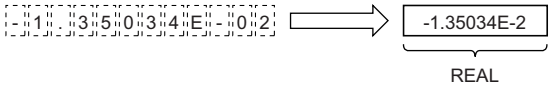
• When (s) is in decimal form



• When (s) is in exponential form



- The ASCII code 30H (0) that exists between E and a numerical value in the STRING data type value is ignored (in exponential form only).



- The ASCII code 20H (space) that exists in the string is ignored.
- Up to 24 characters can be input. The ASCII codes 20H (space) and 30H (0) that exist in the string are counted as one character as well.
- Input a STRING data type value to (s) within the following range.
  - ASCII code: 30H to 39H, 45H, 2BH, 2DH, 2EH, 20H, and 00H

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the device area (between the specified device number and the last device number) specified by (s).
3281H	Invalid data that cannot be converted are set to (s). • The integral part or decimal part contains a character other than 30H(0) to 39H(9). • More than one 2EH (.) exists in the specified string. • The exponent of the specified string contains a character other than 45H(E), 65H(e), 2BH(+), and 2DH(-). • The specified string contains more than one exponent 45H(E) or 65H(e). • The exponent in the specified string contains a numerical value consisting of three digits or more. • The exponent of the specified string contains more than one sign 2BH(+) or 2DH(-). • The specified string (in the integral part if the decimal format is used or in the mantissa if the exponent format is used) contains more than one sign data of 2BH(+) or 2DH(-). • The number of characters in the device specified by (s) and later is 0 or exceeds 24.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

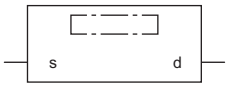
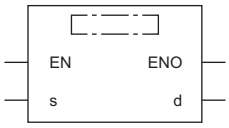
27.60 Converting STRING to TIME

STRING_TO_TIME(_E)

These functions convert a value from STRING data type to TIME data type.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=STRING_TO_TIME(s);
		[With EN/ENO]
		d:=STRING_TO_TIME_E(EN,ENO,s);

Setting data

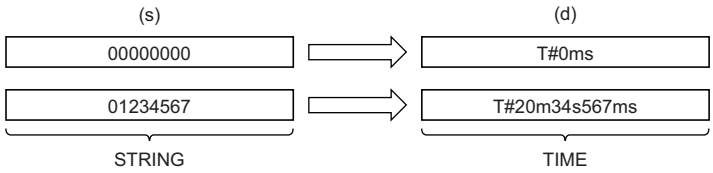
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	STRING(11)
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

Processing details

■Operation processing

- These functions convert the value input to (s) from STRING data type to TIME data type, and output the converted value from (d).



- Input a STRING data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3281H	An ASCII code other than 30H to 39H, 20H, and 00H is input.
	The STRING data type value input is out of the following range: -2147483648 to 2147483647

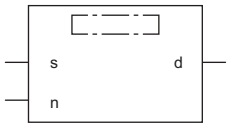
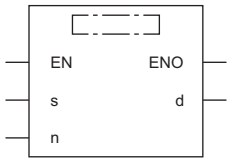
27.61 Converting Bit Array to INT

BITARR_TO_INT(_E)

These functions convert the specified number of bits in a bit array to an INT data type value.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=BITARR_TO_INT(s,n);
		[With EN/ENO]
		d:=BITARR_TO_INT_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (BitArr)	Input (An element can be specified by a variable.)	Input variable	Boolean array element
n	Number of bits (4, 8, 12, or 16)	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	ANY16

Processing details

■Operation processing

- These functions convert the number of bits specified by (n) starting from the bit array element input to (s) to ANY16 type data, and output the converted value from (d).
- Zeros (0s) are set for all the bits exceeding the specified number of bits.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

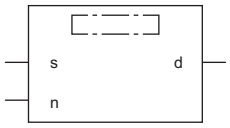
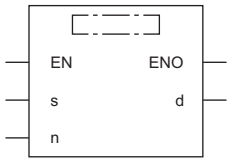
Operation error

There is no operation error.

27.62 Converting Bit Array to DINT

BITARR_TO_DINT(_E)

These functions convert the specified number of bits in a bit array to a DINT data type value.

Ladder, FBD/LD		ST
[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO] d:=BITARR_TO_DINT(s,n) [With EN/ENO] d:=BITARR_TO_DINT_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (BitArr)	Input (An element can be specified by a variable.)	Input variable	Boolean array element
n	Number of bits (4, 8, 12, 16, 20, 24, 28, or 32)	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	ANY32

Processing details

■Operation processing

- These functions convert the number of bits specified by (n) starting from the bit array element input to (s) to ANY32 type data, and output the converted value from (d).
- Zeros (0s) are set for all the bits exceeding the specified number of bits.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

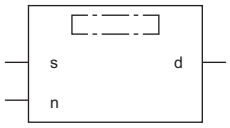
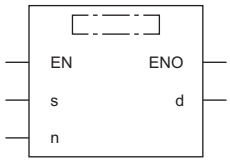
There is no operation error.

27.63 Converting INT to Bit Array

INT_TO_BITARR(_E)

These functions output the lower n bits of the INT data type value to the bit array.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=INT_TO_BITARR(s,n);
		[With EN/ENO]
		d:=INT_TO_BITARR_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s	Input	Input variable	ANY16
n	Number of bits (4, 8, 12, or 16)	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output (An element can be specified by a variable.)	Output variable	Boolean array element

Processing details

■Operation processing

- These functions output the lower (n) bits of ANY16 type data specified by (s) to (d).
- The output bits beyond the specified number of bits are not changed.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

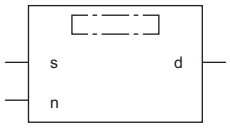
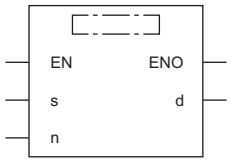
There is no operation error.

27.64 Converting DINT to Bit Array

DINT_TO_BITARR(_E)

These functions output the lower n bits of the DINT data type value to the bit array.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DINT_TO_BITARR(s,n);
		[With EN/ENO]
		d:=DINT_TO_BITARR_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s	Input	Input variable	ANY32
n	Number of bits (4, 8, 12, 16, 20, 24, 28, or 32)	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output (An element can be specified by a variable.)	Output variable	Boolean array element

Processing details

■Operation processing

- These functions output the lower (n) bits of ANY32 type data specified by (s) to (d).
- The output bits beyond the specified number of bits are not changed.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

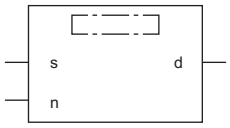
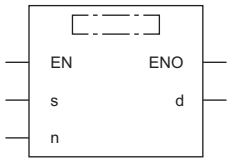
There is no operation error.

27.65 Copying the Bit Array

CPY_BITARR(_E)

These functions copy the bit array by the specified number of bits.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=CPY_BITARR(s,n);
		[With EN/ENO]
		d:=CPY_BITARR_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s(BitArrIn)	Input	Input variable	Boolean array element
n	Number of bits (4, 8, 12, 16, 20, 24, 28, or 32)	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	Boolean array element

Processing details

■Operation processing

- These functions output the bit array (number of (n) bits) specified by (s) to (d).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

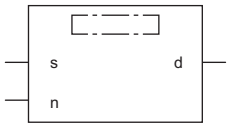
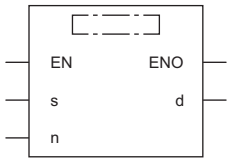
27.66 Reading the Specified Bit of the Word Label

GET_BIT_OF_INT(_E)

These functions read a value from the specified bit of a word label.

Ladder*1		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=GET_BIT_OF_INT(s,n);
		[With EN/ENO]
		d:=GET_BIT_OF_INT_E(EN,ENO,s,n);

*1 FBD/LD is not supported. For FBD/LD, use the bit specification of labels.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s	Input	Input variable	INT*1
n	Number of bits (0 to 15)	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Output	Output variable	BOOL

*1 Digit-specified labels and devices cannot be used.

Processing details

■Operation processing

- These functions output a value in the (n)th bit of (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

Point

By using the bit specification of labels, a concise program having the same operation as GET_BIT_OF_INT can be created.

The following example reads the value in bit 5 (b5) of g_int1 into g_bool1 the same as when GET_BIT_OF_INT is used.

Ladder



ST

```
g_bool1 := g_int1.5;
```

FBD/LD



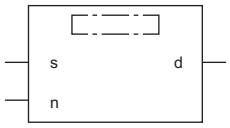
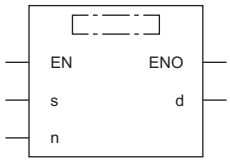
27.67 Writing the Specified Bit of the Word Label

SET_BIT_OF_INT(_E)

These functions write a value to the specified bit of a word label.

Ladder*1		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=SET_BIT_OF_INT(s,n);
		[With EN/ENO]
		d:=SET_BIT_OF_INT_E(EN,ENO,s,n);

*1 FBD/LD is not supported. For FBD/LD, use the bit specification of labels.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s	Input	Input variable	BOOL
n	Number of bits (0 to 15)	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Input/output	Input/output variable	INT*1

*1 Digit-specified labels and devices cannot be used.

Processing details

■Operation processing

- These functions write a BOOL data type value specified by (s) to the (n)th bit of (d).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Precautions

When using SET_BIT_OF_INT(_E) in ST, create a program which assigns the return value to a variable.

```
//The return value of SET_BIT_OF_INT is assigned to a variable and used.
g_int1 := SET_BIT_OF_INT(TRUE, 0);
g_bool1 := GET_BIT_OF_INT(g_int1, 0);
```

The return value of SET_BIT_OF_INT(_E) also works as input/output. Do not directly assign it to the input argument of another instruction, function, or function block.

```
//In the following program, the value of the first argument of GET_BIT_OF_INT is undefined.
g_bool1 := GET_BIT_OF_INT( SET_BIT_OF_INT(TRUE, 0), 0);
```

Operation error

There is no operation error.

Point

By using the bit specification of labels, a concise program having the same operation as SET_BIT_OF_INT can be created.

The following example changes the value in bit 5 (b5) of g_int1 to the value of g_bool1 the same as when SET_BIT_OF_INT is used.

Ladder



ST

```
g_int1.5 := g_bool1;
```

FBD/LD



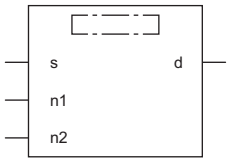
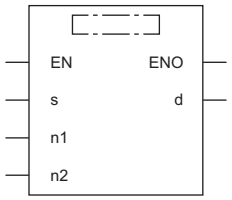
27.68 Copying the Specified Bit of the Word Label

CPY_BIT_OF_INT(_E)

These functions copy the specified bit of the word label to the specified bit of another word label.

Ladder*1		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=CPY_BIT_OF_INT(s,n1,n2);
		[With EN/ENO]
		d:=CPY_BIT_OF_INT_E(EN,ENO,s,n1,n2);

*1 FBD/LD is not supported. For FBD/LD, use the bit specification of labels.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s	Input	Input variable	INT*1
n1	Number of bits in input variable (0 to 15)	Input variable	INT
n2	Number of bits in output variable (0 to 15)	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d	Input/output	Input/output variable	INT*1

*1 Digit-specified labels and devices cannot be used.

Processing details

■Operation processing

- These functions copy the value in the (n1)th bit of the word specified by (s) to the (n2)th bit of (d).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Precautions

When using CPY_BIT_OF_INT(_E) in ST, create a program which assigns the return value to a variable.

```
//The return value of CPY_BIT_OF_INT is assigned to a variable and used.
g_int2 := CPY_BIT_OF_INT(g_int1,5,3);
g_bool1 := GET_BIT_OF_INT(g_int2,3);
```

The return value of CPY_BIT_OF_INT(_E) also works as input/output. Do not directly assign it to the input argument of another instruction, function, or function block.

```
//In the following program, the value of the first argument of GET_BIT_OF_INT is undefined.
g_bool1 := GET_BIT_OF_INT( CPY_BIT_OF_INT(g_int1,5,3), 3);
```

Operation error

There is no operation error.

Point

By using the bit specification of labels, a concise program having the same operation as CPY_BIT_OF_INT can be created.

The following example changes the value in bit 3 (b3) of g_int2 to the value of bit 5 (b5) of g_int1 the same as when CPY_BIT_OF_INT is used.

Ladder



ST

```
g_int2.3 := g_int1.5;
```

FBD/LD

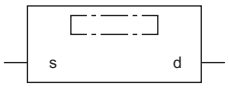


27.69 Getting the Start Data

GET_BOOL_ADDR, GET_INT_ADDR, GET_WORD_ADDR

These functions output the start data of the specified array as BOOL, INT, or WORD type data.

Ladder, FBD/LD	ST

	d:=GET_BOOL_ADDR(s)
	d:=GET_INT_ADDR(s);
	d:=GET_WORD_ADDR(s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s	Input	Input variable	ANY
d	Output	Output variable	BOOL/INT/WORD

Processing details

■Operation processing

- The GET_BOOL_ADDR function outputs the start data of the array specified by (s) as BOOL type data.
- The GET_INT_ADDR function outputs the start data of the array specified by (s) as INT type data. When the data type whose bit length is 32 bits or longer is specified, the lower 16 bits are acquired.
- The GET_WORD_ADDR function outputs the start data of the array specified by (s) as WORD type data. When the data type whose bit length is 32 bits or longer is specified, the lower 16 bits are acquired.

Standard function	Input data type	Output data type
GET_BOOL_ADDR	BOOL ARRAY OF BOOL	BOOL
GET_INT_ADDR	INT DINT WORD REAL TIME STRING ARRAY OF INT ARRAY OF DINT ARRAY OF WORD ARRAY OF DWORD ARRAY OF REAL ARRAY OF TIME	INT
GET_WORD_ADDR		WORD

■Operation result

The operation processing is performed. The operation result is output from (d).

Operation error

There is no operation error.

28 SINGLE VARIABLE FUNCTIONS

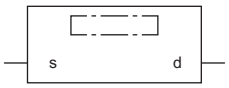
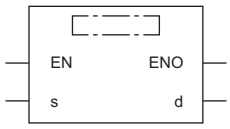
28.1 Calculating the Absolute Value

ABS(_E)

These functions output the absolute value of an input value.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=ABS(s);
		[With EN/ENO]
		d:=ABS_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_NUM

Processing details

■Operation processing

- These functions output the absolute value of the INT, DINT, REAL, or LREAL data type value input to (s), in the same type of data as (s), from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

B = |A|

- Input an INT, DINT, REAL, or LREAL data type value to (s).
- If -32768 in INT data type is input to (s), (d) will output -32768.
- If -2147483648 in DINT data type is input to (s), (d) will output -2147483648. (No operation error occurs. When ABS_E is used, ENO outputs TRUE.)

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s) is of REAL data type

Error code (SD0)	Description
3282H	The value output from (d) is -0, a subnormal number, NaN (not a number), or ±∞.

- When (s) is of LREAL data type

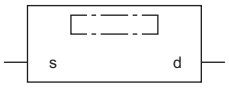
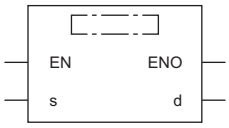
Error code (SD0)	Description
3282H	The value output from (d) is -0, a subnormal number, NaN (not a number), or ±∞.

28.2 Calculating the Square Root

SQRT(_E)

These functions calculate the square root of an input value.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=SQRT(s);
		[With EN/ENO]
		d:=SQRT_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the square root of the REAL/LREAL data type value input to (s) and store the operation result in (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

$B = \sqrt{A}$

- Input a positive REAL/LREAL data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

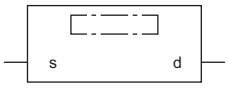
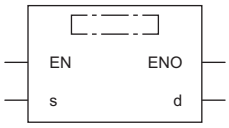
Error code (SD0)	Description
3285H	The input value is negative.

28.3 Calculating the Natural Logarithm

LN(_E)

These functions output the natural logarithm (logarithm with base e) of an input value.

Ladder, FBD/LD		ST

[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO]
		d:=LN(s);
		[With EN/ENO]
		d:=LN_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the natural logarithm of the REAL/LREAL data type value input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

$B=\log_e A$

- Natural logarithm operation is performed with the base (e) defined as 2.71828.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

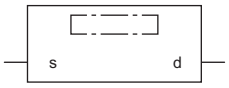
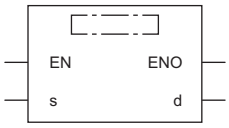
Error code (SD0)	Description
3285H	The input value is 0 or negative.

28.4 Calculating the Common Logarithm

LOG(_E)

These functions output the common logarithm (logarithm with base 10) of an input value.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=LOG(s);
		[With EN/ENO]
		d:=LOG_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the common logarithm of the REAL or LREAL data type value input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

$B=\log_{10}A$

- Input a REAL or LREAL data type value to (s).
- Input a positive value only. (Calculation cannot be performed with a negative value.)
- If the operation result is -0 or an underflow occurs, 0 will be output as the operation result.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s) is of REAL data type

Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3285H	Out-of-range data is set to (s). • The specified value is a negative number. • The specified value is 0.

- When (s) is of LREAL data type

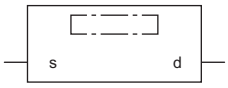
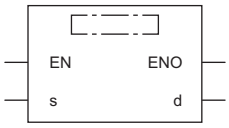
Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3285H	Out-of-range data is set to (s). • The specified value is a negative number. • The specified value is 0.

28.5 Calculating the Exponent

EXP(_E)

These functions output the exponent of an input value.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=EXP(s);
		[With EN/ENO]
		d:=EXP_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the exponent of the REAL/LREAL data type value input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

$B=e^{A}$

- Exponent operation is performed with the base (e) defined as 2.71828.
- Input a REAL or LREAL data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

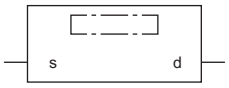
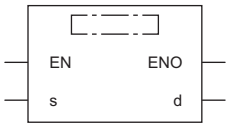
Error code (SD0)	Description
3282H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

28.6 Calculating the Sine

SIN(_E)

These functions output the sine of an input value.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=SIN(s);
		[With EN/ENO]
		d:=SIN_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the sine of the REAL data type value (angle) input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

B=SIN A

- Input a REAL data type value to (s). Input a value (angle) in radians (angle×π/180).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3282H	The data specified by (s) is -0.

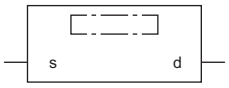
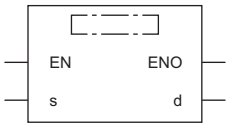
28.7 Calculating the Cosine

COS(_E)

These functions output the cosine of an input value.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=COS(s);
		[With EN/ENO]
		d:=COS_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the cosine of the REAL data type value (angle) input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

B=COS A

- Input a REAL data type value to (s). Input a value (angle) in radians (angle×π/180).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3282H	The data specified by (s) is -0.

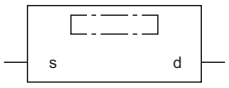
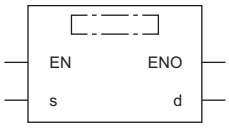
28.8 Calculating the Tangent

TAN(_E)

These functions output the tangent of an input value.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=TAN(s);
		[With EN/ENO]
		d:=TAN_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the tangent of the REAL data type value (angle) input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

B=TAN A

- Note that even if the input value is π/2 radian or (3/2)π radian, no error will be issued because of the truncation error in the radian value.
- Input a REAL data type value to (s). Input a value (angle) in radians (angle×π/180).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

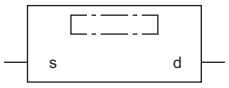
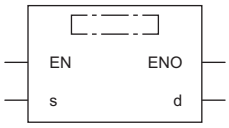
Error code (SD0)	Description
3282H	The data specified by (s) is -0.

28.9 Calculating the Arc Sine

ASIN(_E)

These functions output the arc sine (SIN^{-1}) of an input value.

Ladder, FBD/LD		ST

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=ASIN(s);
		[With EN/ENO]
		d:=ASIN_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the arc sine (SIN^{-1}) of the REAL data type value input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

$B=SIN^{-1} A$

- Input a REAL data type value to (s) within the following range.

ASIN(_E): -1.0 to 1.0

- The value (angle) is output from (d) in radians (angle×π/180).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3282H	The data specified by (s) is -0.
3285H	The value input with ASIN(_E) is other than -1.0 to 1.0.

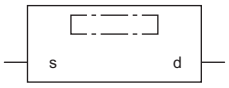
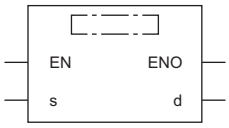
28.10 Calculating the Arc Cosine

ACOS(_E)

These functions output the arc cosine (COS^{-1}) of an input value.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=ACOS(s);
		[With EN/ENO]
		d:=ACOS_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the arc cosine (COS^{-1}) of the REAL data type value input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

$B=COS^{-1} A$

- Input a REAL data type value to (s) within the following range.

ACOS(_E): -1.0 to 1.0

- The value (angle) is output from (d) in radians (angle×π/180).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

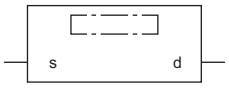
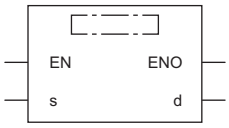
Error code (SD0)	Description
3282H	The data specified by (s) is -0.
3285H	The value input with ACOS(_E) is other than -1.0 to 1.0.

28.11 Calculating the Arc Tangent

ATAN(_E)

These functions output the arc tangent (TAN^{-1}) of an input value.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=ATAN(s);
		[With EN/ENO]
		d:=ATAN_E(EN,ENO,s);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_REAL
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions calculate the arc tangent (TAN^{-1}) of the REAL data type value input to (s), and output the operation result from (d).
- When the input value is defined as A and the output value is defined as B, the relationship of A and B will be as follows:

$B=TAN^{-1} A$

- Input a REAL data type value to (s) within the following range.

ATAN(_E): $\pm 1.17549^{-38}$ to $\pm 3.40282^{+38}$

- The value (angle) is output from (d) in radians (angle$\times\pi$/180).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3282H	The data specified by (s) is -0.

29 ARITHMETIC OPERATION FUNCTIONS

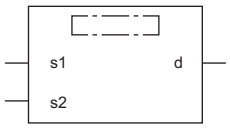
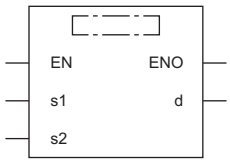
29.1 Addition

ADD(_E)

These functions output the sum of input values ((s1)+(s2)+···+(s28)).

Ladder, FBD/LD*1		ST*1

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=ADD(s1,s2);
		[With EN/ENO]
		d:=ADD_E(EN,ENO,s1,s2);

*1 The input variable s can be changed within the range from 2 to 28.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1) to s28 (IN28)	Input	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_NUM

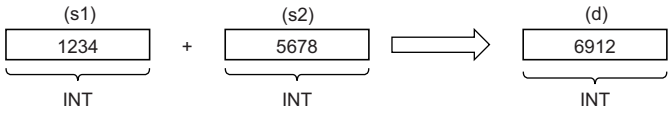
Processing details

■Operation processing

- These functions add the INT, DINT, WORD, DWORD, REAL, or LREAL data type values input to (s1) to (s28) ((s1)+(s2)+···+(s28)), and output the operation result, in the same data type as (s), from (d).

Ex.

Data type: INT



- Input an INT, DINT, WORD, DWORD, REAL, or LREAL data type value to (s1) to (s28).
- If an underflow or overflow occurs in the operation result, the output from (d) will be as follows.

Data type	Description
INT	Even if an underflow or overflow occurs, no operation error is issued. When ADD_E is used, ENO outputs TRUE. [Example 1] 32767+2=-32767 (7FFFH)+(0002H)=(8001H) A negative value results because the most significant bit is 1. [Example 2] -32767+(-2)=32766 (8000H)+(FFFEH)=(7FFEH) A positive value results because the most significant bit is 0.
DINT	Even if an underflow or overflow occurs, no operation error is issued. When ADD_E is used, ENO outputs TRUE. [Example 1] 2147483647+2=-2147483647 (7FFFFFFFH)+(00000002H)=(80000001H) A negative value results because the most significant bit is 1. [Example 2] -2147483648+(-2)=2147483646 (80000000H)+(FFFEH)=(7FFFFFFEH) A positive value results because the most significant bit is 0.
WORD	Even if an overflow occurs, no operation error is issued. When ADD_E is used, ENO outputs TRUE. [Example] 65535 + 1 = 0 (FFFFH) + (0001H) = (0000H)
DWORD	Even if an overflow occurs, no operation error is issued. When ADD_E is used, ENO outputs TRUE. [Example] 4294967295 + 1 = 0 (FFFFFFFFH) + (00000001H) = (00000000H)
REAL	An operation error occurs and an undefined value is output.
LREAL	

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

• When (s1) to (s28) are of REAL data type

Error code (SD0)	Description
3282H	The value input to (s1) to (s28) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value output from (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

• When (s1) to (s28) are of LREAL data type

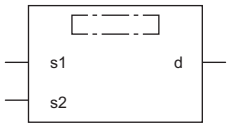
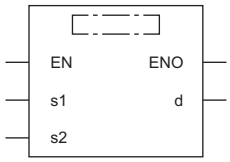
Error code (SD0)	Description
3282H	The value input to (s1) to (s28) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value output from (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{1024}$

29.2 Multiplication

MUL(_E)

These functions output the product of input values ((s1)×(s2)×···×(s28)).

Ladder, FBD/LD*1		ST*1

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=MUL(s1,s2);
		[With EN/ENO]
		d:=MUL_E(EN,ENO,s1,s2);

*1 The input variable s can be changed within the range from 2 to 28.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1) to s28 (IN28)	Input	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_NUM

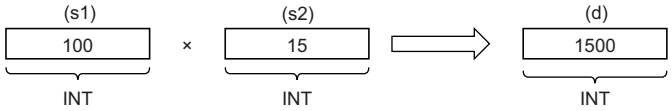
Processing details

■Operation processing

- These functions multiply the INT, DINT, WORD, DWORD, REAL, or LREAL data type values input to (s1) to (s28) ((s1)×(s2)×···×(s28)), and output the operation result, in the same data type as (s), from (d).

Ex.

Data type: INT



- Input an INT, DINT, WORD, DWORD, REAL, or LREAL data type value to (s1) to (s28).
- If an underflow or overflow occurs in the operation result, the output from (d) will be as follows.

Data type	Description
INT WORD	• Even if an underflow or overflow occurs, no operation error is issued. When MUL_E is used, ENO outputs TRUE. • Even if the operation result is outside the INT or WORD data type range, the INT or WORD data type value is output; (In this case, the output value is of INT or WORD data type with the upper 16 bits deleted although the operation result is a DINT or DWORD data type value.) • If the operation result is outside the INT or WORD data type range, convert the input value to the DINT or DWORD data type by using the INT_TO_DINT or WORD_TO_DWORD function, and then perform operation.
DINT DWORD	• Even if an underflow or overflow occurs, no operation error is issued. When MUL_E is used, ENO outputs TRUE. • Even if the operation result is outside the DINT or DWORD data type range, the DINT or DWORD data type value is output; (In this case, the output value is of DINT or DWORD data type with the upper 32 bits deleted although the operation result is 64-bit data.) • If the operation result is outside the DINT or DWORD data type range, convert the input value to the REAL data type by using the DINT_TO_REAL function, and then perform operation.
REAL	An operation error occurs and an undefined value is output.
LREAL	

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Point

If the operation result is outside the data type range, convert the input value as appropriate before operation.

Operation error

• When (s1) to (s28) are of REAL data type

Error code (SD0)	Description
3282H	The value input to (s1) to (s28) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

• When (s1) to (s28) are of LREAL data type

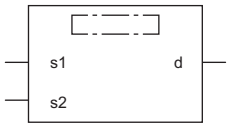
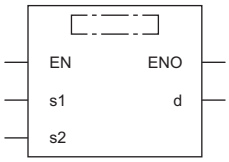
Error code (SD0)	Description
3282H	The value input to (s1) to (s28) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{1024}$

29.3 Subtraction

SUB(_E)

These functions output the difference between input values ((s1)-(s2)).

Ladder, FBD/LD		ST

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=SUB(s1,s2);
		[With EN/ENO]
		d:=SUB_E(EN,ENO,s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1), s2 (IN2)	Input	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_NUM

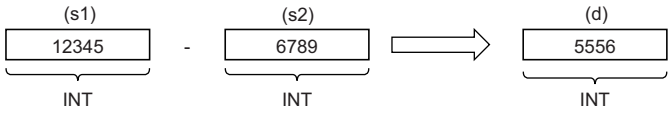
Processing details

■Operation processing

- These functions perform subtraction between the INT, DINT, WORD, DWORD, REAL, or LREAL data type values input to (s1) and (s2) ((s1)-(s2)), and output the operation result, in the same data type as (s), from (d).

Ex.

Data type: INT



- Input an INT, DINT, WORD, DWORD, REAL, or LREAL data type value to (s1) and (s2).
- If an underflow or overflow occurs in the operation result, the output from (d) will be as follows.

Data type	Description
INT	Even if an underflow or overflow occurs, no operation error is issued. When SUB_E is used, ENO outputs TRUE. [Example 1] 32767-(-2)=-32767 (7FFFH)-(FFFEH)=(8001H) A negative value results because the most significant bit is 1. [Example 2] -32767-2=32766 (8000H)-(0002H)=(7FFEH) A positive value results because the most significant bit is 0.
DINT	Even if an underflow or overflow occurs, no operation error is issued. When SUB_E is used, ENO outputs TRUE. [Example 1] 2147483647-(-2)=-2147483647 (7FFFFFFFH)-(0000FFFEH)=(80000001H) A negative value results because the most significant bit is 1. [Example 2] -2147483648-2=2147483646 (80000000H)-(00000002H)=(7FFFFFFEH) A positive value results because the most significant bit is 0.
WORD	Even if an underflow occurs, no operation error is issued. When ADD_E is used, ENO outputs TRUE. [Example] 0 - 1 = 65535 (0000H) - (0001H) = (FFFFH)
DWORD	Even if an underflow occurs, no operation error is issued. When ADD_E is used, ENO outputs TRUE. [Example] 0 - 1 = 4294967295 (00000000H) - (00000001H) = (FFFFFFFFH)
REAL	An operation error occurs and an undefined value is output.
LREAL	

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

• When (s1) and (s2) are of REAL data type

Error code (SD0)	Description
3282H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value output from (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

• When (s1) and (s2) are of LREAL data type

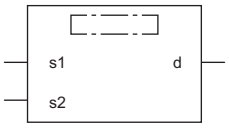
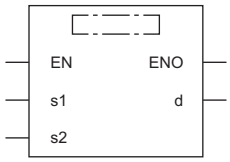
Error code (SD0)	Description
3282H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value output from (d) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{1024}$

29.4 Division

DIV(_E)

These functions output the quotient of input values ((s1)÷(s2)).

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DIV(s1,s2);
		[With EN/ENO]
		d:=DIV_E(EN,ENO,s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1)	Dividend	Input variable	ANY_NUM
s2 (IN2)	Divisor	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_NUM

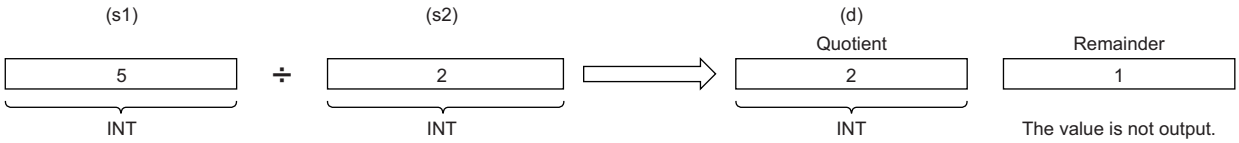
Processing details

■Operation processing

- These functions perform division between the INT, DINT, WORD, DWORD, REAL, or LREAL data type values input to (s1) and (s2) ((s1)÷(s2)), and output the operation result, in the same data type as (s), from (d).

Ex.

Data type: INT



- Input an INT, DINT, WORD, DWORD, REAL, or LREAL data type value to (s1) and (s2). (Note that the value input to (s2) shall be other than 0.)

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s1) and (s2) are of INT or WORD data type

Error code (SD0)	Description
3280H	The value (divisor) input to (s2) is 0.

- When (s1) and (s2) are of DINT or DWORD data type

Error code (SD0)	Description
3280H	The value (divisor) input to (s2) is 0.

- When (s1) and (s2) are of REAL data type

Error code (SD0)	Description
3280H	The value (divisor) input to (s2) is 0.
3282H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

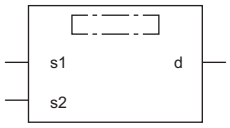
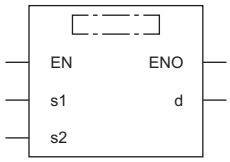
- When (s1) and (s2) are of LREAL data type

Error code (SD0)	Description
3280H	The value (divisor) input to (s2) is 0.
3282H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3283H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{1024}$

29.5 Remainder

MOD(_E)

These functions output the remainder of input values ((s1)÷(s2)).

Ladder, FBD/LD		ST
[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO] The function is described as an operator. (☞ Page 93 Operators) [With EN/ENO] d:=MOD_E(EN,ENO,s1,s2);
		

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1)	Dividend	Input variable	ANY_INT
s2 (IN2)	Divisor	Input variable	ANY_INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_INT

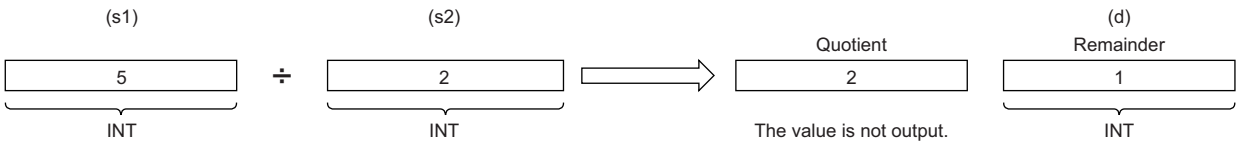
Processing details

■Operation processing

- These functions perform division between the INT, DINT, WORD, or DWORD data type values input to (s1) and (s2) ((s1)÷(s2)), and output the remainder of the operation result, in the same data type as (s), from (d).

Ex.

Data type: INT



- Input an INT, DINT, WORD, or DWORD data type value to (s1) and (s2). (Note that the value input to (s2) shall be other than 0.)

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s1) and (s2) are of INT or WORD data type

Error code (SD0)	Description
3280H	The value (divisor) input to (s2) is 0.

- When (s1) and (s2) are of DINT or DWORD data type

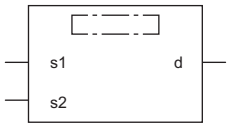
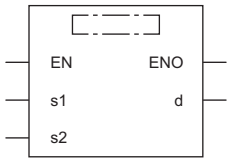
Error code (SD0)	Description
3280H	The value (divisor) input to (s2) is 0.

29.6 Exponentiation

EXPT(_E)

These functions output the exponentiation of an input value.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=EXPT(s1,s2);
		[With EN/ENO]
		d:=EXPT_E(EN,ENO,s1,s2);

Setting data

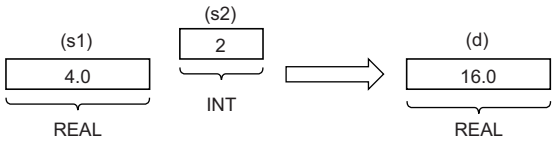
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1)	Base	Input variable	ANY_REAL
s2 (IN2)	Exponent (power)	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_REAL

Processing details

■Operation processing

- These functions exponentiate the REAL or LREAL data type value input to (s1) with the exponent (INT, DINT, REAL, or LREAL data type) input to (s2), and output the operation result from (d).



■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

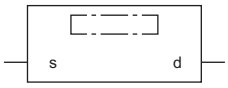
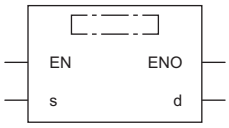
Error code (SD0)	Description
3282H	The value input to (s1) or (s2) is -0, a subnormal number, NaN (not a number), or ±∞.

29.7 Assignment

MOVE(_E)

These functions output the assignment value of an input value.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=MOVE(s);
		[With EN/ENO]
		d:=MOVE_E(EN,ENO,s);

Setting data

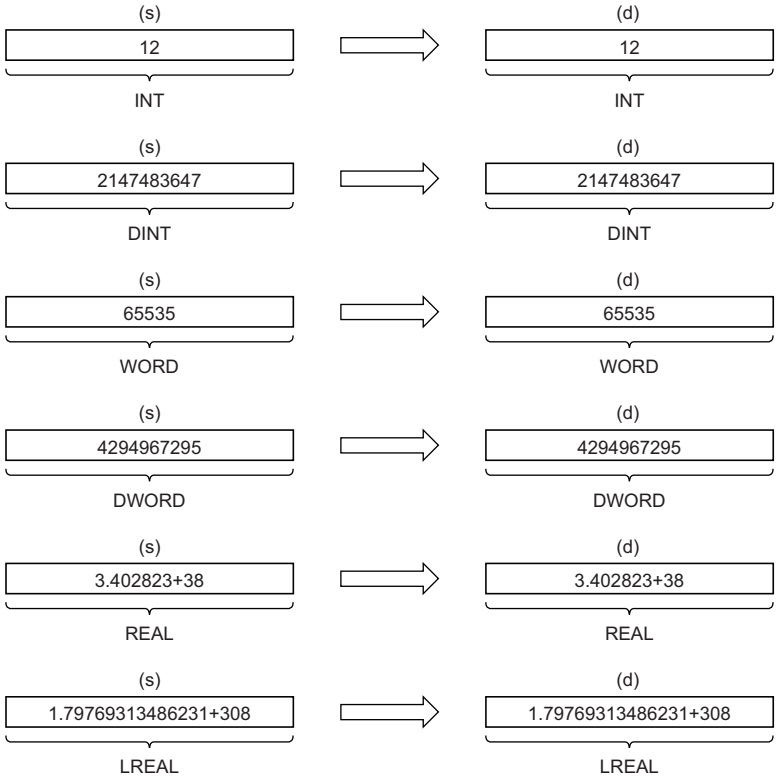
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY

Processing details

■Operation processing

- These functions assign the value of the input variable specified by (s) to the output variable specified by (d).
- Input a BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, TIME, structure, or array data type value to (s) and (d). The values input to (s) and (d) must be of the same data type.



■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s) and later.
3285H	The number of characters in the string input to (s) exceeds 16383.
3286H	The entire string cannot be stored in each setting area in the device/label memory in the device specified by (d) and later. (The number of required points is insufficient.)

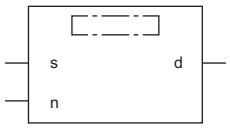
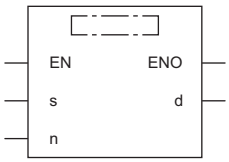
30 BIT SHIFT FUNCTIONS

30.1 Shifting Data to the Left by n Bit(s)

SHL(_E)

These functions shift the input value to the left by (n) bit(s), and output the operation result.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=SHL(s,n);
		[With EN/ENO]
		d:=SHL_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_BIT
n (N)	Number of bits to be shifted	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

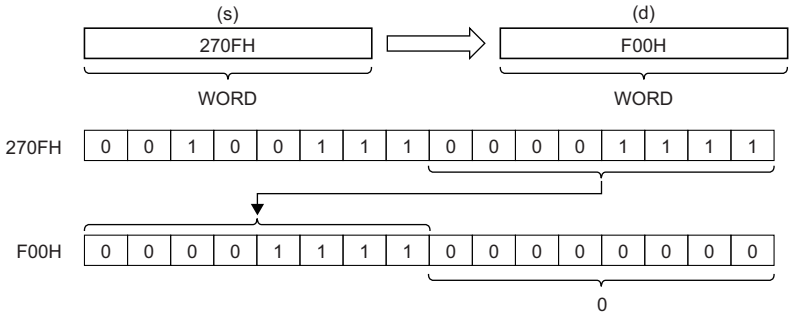
Processing details

■Operation processing

- These functions shift the 16-bit data or 32-bit data (WORD or DWORD data type value) input to (s) to the left by (n) bit(s), and output the operation result, in the same data type as (s), from (d).
- Specify the number of bits to be shifted in (n).

Ex.

Data type of (s): 16-bit data (WORD data type), Value input to (n): 8



- The (n) bit(s) from the least significant bit is/are filled with 0(s).
- Input a 16-bit data or 32-bit data (WORD or DWORD data type value) to (s).
- Input a value to (n) (Number of bits to be shifted) within the following range.

Data type of (s): 16-bit data (WORD data type)	Data type of (s): 32-bit data (DWORD data type)
Range: 0 to 15 The lower 4-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 6	Range: 0 to 31 The lower 5-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 22

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

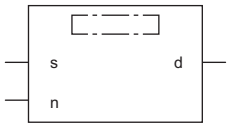
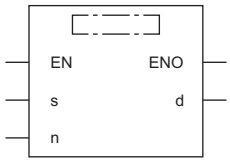
30.2 Shifting Data to the Right by n Bit(s)

SHR(_E)

These functions shift the input value to the right by (n) bit(s), and output the operation result.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=SHR(s,n);
		[With EN/ENO]
		d:=SHR_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_BIT
n (N)	Number of bits to be shifted	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

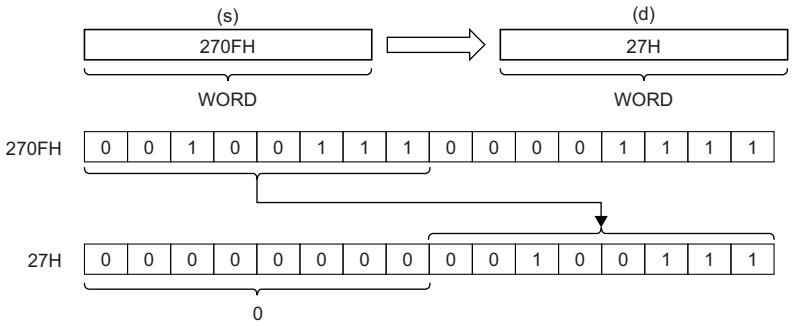
Processing details

■Operation processing

- These functions shift the 16-bit data or 32-bit data (WORD or DWORD data type value) input to (s) to the right by (n) bit(s), and output the operation result, in the same data type as (s), from (d).
- Specify the number of bits to be shifted in (n).

Ex.

Data type of (s): 16-bit data (WORD data type), Value input to (n): 8



- The (n) bit(s) from the most significant bit is/are filled with 0(s).
- Input a 16-bit data or 32-bit data (WORD or DWORD data type value) to (s).
- Input a value to (n) (Number of bits to be shifted) within the following range.

Data type of (s): 16-bit data (WORD data type)	Data type of (s): 32-bit data (DWORD data type)
Range: 0 to 15 The lower 4-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 6	Range: 0 to 31 The lower 5-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 22

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

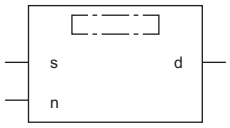
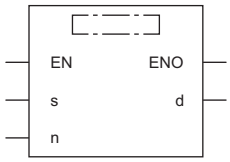
There is no operation error.

30.3 Rotating Data to the Left by n Bit(s)

ROL(_E)

These functions rotate the input value to the left by (n) bit(s), and output the operation result.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=ROL(s,n);
		[With EN/ENO]
		d:=ROL_E(EN,ENO,s,n);

Setting data

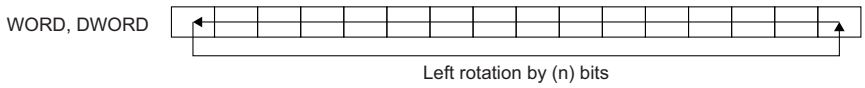
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_BIT
n (N)	Number of bits to be shifted	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

Processing details

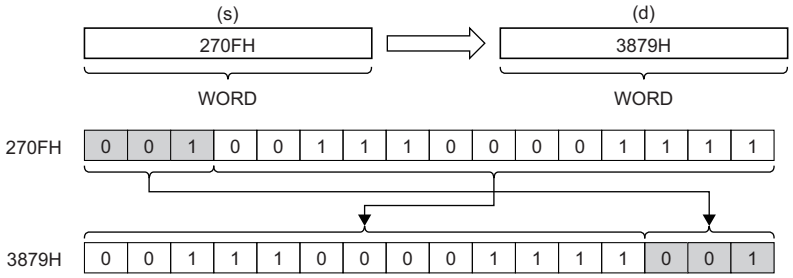
■Operation processing

- These functions rotate the 16-bit data or 32-bit data (WORD or DWORD data type value) input to (s) to the left by (n) bit(s), and output the operation result, in the same data type as (s), from (d).
- Specify the number of bits to be rotated in (n).



Ex.

Data type of (s): 16-bit data (WORD data type), Value input to (n): 3 (The data rotates to the left by 3 bits.)



- Input a 16-bit data or 32-bit data (WORD or DWORD data type value) to (s).
- Input a value to (n) (Number of bits to be shifted) within the following range.

Data type of (s): 16-bit data (WORD data type)	Data type of (s): 32-bit data (DWORD data type)
Range: 0 to 15 The lower 4-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 6	Range: 0 to 31 The lower 5-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 22

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

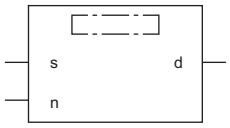
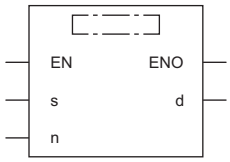
There is no operation error.

30.4 Rotating Data to the Right by n Bit(s)

ROR(_E)

These functions rotate the input value to the right by (n) bit(s), and output the operation result.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=ROR(s,n);
		[With EN/ENO]
		d:=ROR_E(EN,ENO,s,n);

Setting data

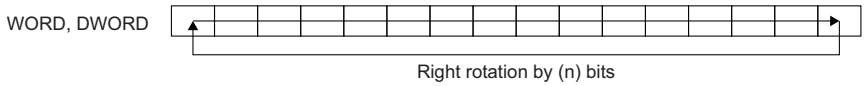
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_BIT
n (N)	Number of bits to be shifted	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

Processing details

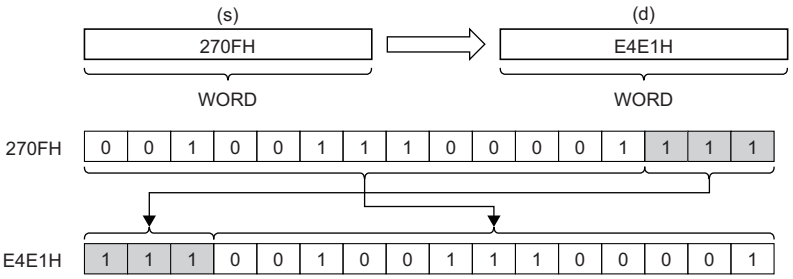
■Operation processing

- These functions rotate the 16-bit data or 32-bit data (WORD or DWORD data type value) input to (s) to the right by (n) bit(s), and output the operation result, in the same data type as (s), from (d).
- Specify the number of bits to be rotated in (n).



Ex.

Data type of (s): 16-bit data (WORD data type), Value input to (n): 3 (The data rotates to the right by 3 bits.)



- Input a 16-bit data or 32-bit data (WORD or DWORD data type value) to (s).
- Input a value to (n) (Number of bits to be shifted) within the following range.

Data type of (s): 16-bit data (WORD data type)	Data type of (s): 32-bit data (DWORD data type)
Range: 0 to 15 The lower 4-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 6	Range: 0 to 31 The lower 5-bit data is used. [Example] If the input value is 6: 6 If the input value is 22: 22

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

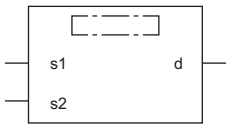
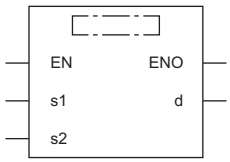
31 BOOLEAN FUNCTIONS

31.1 AND Operation, OR Operation, and XOR Operation

AND(_E), OR(_E), XOR(_E)

- AND(_E): These functions output the logical product of input values.
- OR(_E): These functions output the logical sum of input values.
- XOR(_E): These functions output the exclusive logical sum of input values.

Ladder, FBD/LD*1		ST*1

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		The function is described as an operator. (☞ Page 93 Operators)
		[With EN/ENO]
		d:=AND_E(EN,ENO,s1,s2);
		d:=OR_E(EN,ENO,s1,s2);
		d:=XOR_E(EN,ENO,s1,s2);

*1 The input variable s can be changed within the range from 2 to 28.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1) to s28 (IN28)	Input	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

Processing details

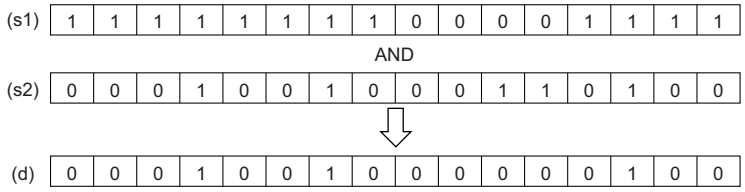
■Operation processing

1. AND(_E)

- These functions perform an AND operation (bit-by-bit) on the BOOL, WORD, or DWORD data type values input to (s1) to (s28), and output the operation result, in the same data type as (s), from (d).

Ex.

Data type: WORD

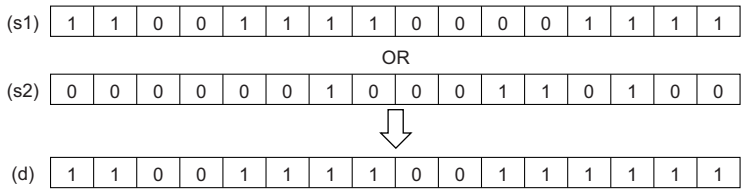


2. OR(_E)

- These functions perform an OR operation (bit-by-bit) on the BOOL, WORD, or DWORD data type values input to (s1) to (s28), and output the operation result, in the same data type as (s), from (d).

Ex.

Data type: WORD

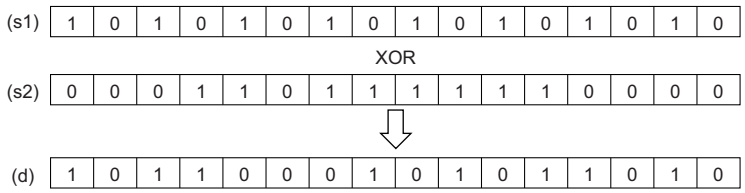


3. XOR(_E)

- These functions perform an XOR operation (bit-by-bit) on the BOOL, WORD, or DWORD data type values input to (s1) to (s28), and output the operation result, in the same data type as (s), from (d).

Ex.

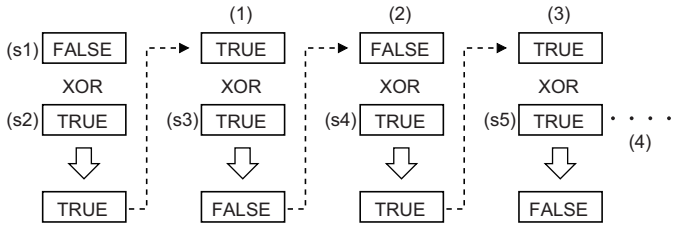
Data type: WORD



- If three or more (s) settings exist, (s3) will be XORed with the result of XOR between (s1) and (s2). In addition, if (s4) exists, (s4) will be XORed with the result of XOR for (s3). After this, XOR will repeat for the number of (s) settings.

Ex.

Data type: BOOL



(1) Number of s's: 3

(2) Number of s's: 4

(3) Number of s's: 5

(4) The XOR operation is repeated by the number of s's.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

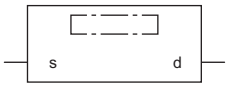
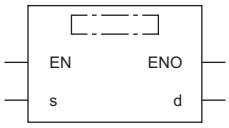
Operation error

There is no operation error.

31.2 NOT Operation

NOT(_E)

These functions output the logical NOT of input values.

Ladder, FBD/LD		ST
[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO] The function is described as an operator. (☞ Page 93 Operators) [With EN/ENO] d:=NOT_E(EN,ENO,s);
		

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s(IN)*1	Input	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT

*1 DX cannot be used.

Processing details

■Operation processing

- These functions perform a NOT operation (bit-by-bit) on the BOOL, WORD, or DWORD data type value input to (s), and output the operation result, in the same data type as (s), from (d).

Ex.

Data type: WORD

(s)	0	1	1	0	1	0	1	1	0	0	0	0	1	1	1	1
								NOT							
(d)	1	0	0	1	0	1	0	0	1	1	1	1	0	0	0	0

- Input a BOOL, WORD, or DWORD data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

There is no operation error.

32 SELECTION FUNCTIONS

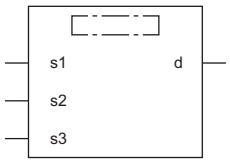
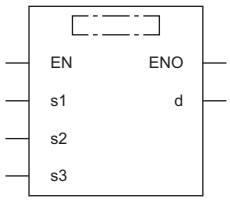
32.1 Selecting a Value

SEL(_E)

These functions output the selected input value.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=SEL(s1,s2,s3);
		[With EN/ENO]
		d:=SEL_E(EN,ENO,s1,s2,s3);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (G)	Output condition (TRUE: s3 output, FALSE: s2 output)	Input variable	BOOL
s2 (IN0)	Input	Input variable	ANY
s3 (IN1)			
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY

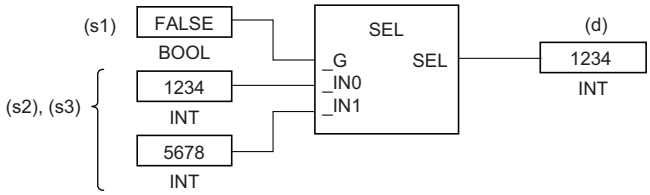
Processing details

■Operation processing

- These functions output either the (s2) or (s3) input value, in the same data type as (s2) or (s3), from (d) according to the value input to (s1).
- If the value input to (s1) is FALSE (=0), the (s2) input value is output from (d).
- If the value input to (s1) is TRUE (=1), the (s3) input value is output from (d).

Ex.

Data type of (s2) and (s3): INT (Argument names (s2) and (s3) correspond to the bit value (0 or 1) of (s1).)



- Input a BOOL data type value to (s1).
- Input a BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, TIME, structure, or array data type value to (s2) and (s3).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

• When (s2) and (s3) are of STRING data type

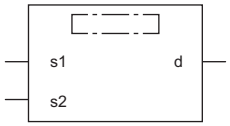
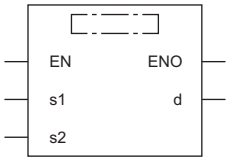
Error code (SD0)	Description
2820H	There is no NULL code (00H) in the label or device area (between the specified device number and the last device number) specified by (s2).
	There is no NULL code (00H) in the label or device area (between the specified device number and the last device number) specified by (s3).
3286H	The entire string cannot be stored in the label or device area (between the specified device number and the last device number) specified by (d). (The number of required points is insufficient.)

32.2 Selecting the Maximum/Minimum Value

MAX(_E), MIN(_E)

- MAX(_E): These functions output the maximum input value.
- MIN(_E): These functions output the minimum input value.

Ladder, FBD/LD*1		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=MAX(s1,s2);
		d:=MIN(s1,s2);
		[With EN/ENO]
		d:=MAX_E(EN,ENO,s1,s2);
		d:=MIN_E(EN,ENO,s1,s2);

*1 The input variable s can be changed within the range from 2 to 28.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1) to s28 (IN28)	Input	Input variable	ANY_ELEMENTARY
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_ELEMENTARY

Processing details

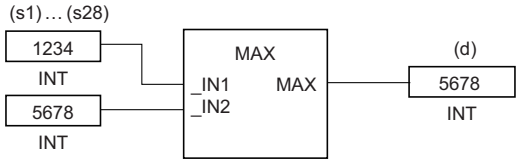
■Operation processing

• MAX(_E)

These functions output the maximum value of the BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, or TIME data type values input to (s1) to (s28), in the same data type as (s), from (d).

Ex.

Data type: INT

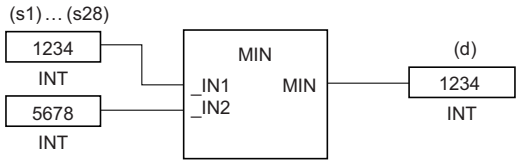


• MIN(_E)

These functions output the minimum value of the BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, or TIME data type values input to (s1) to (s28), in the same data type as (s), from (d).

Ex.

Data type: INT



• Input a BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, or TIME data type value to (s1) to (s28).

• Conditions for comparing the STRING data type values are as follows:

Match:	• All characters matched
Bigger string:	• The one having a character with a bigger code (when strings consist of different characters) • The one having a longer length (when strings are of different lengths)
Smaller string:	• The one having a character with a smaller code (when strings consist of different characters) • The one having a shorter length (when strings are of different lengths)

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

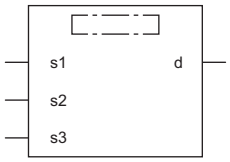
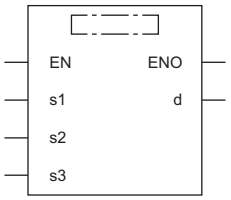
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s1) to (s28) and later.
3285H	The number of characters in the strings input to (s1) to (s28) exceeds 16383.
3286H	The entire string cannot be stored in each setting area in the device/label memory in the device specified by (d) and later. (The number of required points is insufficient.)

32.3 Controlling the Upper/Lower Limit

LIMIT(_E)

These functions output an input value that has been controlled in terms of the upper and lower limits.

Ladder, FBD/LD		ST

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=LIMIT(s1,s2,s3);
		[With EN/ENO]
		d:=LIMIT_E(EN,ENO,s1,s2,s3);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1(MN)*1	Lower limit value (minimum output threshold value)	Input variable	ANY_ELEMENTARY
s2(IN)*1	Input value to be controlled with the upper and lower limits	Input variable	ANY_ELEMENTARY
s3(MX)*1	Upper limit value (maximum output threshold value)	Input variable	ANY_ELEMENTARY
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_ELEMENTARY

*1 DX cannot be used.

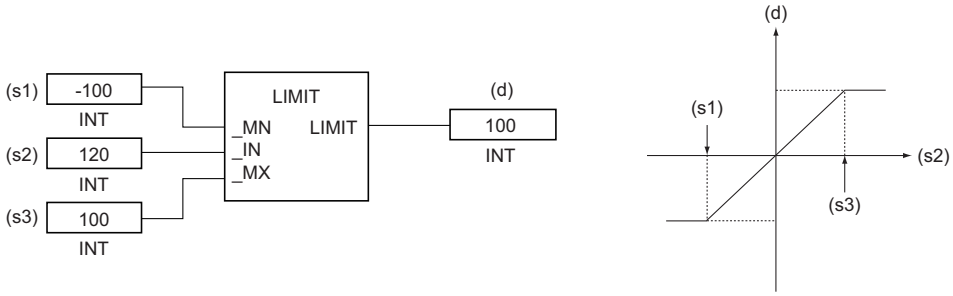
Processing details

■Operation processing

- These functions output the value, in the same data type as (s1), (s2), or (s3), from (d) according to the BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, or TIME data type value input to (s1), (s2), and (s3).
  - If the input values are (s2)>(s3), the value input to (s3) is output from (d).
  - If the input values are (s2)<(s1), the value input to (s1) is output from (d).
  - If the input values are (s1)≤(s2)≤(s3), the value input to (s2) is output from (d).

Ex.

Data type: INT



- Input a BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, or TIME data type value to (s1), (s2), and (s3), provided that the input value is (s1) < (s3).
- Conditions for comparing the STRING data type values are as follows:

Match:	• All characters matched
Bigger string:	• The one having a character with a bigger code (when strings consist of different characters) • The one having a longer length (when strings are of different lengths)
Smaller string:	• The one having a character with a smaller code (when strings consist of different characters) • The one having a shorter length (when strings are of different lengths)

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s1), (s2), and (s3) are of INT or WORD data type

Error code (SD0)	Description
3285H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

- When (s1), (s2), and (s3) are of DINT, DWORD, or TIME data type

Error code (SD0)	Description
3285H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

- When (s1), (s2), or (s3) are of BOOL data type

Error code (SD0)	Description
3285H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s3).

- When (s1), (s2), and (s3) are of REAL data type

Error code (SD0)	Description
3282H	The value input to (s1) is out of the following range: $-2^{128}<(s1)\leq-2^{-126}$, 0, $2^{-126}\leq(s1)<2^{128}$ (E-3.40282347+38 to E-1.17549435-38, 0, E1.17549435-38 to E3.40282347+38)
	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is out of the following range: $-2^{128}<(s2)\leq-2^{-126}$, 0, $2^{-126}\leq(s2)<2^{128}$ (E-3.40282347+38 to E-1.17549435-38, 0, E1.17549435-38 to E3.40282347+38)
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s3) is out of the following range: $-2^{128}<(s3)\leq-2^{-126}$, 0, $2^{-126}\leq(s3)<2^{128}$ (E-3.40282347+38 to E-1.17549435-38, 0, E1.17549435-38 to E3.40282347+38)
	The value input to (s3) is -0, a subnormal number, NaN (not a number), or ±∞.
3285H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s3).

- When (s1), (s2), and (s3) are of LREAL data type

Error code (SD0)	Description
3282H	The value input to (s1) is out of the following range: $-2^{1024}<(s1)\leq-2^{-1022}$, 0, $2^{-1022}\leq(s1)<2^{1024}$ (E-1.7976931348623157+308 to E-2.2250738585072014-308, 0, E2.2250738585072014-308 to E1.7976931348623157+308)
	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is out of the following range: $-2^{1024}<(s2)\leq-2^{-1022}$, 0, $2^{-1022}\leq(s2)<2^{1024}$ (E-1.7976931348623157+308 to E-2.2250738585072014-308, 0, E2.2250738585072014-308 to E1.7976931348623157+308)
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s3) is out of the following range: $-2^{1024}<(s3)\leq-2^{-1022}$, 0, $2^{-1022}\leq(s3)<2^{1024}$ (E-1.7976931348623157+308 to E-2.2250738585072014-308, 0, E2.2250738585072014-308 to E1.7976931348623157+308)
	The value input to (s3) is -0, a subnormal number, NaN (not a number), or ±∞.
3285H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s3).

- When (s1), (s2), and (s3) are of STRING data type

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the label or device area (between the specified device number and the last device number) specified by (s1), (s2), or (s3).
3285H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s3).
	The number of characters in the strings input to (s1), (s2), and (s3) exceeds 16383.
3286H	The entire string cannot be stored in the label or device area (between the specified device number and the last device number) specified by (d). (The number of required points is insufficient.)

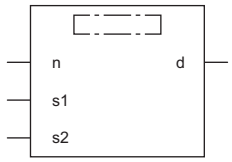
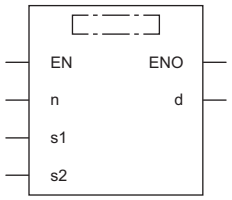
32.4 Multiplexer

MUX(_E)

These functions output one of the input values.

Ladder, FBD/LD*1		ST*1

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=MUX(n,s1,s2);
		[With EN/ENO]
		d:=MUX_E(EN,ENO,n,s1,s2);

*1 The input variable s can be changed within the range from 2 to 28.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
n(K)	Output value selection	Input variable	ANY_INT
s1 (IN0) to s28 (IN27)	Input	Input variable	ANY
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY

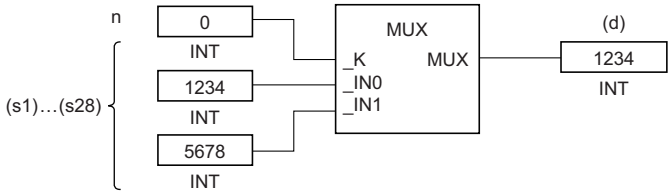
Processing details

■Operation processing

- These functions output one of the values input to (s1) to (s28), in the same data type as (s), from (d) according to the value input to (n).
- If the (n) input value is 0, the value input to (s1) is output from (d).
- If the (n) input value is (n)-1, the value input to (sn) is output from (d).

Ex.

Data type: INT



- If a value outside the range of the number of pins in (s) is input to (n), an undefined value is output from (d). (No operation error occurs. When MUX_E is used, ENO outputs FALSE.)
- Input an INT, DINT, WORD, DWORD data type value to (n) within the range of 0 to 27, provided that it is within the range of the number of pins in (s).
- Input a BOOL, INT, DINT, WORD, DWORD, REAL, LREAL, STRING, TIME, structure, or array data type value to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s1) to (s28) and later.
3285H	The number of characters in the strings input to (s1) to (s28) exceeds 16383.
3286H	The entire string cannot be stored in each setting area in the device/label memory in the device specified by (d) and later. (The number of required points is insufficient.)

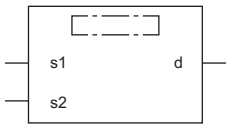
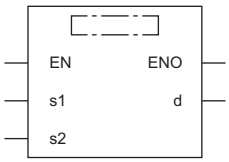
33 COMPARISON FUNCTIONS

33.1 Comparing Data

GT(_E), GE(_E), EQ(_E), LE(_E), LT(_E)

These functions output the comparison result of input values.

Ladder, FBD/LD*1		ST*1

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=GT(s1,s2);
		d:=GE(s1,s2);
		d:=EQ(s1,s2);
		d:=LE(s1,s2);
		d:=LT(s1,s2);
		[With EN/ENO]
		d:=GT_E(EN,ENO,s1,s2);
		d:=GE_E(EN,ENO,s1,s2);
		d:=EQ_E(EN,ENO,s1,s2);
		d:=LE_E(EN,ENO,s1,s2);
		d:=LT_E(EN,ENO,s1,s2);

*1 The input variable s can be changed within the range from 2 to 28.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1) to s28 (IN28)*1	Input	Input variable	ANY_ELEMENTARY
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output (TRUE, FALSE)	Output variable	BOOL

*1 DX cannot be used.

Processing details

■Operation processing

- These functions perform comparison operation between the input values to (s), and output the operation result, in the BOOL data type, from (d).
  - GT(_E): Performs comparison of [(s1)>(s2)]&[(s2)>(s3)]&···&[$(s)_{(n-1)}$>$(s)_{(n)}$].
    - If all values satisfy $(s)_{(n-1)}$>$(s)_{(n)}$, TRUE is output.
    - If one of the values satisfies $(s)_{(n-1)}$≤$(s)_{(n)}$, FALSE is output.
  - GE(_E): Performs comparison of [(s1)≥(s2)]&[(s2)≥(s3)]&···&[$(s)_{(n-1)}$≥$(s)_{(n)}$].
    - If all values satisfy $(s)_{(n-1)}$≥$(s)_{(n)}$, TRUE is output.
    - If one of the values satisfies $(s)_{(n-1)}$<$(s)_{(n)}$, FALSE is output.
  - EQ(_E): Performs comparison of [(s1)=(s2)]&[(s2)=(s3)]&···&[$(s)_{(n-1)}$=$(s)_{(n)}$].
    - If all values satisfy $(s)_{(n-1)}$=$(s)_{(n)}$, TRUE is output.
    - If one of the values satisfies $(s)_{(n-1)}$≠$(s)_{(n)}$, FALSE is output.
  - LE(_E): Performs comparison of [(s1)≤(s2)]&[(s2)≤(s3)]&···&[$(s)_{(n-1)}$≤$(s)_{(n)}$].
    - If all values satisfy $(s)_{(n-1)}$≤$(s)_{(n)}$, TRUE is output.
    - If one of the values satisfies $(s)_{(n-1)}$>$(s)_{(n)}$, FALSE is output.
  - LT(_E): Performs comparison of [(s1)<(s2)]&[(s2)<(s3)]&···&[$(s)_{(n-1)}$<$(s)_{(n)}$].
    - If all values satisfy $(s)_{(n-1)}$<$(s)_{(n)}$, TRUE is output.
    - If one of the values satisfies $(s)_{(n-1)}$≥$(s)_{(n)}$, FALSE is output.
- Input an INT, DINT, REAL, LREAL, BOOL, WORD, DWORD, TIME, or STRING type data value to (s). No WSTRING type Unicode string can be specified.
- Conditions for comparing the STRING data type values are as follows:

Match:	• All characters matched
Bigger string:	• The one having a character with a bigger code (when strings consist of different characters) • The one having a longer length (when strings are of different lengths)
Smaller string:	• The one having a character with a smaller code (when strings consist of different characters) • The one having a shorter length (when strings are of different lengths)

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

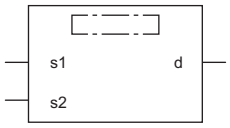
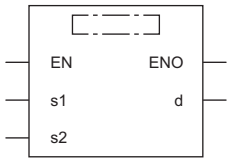
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s1) to (s28) and later.
3285H	The number of characters in the strings input to (s1) to (s28) exceeds 16383.
3286H	The entire string cannot be stored in each setting area in the device/label memory in the device specified by (d) and later. (The number of required points is insufficient.)

33.2 Comparing Data

NE(_E)

These functions output the comparison result of input values.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=NE(s1,s2);
		[With EN/ENO]
		d:=NE_E(EN,ENO,s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1, s2	Input	Input variable	ANY_ELEMENTARY
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output (TRUE, FALSE)	Output variable	BOOL

Processing details

■Operation processing

- These functions perform comparison operation between the input values to (s), and output the operation result, in the BOOL data type, from (d).
  - NE(_E): Performs comparison of [(s1)≠(s2)].
    - If (s1)≠(s2), TRUE is output.
    - If (s1)=(s2), FALSE is output.
- Input an INT, DINT, REAL, LREAL, BOOL, WORD, DWORD, TIME, or STRING type data value to (s). No WSTRING type Unicode string can be specified.
- Conditions for comparing the STRING data type values are as follows:

Match:	• All characters matched
Bigger string:	• The one having a character with a bigger code (when strings consist of different characters) • The one having a longer length (when strings are of different lengths)
Smaller string:	• The one having a character with a smaller code (when strings consist of different characters) • The one having a shorter length (when strings are of different lengths)

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s) and later.
3285H	The number of characters in the string input to (s) exceeds 16383.
3286H	The entire string cannot be stored in each setting area in the device/label memory in the device specified by (d) and later. (The number of required points is insufficient.)

34 STRING FUNCTIONS

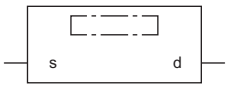
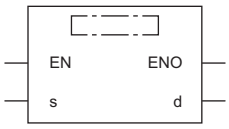
34.1 Detecting a String Length

LEN(_E)

These functions detect and output the length of the string input.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=LEN(s);
		[With EN/ENO]
		d:=LEN_E(EN,ENO,s);

Setting data

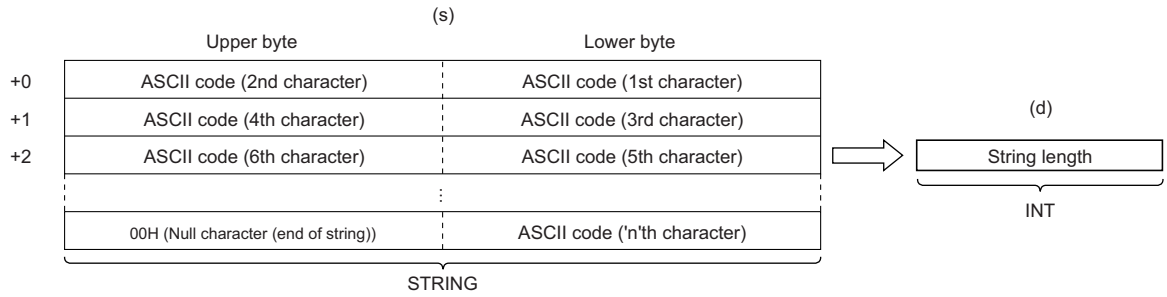
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANYSTRING_SINGLE
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions detect the length of the string input to (s), and output the length from (d).



- Input a STRING data type value to (s) within the range of 0 to 255 bytes.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

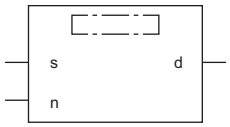
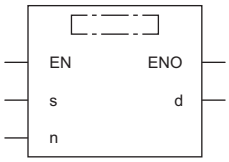
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s) and later.
3285H	The number of characters in the string input to (s) exceeds 16383.

34.2 Extracting String Data From the Left/Right

LEFT(_E), RIGHT(_E)

- LEFT(_E): These functions extract and output the specified number of characters, starting from the left end of the string input.
- RIGHT(_E): These functions extract and output the specified number of characters, starting from the right end of the string input.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=LEFT(s,n);
		d:=RIGHT(s,n);
		[With EN/ENO]
		d:=LEFT_E(EN,ENO,s,n);
		d:=RIGHT_E(EN,ENO,s,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANYSTRING_SINGLE
n (L)	Number of characters to be extracted	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANYSTRING_SINGLE

Processing details

■Operation processing

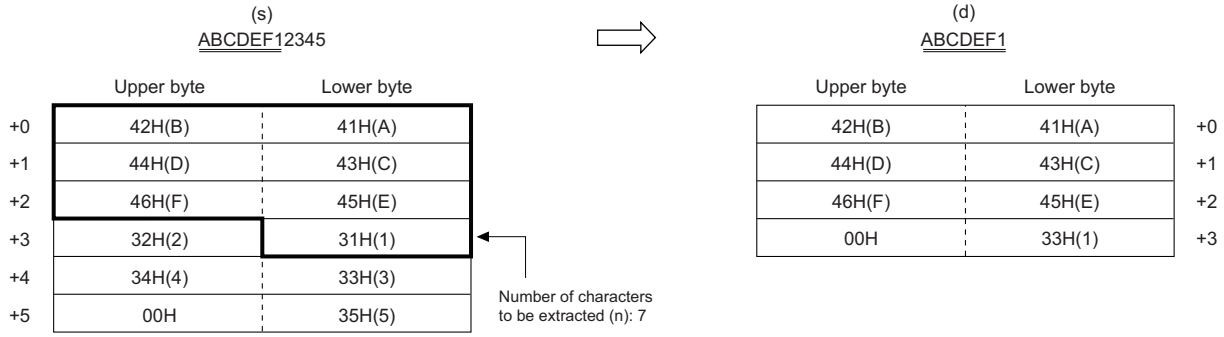
• LEFT(_E)

These functions extract the specified number of characters, starting from the left end of the string input to (s), and output the operation result from (d).

Specify the number of characters to be extracted in (n).

Ex.

When (n)=7



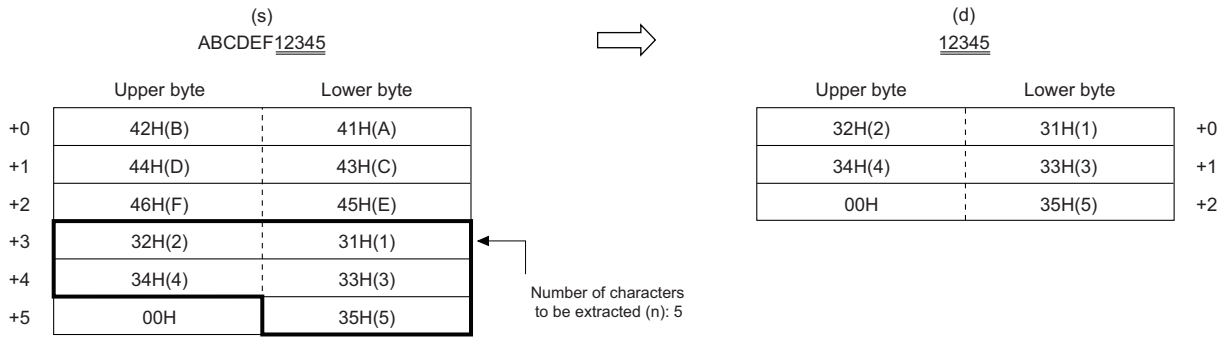
• RIGHT(_E)

These functions extract the specified number of characters, starting from the right end of the string input to (s), and output the operation result from (d).

Specify the number of characters to be extracted in (n).

Ex.

When (n)=5



- Input a STRING data type value to (s) within the range of 0 to 255 bytes.
- Input an INT data type value to (n) within the range of 0 to 255, provided that it is within the number of characters in the string input to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s) and later.
3285H	The number of characters in the string input to (s) exceeds 16383.
	The number of characters in the string input to (s) is 0.
	The value input to (n) exceeds the number of characters in (s).

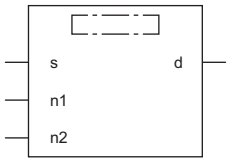
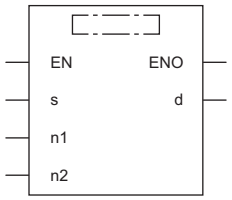
34.3 Extracting String Data

MID(_E)

These functions extract and output the specified number of characters, starting from the specified position of the string input.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=MID(s,n1,n2);
		[With EN/ENO]
		d:=MID_E(EN,ENO,s,n1,n2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANYSTRING_SINGLE
n1 (L)	Number of characters to be extracted	Input variable	INT
n2 (P)	Extraction target character start position	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANYSTRING_SINGLE

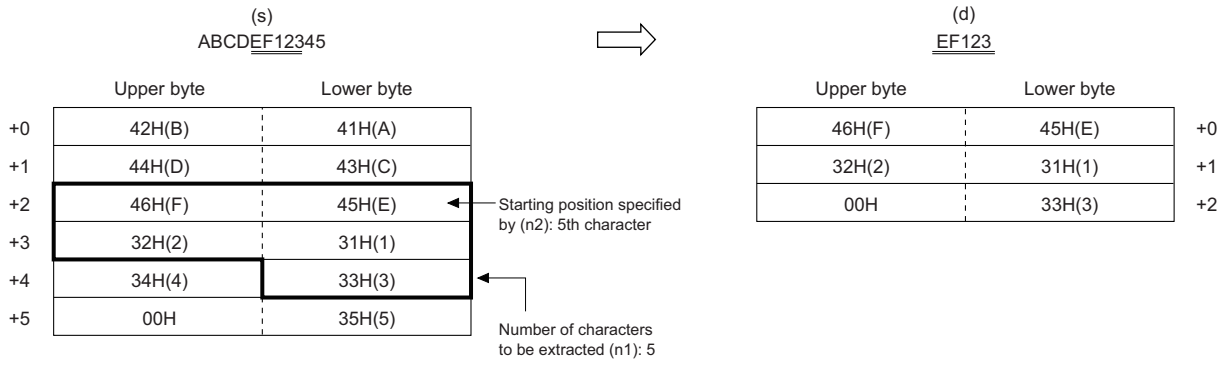
Processing details

■Operation processing

- These functions extract the specified number of characters, starting from the specified position of the string input to (s), and output the operation result from (d).
- Specify the number of characters to be extracted in (n1).
- Specify the start position of the string to be extracted in (n2).

Ex.

When (n1)=5, (n2)=5



- Input a STRING data type value to (s) within the range of 0 to 255 bytes.
- Input an INT data type value to (n1) within the range of 0 to 255, provided that it is within the number of characters in the string input to (s).
- Input an INT data type value to (n2) within the range of 1 to 255, provided that it is within the number of characters in the string input to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s) and later.
3285H	The number of characters in the string input to (s) exceeds 16383.
	Out-of-range data is set to (n1) or (n2). • The value input to (n1) or (n2) is 0 or smaller. • The value input to (n2) is other than the valid values (-1, 0, 1 or bigger). • The value input to (n1) exceeds the number of characters in (s). • The sum of (n1) and (n2) exceeds the number of characters in (s).

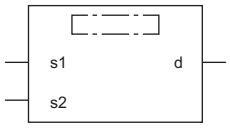
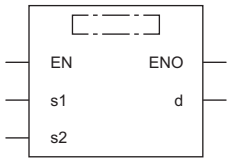
34.4 Concatenating String Data

CONCAT(_E)

These functions concatenate strings, and output the operation result.

Ladder, FBD/LD*1		ST*1

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=CONCAT(s1,s2);
		[With EN/ENO]
		d:=CONCAT_E(EN,ENO,s1,s2);

*1 The input variable s can be changed within the range from 2 to 28.

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1) to s28 (IN28)	Input	Input variable	ANYSTRING_SINGLE
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANYSTRING_SINGLE

Processing details

■Operation processing

- These functions concatenate the strings input to (s2) to (s28) to the end of the string input to (s1), and output the operation result from (d).
- The (s2) to (s28) strings are concatenated successively, ignoring 00H, which indicates the end of the (s1) string.
- If the string after concatenation exceeds 255 bytes, the substring up to the 255th byte will be output.

(s1) ABCDE

	Upper byte	Lower byte
+0	42H(B)	41H(A)
+1	44H(D)	43H(C)
+2	00H	45H(E)

+

(s2) 123456

	Upper byte	Lower byte
+0	32H(2)	31H(1)
+1	34H(4)	33H(3)
+2	36H(6)	35H(5)
+3	00H	

⇨

(d) ABCDE123456

	Upper byte	Lower byte
+0	42H(B)	41H(A)
+1	44H(D)	43H(C)
+2	31H(1)	45H(E)
+3	33H(3)	32H(2)
+4	35H(5)	34H(4)
+5	00H	36H(6)

- Input a STRING data type value to (s1) and (s2) to (s28) within the range of 0 to 255 bytes.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s1) to (s28) and later.
	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (d) and later.
2821H	The device numbers are overlapping between (s1) to (s28) and (d).
3285H	The number of characters in the strings input to (s1) to (s28) exceeds 16383.
	The number of characters in the strings input to (s1) to (s28) is 0.
	The number of characters of the string in the device specified by (d) exceeds 16383.
3286H	The entire string after concatenate processing cannot be stored in each setting area in the device/label memory in the device specified by (d) and later. (The number of required points is insufficient.)

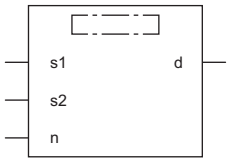
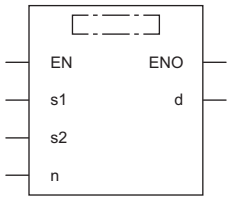
34.5 Inserting String Data

INSERT(_E)

These functions insert a string into another string, and output the operation result.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=INSERT(s1,s2,n);
		[With EN/ENO]
		d:=INSERT_E(EN,ENO,s1,s2,n);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1), s2 (IN2)	Input	Input variable	ANYSTRING_SINGLE
n (P)	Insertion target character start position	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANYSTRING_SINGLE

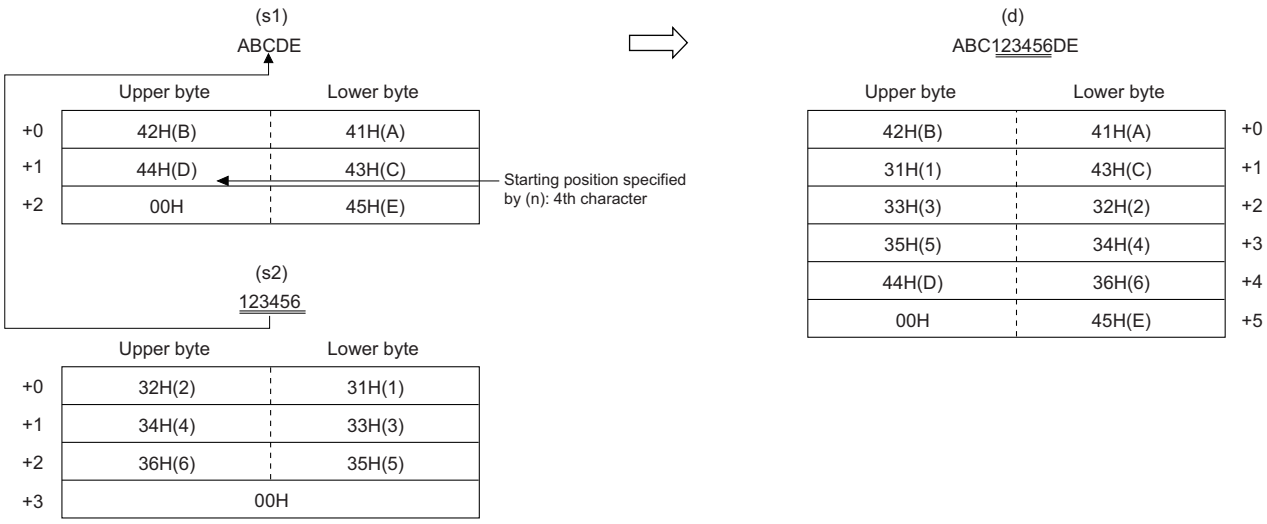
Processing details

■Operation processing

- These functions insert the string input to (s2) into the insertion start position, i.e. the 'n'th character position from the beginning of the string input to (s1), and output the operation result from (d).
- After the (s2) string is inserted into the (s1) string, 00H, which indicates the end of the (s2) string, is ignored.
- If the string after insertion exceeds 255 bytes, the substring up to the 255th byte will be output.

Ex.

When (n)=4



- Input a STRING data type value to (s1) and (s2) within the range of 0 to 255 bytes.
- Input an INT data type value to (n) within the range of 1 to 255, provided that it is within the number of characters in the string input to (s1).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s1) to (s28) and later.
	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (d) and later.
2821H	The device numbers are overlapping between (s1) to (s28) and (d).
3285H	The number of characters in the strings input to (s1) to (s28) exceeds 16383.
	The number of characters in the strings input to (s1) to (s28) is 0.
	The number of characters of the string in the device specified by (d) exceeds 16383.
3286H	The entire string after concatenate processing cannot be stored in each setting area in the device/label memory in the device specified by (d) and later. (The number of required points is insufficient.)

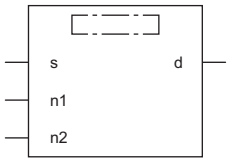
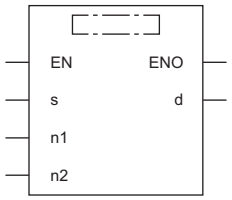
34.6 Deleting String Data

DELETE(_E)

These functions delete the specified range in a string, and output the operation result.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=DELETE(s,n1,n2);
		[With EN/ENO]
		d:=DELETE_E(EN,ENO,s,n1,n2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANYSTRING_SINGLE
n1 (L)	Number of characters to be deleted	Input variable	INT
n2 (P)	Deletion target character start position	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANYSTRING_SINGLE

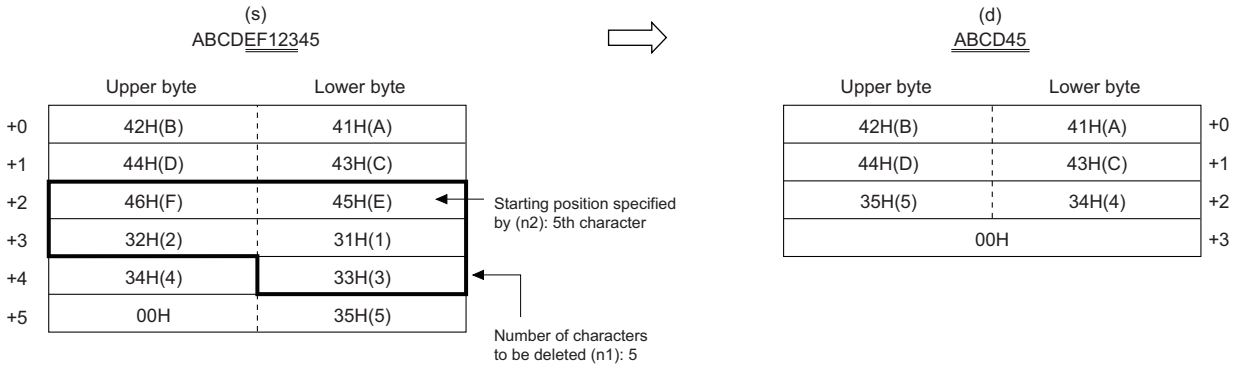
Processing details

■Operation processing

- These functions delete the specified number of characters, starting from the desired position of the string input to (s), and output the remaining substring from (d).
- Specify the number of characters to be deleted in (n1).
- Specify the start position of the string to be deleted in (n2).

Ex.

When (n1)=5, (n2)=5



- Input a STRING data type value to (s) within the range of 0 to 255 bytes.
- Input an INT data type value to (n1) within the range of 0 to 255, provided that it is within the number of characters in the string input to (s).
- Input an INT data type value to (n2) within the range of 1 to 255, provided that it is within the number of characters in the string input to (s).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the label or device area (between the specified device number and the last device number) specified by (s).
3285H	The number of characters in the string input to (s) exceeds 255.
	The value input to (n1) is out of the range, 0 to 255.
	The value input to (n2) is out of the range, 1 to 255.
	The value input to (n1) exceeds the number of characters in (s).
	The value input to (n2) exceeds the number of characters in (s).
3286H	The entire string after delete processing cannot be stored in the label or device area (between the specified device number and the last device number) specified by (d).

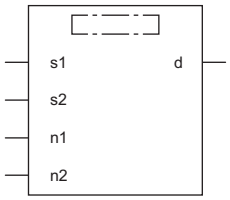
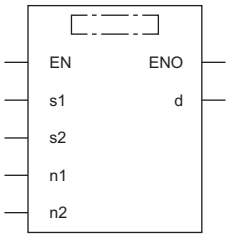
34.7 Replacing String Data

REPLACE(_E)

These functions replace the specified range in a string, and output the operation result.

Ladder, FBD/LD		ST

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO]
		d:=REPLACE(s1,s2,n1,n2);
		[With EN/ENO]
		d:=REPLACE_E(EN,ENO,s1,s2,n1,n2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1), s2 (IN2)	Input	Input variable	ANYSTRING_SINGLE
n1 (L)	Number of characters to be replaced	Input variable	INT
n2 (P)	Replacement target character start position	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANYSTRING_SINGLE

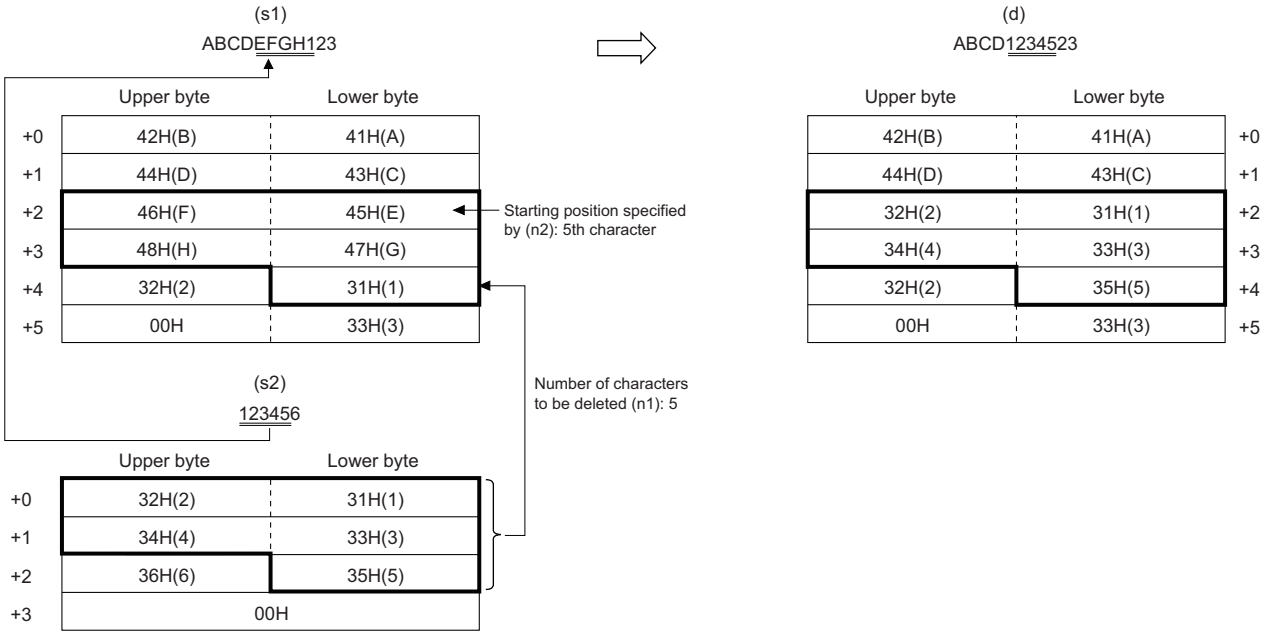
Processing details

■Operation processing

- These functions replace the specified number of characters starting from the desired position of the string input to (s1) with the string input to (s2), and output the operation result from (d).
- Specify the number of characters to be replaced in (n1).
- Specify the start position of the string to be replaced in (n2).

Ex.

When (n1)=5, (n2)=5



- Input a STRING data type value to (s1) and (s2) within the range of 0 to 255 bytes.
- Input an INT data type value to (n1) within the range of 0 to 255, provided that it is within the number of characters in the string input to (s1).
- Input an INT data type value to (n2) within the range of 1 to 255, provided that it is within the number of characters in the string input to (s1).

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

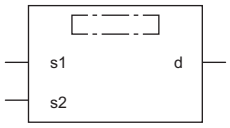
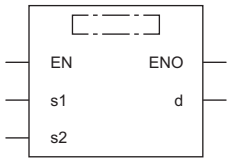
Error code (SD0)	Description
2820H	There is no NULL code (00H) in the label or device area (between the specified device number and the last device number) specified by (s1).
	There is no NULL code (00H) in the label or device area (between the specified device number and the last device number) specified by (s2).
3285H	The number of characters in the string input to (s1) exceeds 255.
	The number of characters in the string input to (s2) exceeds 255.
	The value input to (n1) is out of the range, 0 to 255.
	The value input to (n2) is out of the range, 1 to 255.
	The value input to (n1) exceeds the number of characters in (s2).
	The value input to (n2) exceeds the number of characters in (s1).
3286H	The entire string after delete processing cannot be stored in the label or device area (between the specified device number and the last device number) specified by (d).

34.8 Searching String Data

FIND(_E)

These functions search a string, and output the operation result.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=FIND(s1,s2);
		[With EN/ENO]
		d:=FIND_E(EN,ENO,s1,s2);

Setting data

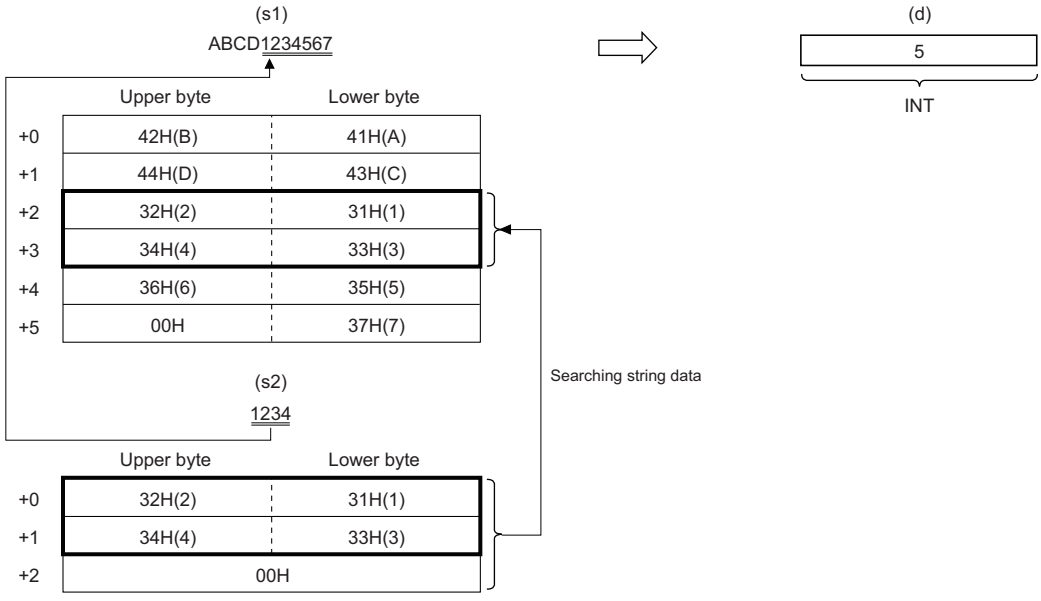
■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1), s2 (IN2)	Input	Input variable	ANYSTRING_SINGLE
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	INT

Processing details

■Operation processing

- These functions search the string input to (s2) from the beginning of the string input to (s1), and output the search result from (d).
- The start character position of the first string found is output as the search result.
- If the (s2) string is not found in the (s1) string, 0 will be output.



- Input a STRING data type value to (s1) and (s2) within the range of 0 to 255 bytes.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s1) and later.
	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s2) and later.
3285H	The number of characters of the string in the device specified by (s1) exceeds 16383.
	The number of characters in the string input to (s1) is 0.
	The number of characters in the string specified by (s2) exceeds 16383.
	The number of characters in the string input to (s2) is 0.

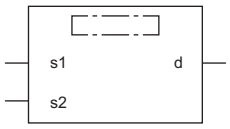
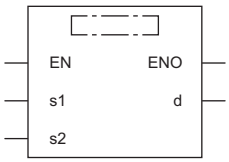
35 TIME DATA TYPE FUNCTIONS

35.1 Addition

ADD_TIME(_E)

These functions output the sum ((s1)+(s2)) of the TIME data type input values.

Ladder, FBD/LD		ST

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=ADD_TIME(s1,s2);
		[With EN/ENO]
		d:=ADD_TIME_E(EN,ENO,s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1), s2 (IN2)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

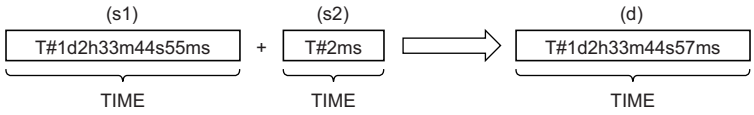
Processing details

■Operation processing

- These functions perform addition of the TIME data type values input to (s1) and (s2) ((s1)+(s2)), and output the operation result, in the TIME data type, from (d).

Ex.

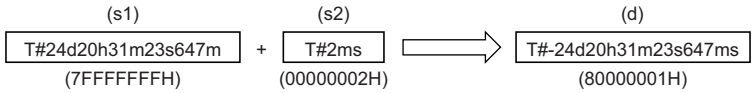
When (s1)=T#1d2h33m44s55ms (1 day, 2 hours, 33 minutes, 44 seconds, 55 milliseconds) and (s2)=T#2ms (2 milliseconds)



- Input a TIME data type value to (s1) and (s2).
- Even if an underflow or overflow occurs in the operation result, no operation error is issued. The following is output to (d). When ADD_TIME_E is used, ENO outputs TRUE.

Ex.

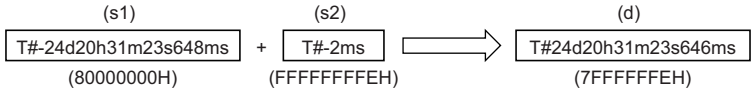
Overflow



A negative time value results because the most significant bit is 1.

Ex.

Underflow



A positive time value results because the most significant bit is 0.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

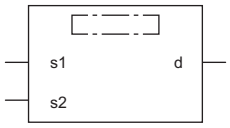
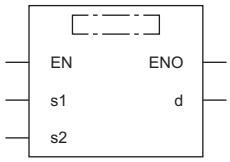
There is no operation error.

35.2 Subtraction

SUB_TIME(_E)

These functions output the difference ((s1)-(s2)) between the TIME data type input values.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=SUB_TIME(s1,s2);
		[With EN/ENO]
		d:=SUB_TIME_E(EN,ENO,s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1), s2 (IN2)	Input	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

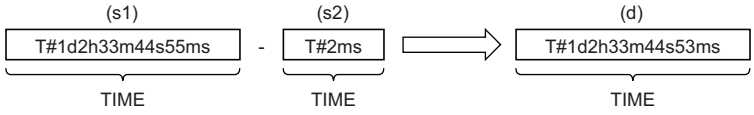
Processing details

■Operation processing

- These functions perform subtraction between the TIME data type values input to (s1) and (s2) ((s1)-(s2)), and output the operation result, in the TIME data type, from (d).

Ex.

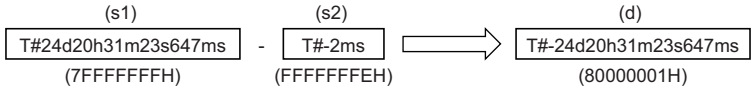
When (s1)=T#1d2h33m44s55ms (1 day, 2 hours, 33 minutes, 44 seconds, 55 milliseconds) and (s2)=T#2ms (2 milliseconds)



- Input a TIME data type value to (s1) and (s2).
- Even if an underflow or overflow occurs in the operation result, no operation error is issued. The following is output to (d). When SUB_TIME_E is used, ENO outputs TRUE.

Ex.

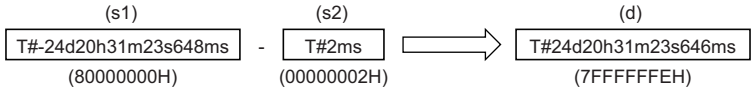
Overflow



A negative time value results because the most significant bit is 1.

Ex.

Underflow



A positive time value results because the most significant bit is 0.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

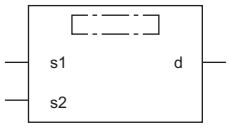
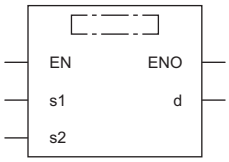
There is no operation error.

35.3 Multiplication

MUL_TIME(_E)

These functions output the product ((s1)×(s2)) of the TIME data type input values.

Ladder, FBD/LD		ST

[Without EN/ENO]

 	
	[With EN/ENO]
	
	
		[Without EN/ENO]
		d:=MUL_TIME(s1,s2);
		[With EN/ENO]
		d:=MUL_TIME_E(EN,ENO,s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1)	Input	Input variable	TIME
s2 (IN2)	Input	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

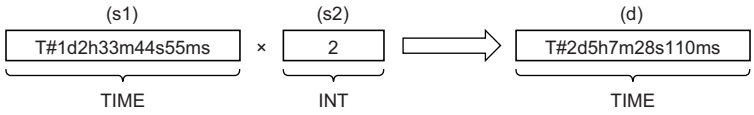
Processing details

■Operation processing

- These functions perform multiplication between the TIME data type values input to (s1) and (s2) ((s1)×(s2)), and output the operation result, in the TIME data type, from (d).

Ex.

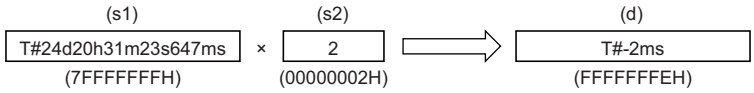
When (s1)=T#1d2h33m44s55ms (1 day, 2 hours, 33 minutes, 44 seconds, 55 milliseconds) and (s2)=2



- Input a TIME data type value to (s1).
- Input an INT, DINT, REAL, or LREAL data type value to (s2).
- Even if an underflow or overflow occurs in the operation result, no operation error is issued. The following is output to (d). When MUL_TIME_E is used, ENO outputs TRUE. (In this case, the output value is of TIME data type with the upper 32 bits deleted although the operation result is 64-bit data.)

Ex.

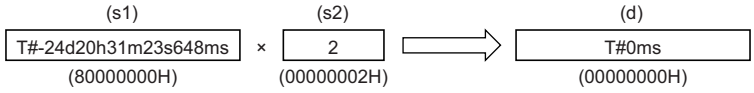
Overflow



A negative time value results because the most significant bit is 1.

Ex.

Underflow



A positive time value results because the most significant bit is 0.

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

- When (s2) is of LREAL data type

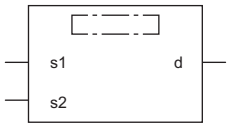
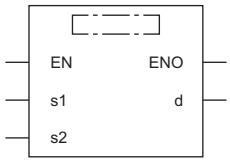
Error code (SD0)	Description
3282H	The value input to (s2) is out of the following range: $-2^{1024}<(s2)\leq-2^{-1022}$, 0, $2^{-1022}\leq(s2)<2^{1024}$ (E-1.7976931348623157+308 to E-2.2250738585072014-308, 0, E2.2250738585072014-308 to E1.7976931348623157+308)
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3285H	The single-precision real number input to (s2) is out of the range, -2147483648 to 2147483647.

35.4 Division

DIV_TIME(_E)

These functions output the quotient ((s1)÷(s2)) of the TIME data type input values.

Ladder, FBD/LD		ST

[Without EN/ENO]

	[With EN/ENO]
	
		[Without EN/ENO]
		d:=DIV_TIME(s1,s2);
		[With EN/ENO]
		d:=DIV_TIME_E(EN,ENO,s1,s2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (IN1)	Input	Input variable	TIME
s2 (IN2)	Input	Input variable	ANY_NUM
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

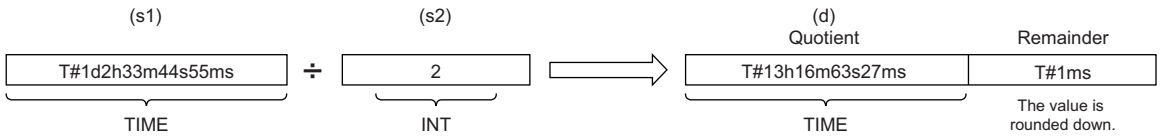
Processing details

■Operation processing

- These functions perform division between the TIME data type values input to (s1) and (s2) ((s1)÷(s2)), and output the operation result, in the TIME data type, from (d). The remainder is rounded down.

Ex.

When (s1)=T#1d2h33m44s55ms (1 day, 2 hours, 33 minutes, 44 seconds, 55 milliseconds) and (s2)=2



- Input a TIME data type value to (s1).
- Input an INT, DINT, REAL, or LREAL data type value to (s2). (Note that the value input to (s2) shall be other than 0.)

■Operation result

1. Function without EN/ENO

The operation processing is performed. The operation result is output from (d).

2. Function with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	ENO	(d)
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE*1	Undefined value

*1 If the value FALSE is output from ENO, the output data from (d) will be undefined. Create a program so that the undefined value will not be used in operations.

Operation error

Error code (SD0)	Description
3280H	The value input to (s2) is 0. (The value is divided by zero.)

- When (s2) is of LREAL data type

Error code (SD0)	Description
3282H	The value input to (s2) is out of the following range: $-2^{1024}<(s2)\leq-2^{-1022}$, 0, $2^{-1022}\leq(s2)<2^{1024}$ (E-1.7976931348623157+308 to E-2.2250738585072014-308, 0, E2.2250738585072014-308 to E1.7976931348623157+308)
3285H	The data input to (s2) is out of the range, -2147483648 to 2147483647.

PART 7 STANDARD FUNCTION BLOCKS

This part consists of the following chapters.

36 BISTABLE FUNCTION BLOCKS

37 EDGE DETECTION FUNCTION BLOCKS

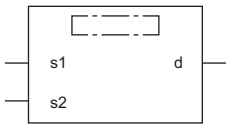
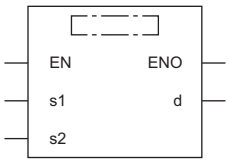
38 COUNTER/TIMER FUNCTION BLOCKS

36 BISTABLE FUNCTION BLOCKS

36.1 Bistable Function Block (Set-Dominant)

SR(_E)

These function blocks discriminate between two input values, and output 1 (TRUE) or 0 (FALSE).

Ladder, FBD/LD		ST
[Without EN/ENO]  	[With EN/ENO]  	[Without EN/ENO] Instance name(S1:=s1,R:=s2,Q1:=d); [With EN/ENO] Instance name(EN:=en,ENO:=eno,S1:=s1,R:=s2,Q1:=d);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (S1)	Set command	Input variable	BOOL
s2 (R)	Reset command	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d (Q1)	Output	Output variable	BOOL

Processing details

■Operation processing

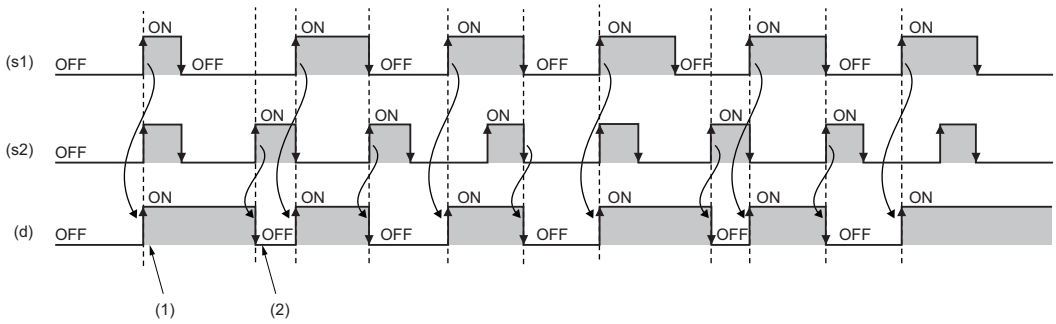
- When (s1) turns on, (d) is set. Turning on (s2) while (s1) is off resets (d).
- Even when (s2) turns on while (s1) is on, (d) is not reset.

■Operation result

1. Function block without EN/ENO

The operation processing is performed. The operation result is output from (d).

- Timing chart



(1) When (s1) turns on, (d) turns on.

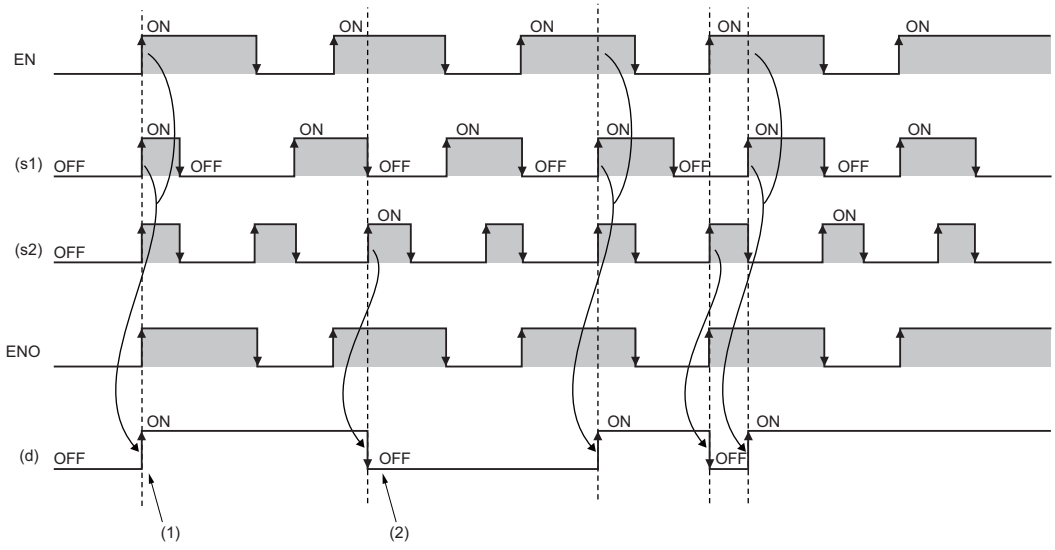
(2) When (s2) turns on while (s1) is off, (d) turns off.

2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE	Previous output value

- Timing chart



(1) When (s1) turns on while EN is on, (d) turns on.

(2) When (s2) turns on while EN is on and (s1) is off, (d) turns off.

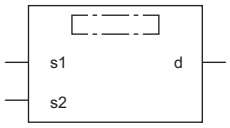
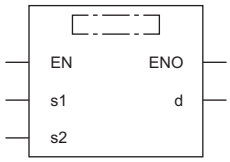
Operation error

There is no operation error.

36.2 Bistable Function Block (Reset-Dominant)

RS(_E)

These function blocks discriminate between two input values, and output 1 (TRUE) or 0 (FALSE).

Ladder, FBD/LD		ST
[Without EN/ENO]  	[With EN/ENO]  	[Without EN/ENO] Instance name(S:=s1,R1:=s2,Q1:=d); [With EN/ENO] Instance name(EN:=en,ENO:=eno,S:=s1,R1:=s2,Q1:=d);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (S)	Set command	Input variable	BOOL
s2 (R1)	Reset command	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d (Q1)	Output	Output variable	BOOL

Processing details

■Operation processing

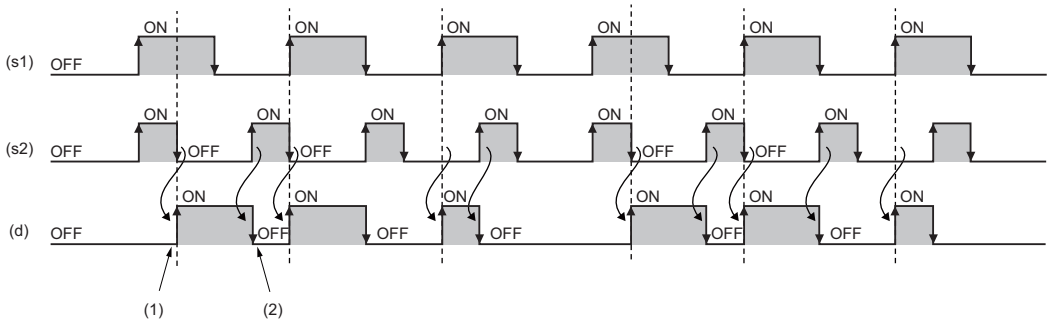
- When (s1) turns on, (d) is set. When (s2) turns on, (d) is reset.
- Even when (s1) turns on while (s2) is on, (d) is not set.

■Operation result

1. Function block without EN/ENO

The operation processing is performed. The operation result is output from (d).

- Timing chart



(1) When (s2) turns off while (s1) is on, (d) turns on.

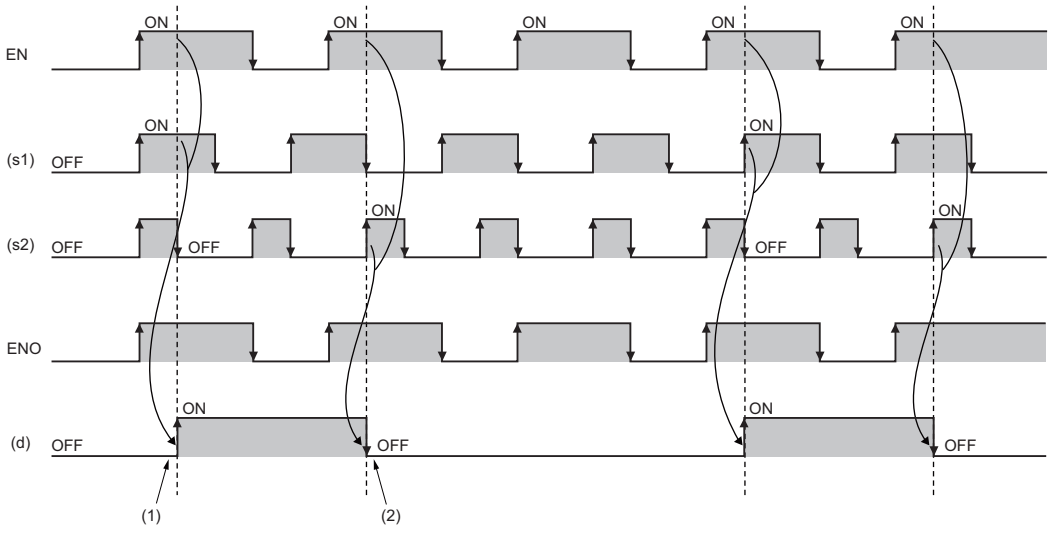
(2) When (s2) turns on, (d) turns off.

2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE	Previous output value

- Timing chart



(1) When (s2) turns off while EN is on and (s1) is on, (d) turns on.

(2) When (s2) turns on while EN is on, (d) turns off.

Operation error

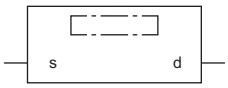
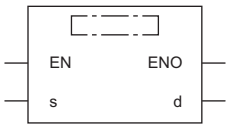
There is no operation error.

37 EDGE DETECTION FUNCTION BLOCKS

37.1 Detecting a Rising Edge

R_TRIG(_E)

These function blocks detect a signal rising edge, and outputs the pulse signal.

Ladder, FBD/LD		ST
[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO] Instance name(CLK:=s,Q:=d); [With EN/ENO] Instance name(EN:=en,ENO:=eno,CLK:=s,Q:=d);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (CLK)	Rising edge detection input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d (Q)	Output	Output variable	BOOL

Processing details

■Operation processing

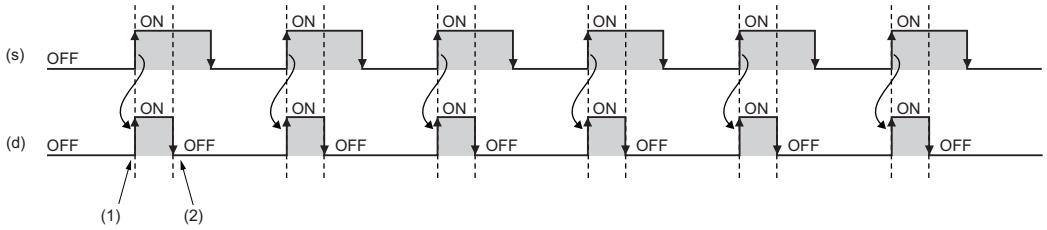
When (s) turns on, (d) turns on only for one scan.

■Operation result

1. Function block without EN/ENO

The operation processing is performed. The operation result is output from (d).

• Timing chart



(1) (d) turns on at the rising edge of (s).

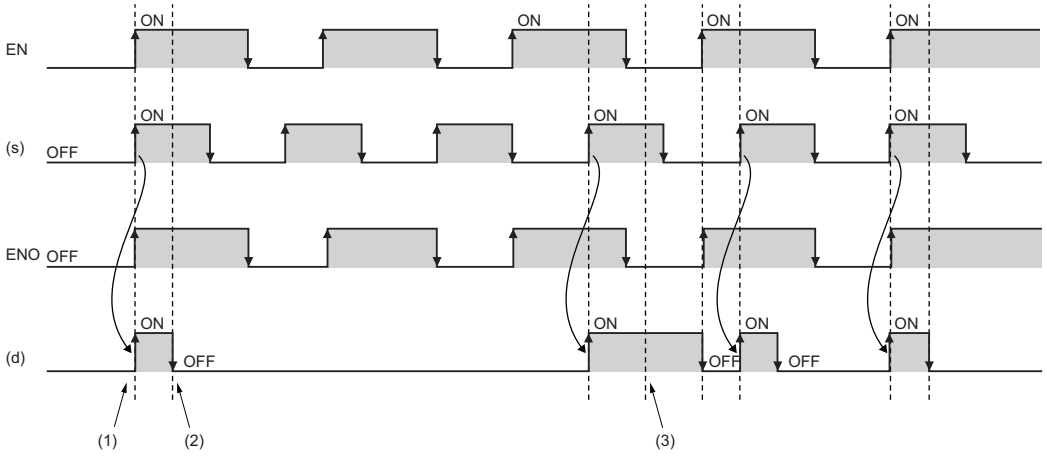
(2) (d) turns off in the next scan.

2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE	Previous output value

• Timing chart



(1) (d) turns on at the rising edge of (s) while EN is on.

(2) (d) turns off in the next scan.

(3) If EN is off, (d) holds the output value of the last scan.

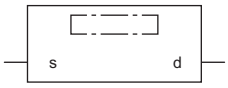
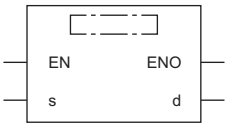
Operation error

There is no operation error.

37.2 Detecting a Falling Edge

F_TRIG(_E)

These function blocks detect a signal falling edge, and outputs the pulse signal.

Ladder, FBD/LD		ST
[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO] Instance name(CLK:=s,Q:=d); [With EN/ENO] Instance name(EN:= en,ENO:=eno,CLK:=s,Q:=d);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (CLK)	Falling edge detection input	Input variable	BOOL
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d (Q)	Output	Output variable	BOOL

Processing details

■Operation processing

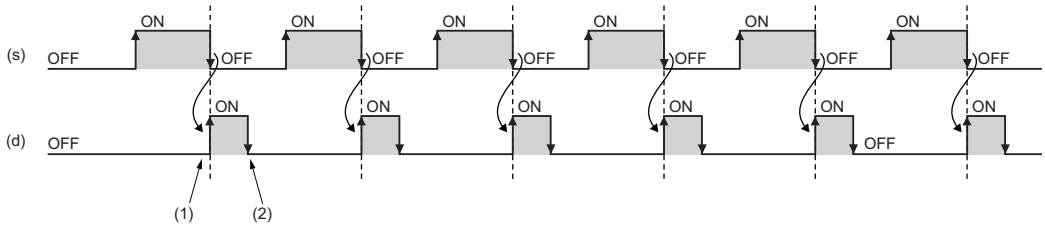
When (s) turns off, (d) turns on only for one scan.

■Operation result

1. Function block without EN/ENO

The operation processing is performed. The operation result is output from (d).

• Timing chart



(1) (d) turns on at the falling edge of (s).

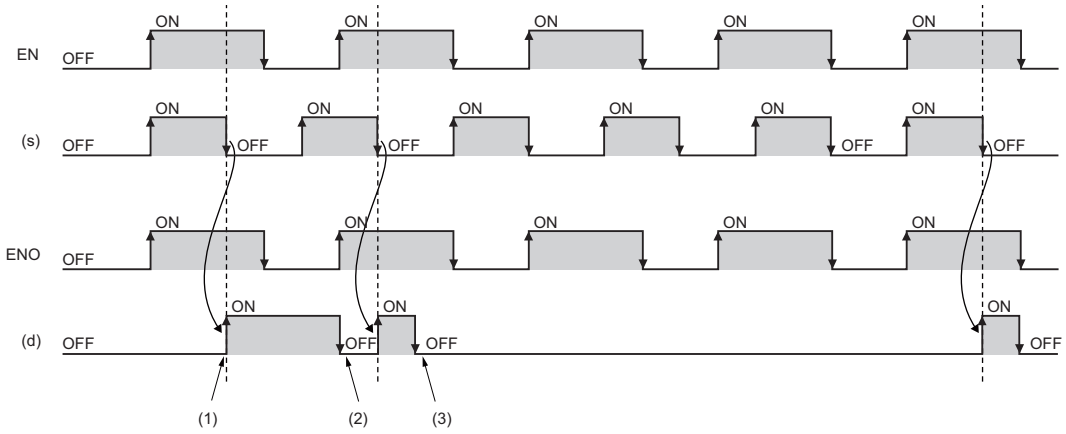
(2) (d) turns off in the next scan.

2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE	Previous output value

• Timing chart



(1) (d) turns on at the falling edge of (s) while EN is on.

(2) (d) turns off in the next scan.

(3) If EN is off, (d) holds the output value of the last scan.

Operation error

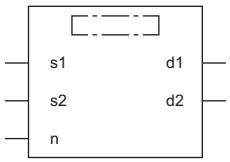
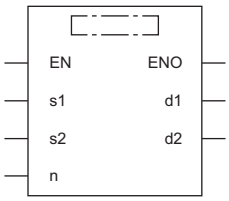
There is no operation error.

38 COUNTER/TIMER FUNCTION BLOCKS

38.1 Up Counter

CTU(_E)

These function blocks count up the number of rising edges of a signal.

Ladder, FBD/LD		ST
[Without EN/ENO]  	[With EN/ENO]  	[Without EN/ENO] Instance name(CU:=s1,R:=s2,PV:=n,Q:=d1,CV:=d2); [With EN/ENO] Instance name(EN:=en,ENO:=eno,CU:=s1,R:=s2,PV:=n,Q:=d1,CV:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (CU)	Count signal input	Input variable	BOOL
s2 (R)	Count value reset	Input variable	BOOL
n (PV)	Maximum count value	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d1 (Q)	End of count	Output variable	BOOL
d2 (CV)	Count value	Output variable	INT

Processing details

■Operation processing

1. Counting up

- When (s1) changes from off to on, the value in (d2) is counted up by one.
- When (d2) reaches the (n) value, (d1) turns on and the counting stops.
- Set the maximum counter value to (n). When (s2) turns on, (d1) turns off and (d2) is set to 0.

2. Maximum count value

The valid setting range of (n) is 0 to 32767.

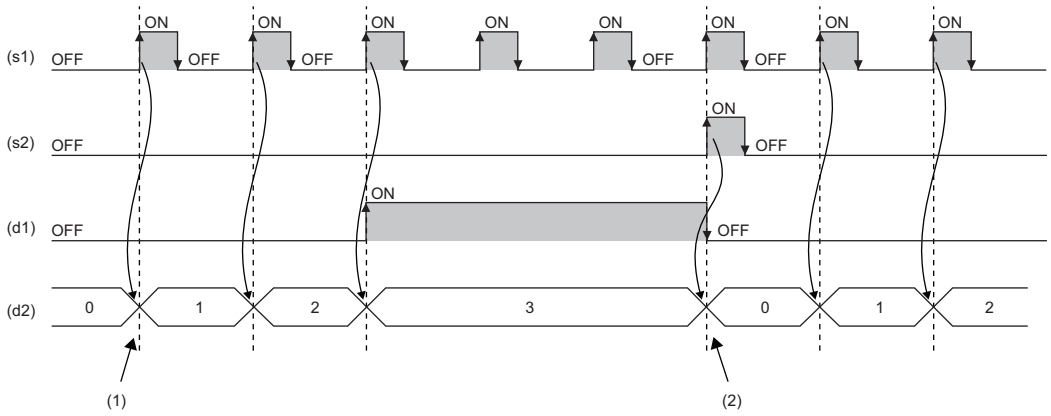
■Operation result

1. Function block without EN/ENO

The operation processing is performed. The operation result is output from (d1) and (d2).

• Timing chart

When (n)=3



(1) When (s1) turns on, the value in (d2) is counted up.

(2) When (s2) turns on, the value in (d2) is initialized.

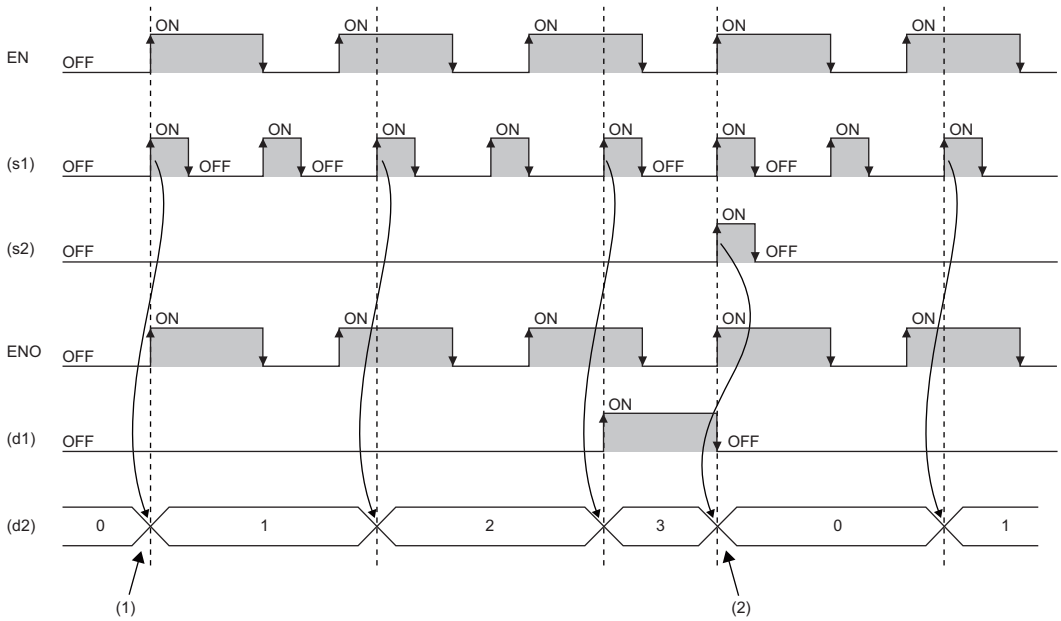
2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d1), (d2)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE	Previous output value

• Timing chart

When (n)=3



(1) When (s1) turns on while EN is on, the value in (d2) is counted up.

(2) When (s2) turns on while EN is on, the value in (d2) is initialized.

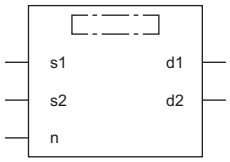
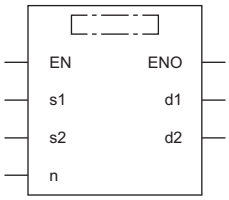
Operation error

There is no operation error.

38.2 Down Counter

CTD(_E)

These function blocks count down the number of rising edges of a signal.

Ladder, FBD/LD		ST
[Without EN/ENO]  	[With EN/ENO]  	[Without EN/ENO] Instance name(CD:=s1,LD:=s2,PV:=n,Q:=d1,CV:=d2); [With EN/ENO] Instance name(EN:=en,ENO:=eno,CD:=s1,LD:=s2,PV:=n,Q:=d1,CV:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1(CD)	Count signal input	Input variable	BOOL
s2 (LD)	Count value set	Input variable	BOOL
n (PV)	Start count value	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d1 (Q)	End of count	Output variable	BOOL
d2 (CV)	Count value	Output variable	INT

Processing details

■Operation processing

1. Counting down

- When (s1) changes from off to on, the value in (d2) is counted down by one.
- When (d2) is 0, (d1) turns on and the counting stops.
- Set the start count value to (n). When (s2) turns on, (d1) turns off and (n) is set to (d2).

2. Start count value

The valid setting range of (n) is 0 to 32767.

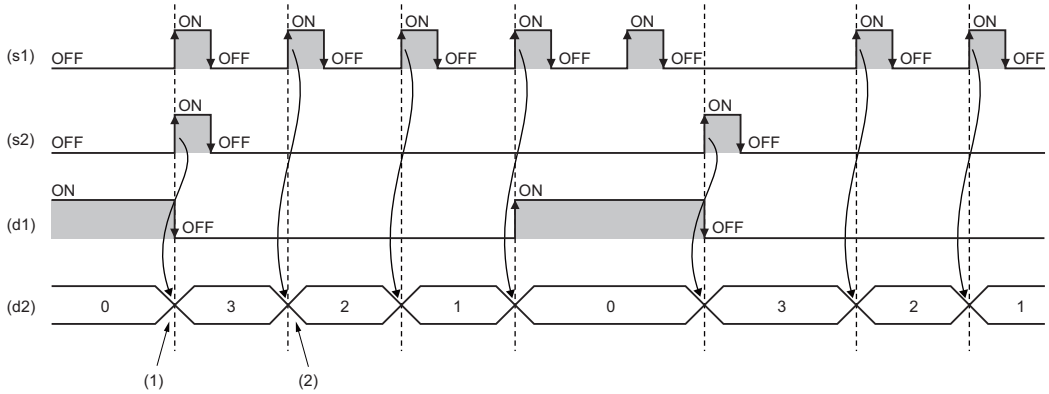
■Operation result

1. Function block without EN/ENO

The operation processing is performed. The operation result is output from (d1) and (d2).

• Timing chart

When (n)=3



(1) When (s2) turns on, the value in (d2) is initialized.

(2) When (s1) turns on, the value in (d2) is counted down.

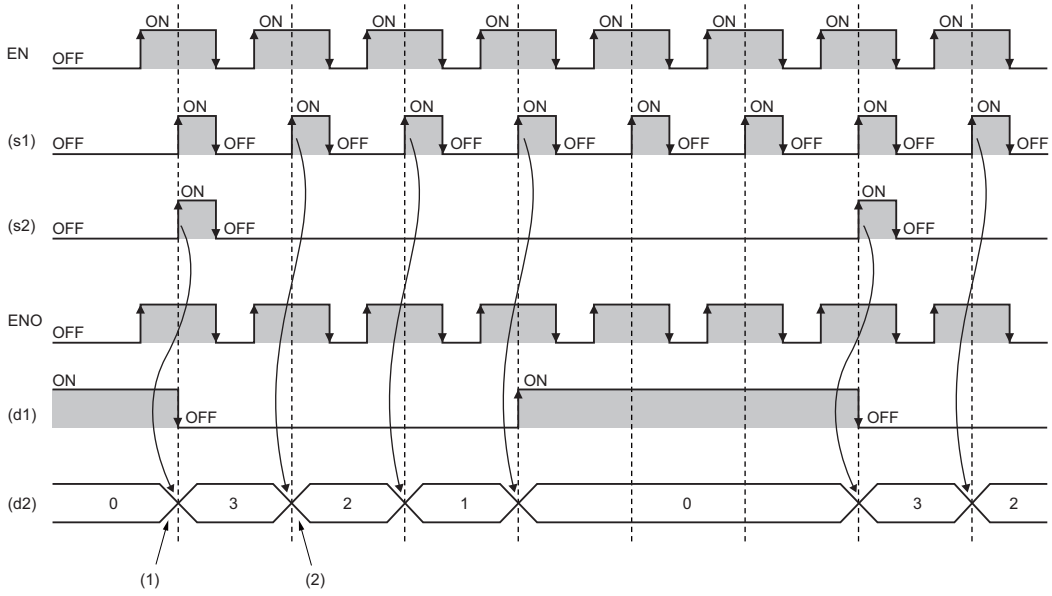
2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d1), (d2)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE	Previous output value

• Timing chart

When (n)=3



(1) When (s2) turns on while EN is on, the value in (d2) is initialized.

(2) When (s1) turns on while EN is on, the value in (d2) is counted down.

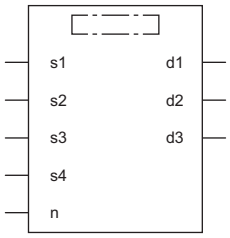
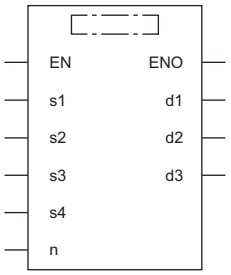
Operation error

There is no operation error.

38.3 Up/Down Counter

CTUD(_E)

These function blocks count up or down the number of rising edges of a signal.

Ladder, FBD/LD		ST
[Without EN/ENO]  	[With EN/ENO]  	[Without EN/ENO] Instance name(CU:=s1,CD:=s2,R:=s3,LD:=s4,PV:=n,QU:=d1,QD:=d2,CV:=d3); [With EN/ENO] Instance name(EN:=en,ENO:=eno,CU:=s1,CD:=s2,R:=s3,LD:=s4,PV:=n,QU:=d1,QD:=d2,CV:=d3);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s1 (CU)	Count signal input (for counting-up)	Input variable	BOOL
s2 (CD)	Count signal input (for counting-down)	Input variable	BOOL
s3 (R)	Count value reset	Input variable	BOOL
s4 (LD)	Count value set	Input variable	BOOL
n (PV)	Maximum count value or start count value	Input variable	INT
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d1 (QU)	End of count (for counting-up)	Output variable	BOOL
d2 (QD)	End of count (for counting-down)	Output variable	BOOL
d3 (CV)	Current count value	Output variable	INT

Processing details

■Operation processing

In (n), set the maximum count value for up count and the start count value for down count. The valid range of (n) is 0 to 32767.

1. Counting up

- When (s1) changes from off to on, the value in (d3) is counted up by one.
- When (d3) reaches the (n) value, (d1) turns on and the counting stops.
- When (s3) turns on, (d1) turns off and (d3) is set to 0.

2. Counting down

- When (s2) changes from off to on, the value in (d3) is counted down by one.
- When (d3) is 0, (d2) turns on and the counting stops.
- When (s4) turns on, (d2) turns off and (n) is set to (d3).

3. Others

- When (s1) and (s2) change from off to on simultaneously, the value in (d3) is counted up by one with priority given to (s1).
- When (s3) and (s4) turn on simultaneously, (d3) is set to 0 with priority given to (s3).

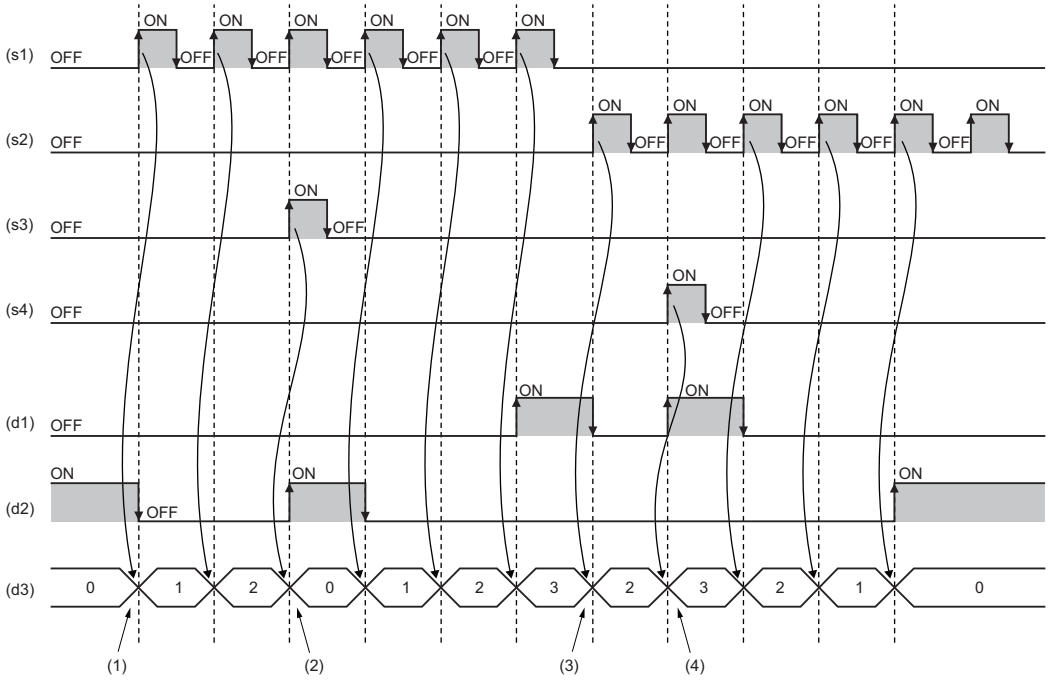
■Operation result

1. Function block without EN/ENO

The operation processing is performed. The operation result is output from (d1), (d2), and (d3).

- Timing chart

When (n)=3



(1) When (s1) turns on, the value in (d3) is counted up.
(2) When (s3) turns on, the value in (d3) is initialized.
(3) When (s2) turns on, the value in (d3) is counted down.
(4) When (s4) turns on, the value in (d3) is initialized.

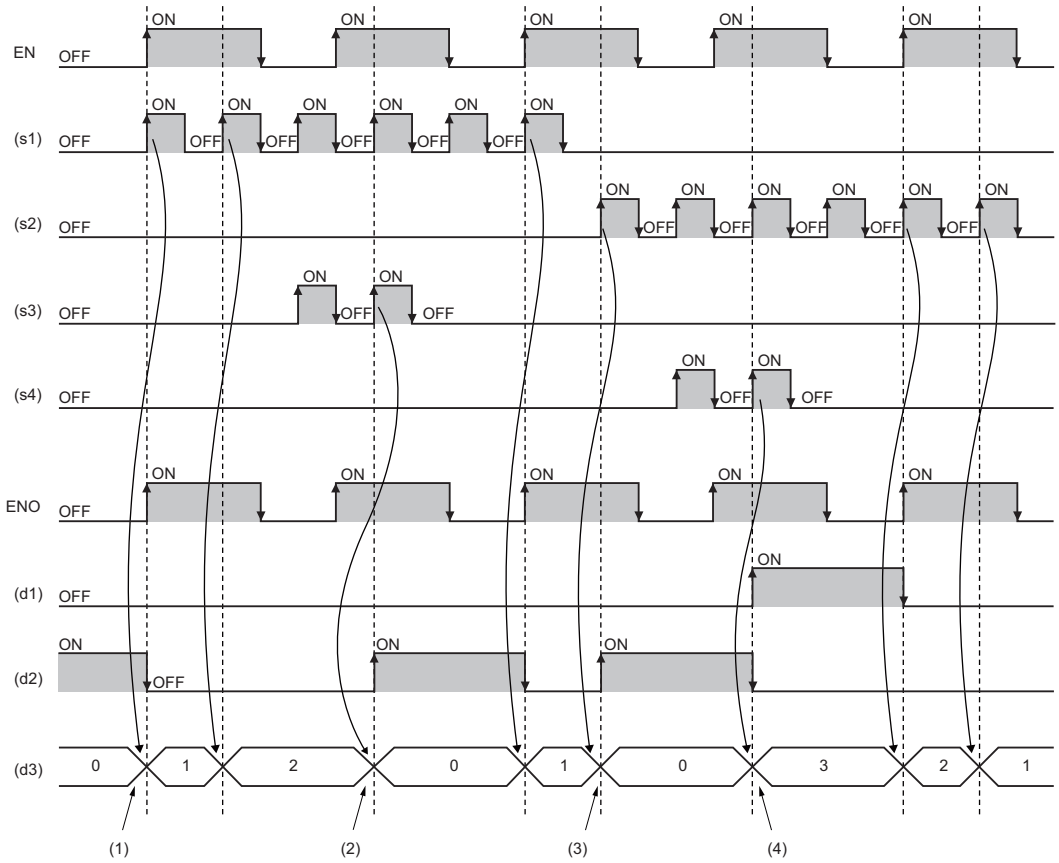
2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d1), (d2), (d3)**
TRUE (executed)	TRUE	Operation result output value
FALSE (not executed)	FALSE	Previous output value

• Timing chart

When (n)=3



(1) When (s1) turns on while EN is on, the value in (d3) is counted up.

(2) When (s3) turns on while EN is on, the value in (d3) is initialized.

(3) When (s2) turns on while EN is on, the value in (d3) is counted down.

(4) When (s4) turns on while EN is on, the value in (d3) is initialized.

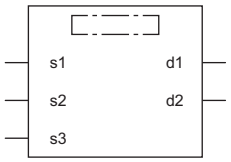
Operation error

There is no operation error.

38.4 Counter Function Block

COUNTER_FB_M

This function block starts counting up when the execution condition is satisfied.

Ladder, FBD/LD	ST
	Instance name(Coil:=s1,Preset:=s2,ValueIn:=s3,ValueOut:=d1,Status:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s1 (Coil)	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s2 (Preset)	Counter setting value	Input variable	INT
s3 (ValueIn)	Initial counter value	Input variable	INT
d1 (ValueOut)	Current counter value	Output variable	ANY16
d2 (Status)	Output	Output variable	BOOL

Processing details

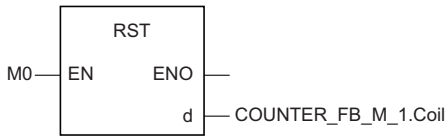
■Operation processing

- The number of rising edges (status changes (off to on)) of (s1) is counted. Counting is not performed while (s1) remains on. The counting starts from the (s3) value. When it reached the (s2) value, (d2) turns on. The current value is stored in (d1).
- The valid setting range of (s2) is 0 to 32767.
- The valid setting range of (s3) is -32768 to 32767. Note that if a negative value is specified, 0 will be used as the initial value.
- To reset the current value (d1), reset (s1) of FB directly.

Ex.

Label name: COUNTER_FB_M_1

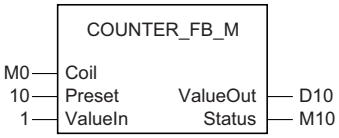
[Ladder program]



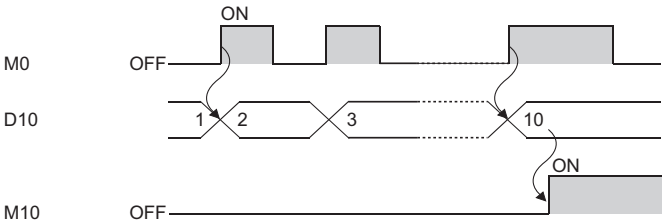
[ST program]

RST(M0, COUNTER_FB_M_1.Coil)

[Ladder example]



[Timing chart]



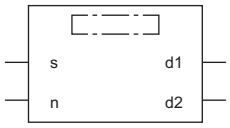
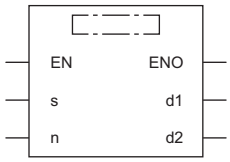
Operation error

There is no operation error.

38.5 Pulse Timer

TP(_E)

These function blocks keep the signal on for the specified period of time.

Ladder, FBD/LD		ST
[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO] Instance name(IN:=s,PT:=n,Q:=d1,ET:=d2); [With EN/ENO] Instance name(EN:=en,ENO:=eno,IN:=s,PT:=n,Q:=d1,ET:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Start of output	Input variable	BOOL
n (PT)	Output time setting value	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d1 (Q)	Output	Output variable	BOOL
d2 (ET)	Elapsed time	Output variable	TIME

Processing details

■Operation processing

1. Output

- When (s) turns on, (d1) turns on for the period of time set by (n). The time elapsed after (d1) turns on is set to (d2).
- Use the long timer to count the elapsed time.

2. End of output

- Once the elapsed time reaches the setting time, (d1) turns off.
- If (s) is off after (d1) turns off, the elapsed time is reset.
- Even when (s) turns off while (d1) is on, (d1) does not turn off.

3. Output time setting

The valid setting range of (n) is T#1ms to T#2147483ms. Note that the valid setting range will be as follows by changing the timer limit setting using the engineering tool.

Minimum value	Maximum value
Identical to the long timer setting value [ms] in the timer limit setting. Note that if the long timer setting value is smaller than 1ms, the minimum value will be 1ms.	The time satisfying the following condition is used. Note that the maximum value is a value that can be included within the range of time type because the output time setting value is of time type (32-bit value). • Output time setting value [ms] ≤ 2147483647 [ms] × Long timer setting value in the timer limit setting [ms] [Example] • If the long timer setting value is 0.001ms: T#1ms to T#2147483ms • If the long timer setting value is 1000ms: T#1000ms to T#2147483000ms

The value at the rising edge (off to on) of (d1) is used for the setting value of (n). When the (n) value is changed when (d1) is on, the new value will be enabled at the next output start timing.

■Operation result

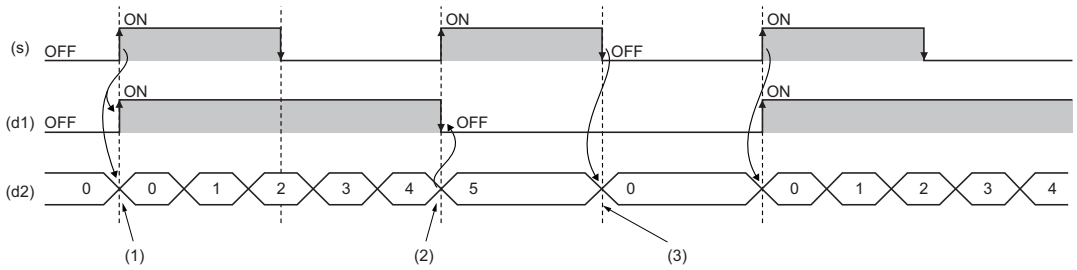
1. Function block without EN/ENO

The operation result will be as follows.

Operation result	(d1), (d2)
No operation error	Operation result output value
Operation error	Undefined value

• Timing chart

When n=T#5s (5s)



(1) When (s) turns on, (d1) turns on. When (s) turns on, (d2) starts measuring time.

(2) When the time measured in (d2) reaches the time set in (n), (d1) turns off.

(3) When both (s) and (d1) are off, the value in (d2) is initialized.

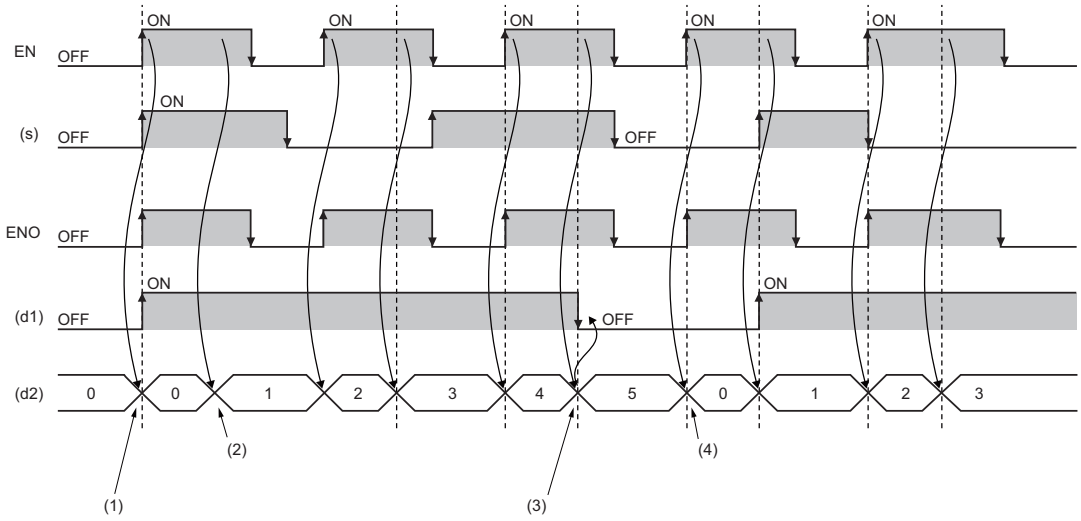
2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d1), (d2)**
TRUE (executed)	TRUE (no operation error)	Operation result output value
	FALSE (operation error)	Undefined value
FALSE (not executed)	FALSE	Previous output value

• Timing chart

When n=T#5s (5s)



(1) When (s) turns on while EN is on, (d1) turns on. When (s) turns on while EN is on, (d2) starts measuring time.

(2) While EN is on, the time value is incremented by 1.

(3) When the time measured in (d2) reaches the time set in (n), (d1) turns off.

(4) When EN is on and both (s) and (d1) are off, the value in (d2) is initialized.

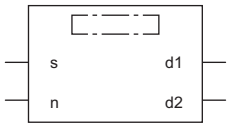
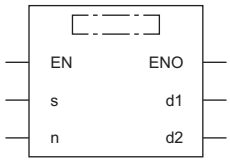
Operation error

Error code (SD0)	Description
3281H	The output time setting value exceeds the valid range.

38.6 On Delay Timer

TON(_E)

These function blocks turn on a signal after the specified period of time.

Ladder, FBD/LD		ST
[Without EN/ENO] 	[With EN/ENO] 	[Without EN/ENO] Instance name(IN:=s,PT:=n,Q:=d1,ET:=d2); [With EN/ENO] Instance name(EN:=en,ENO:=eno,IN:=s,PT:=n,Q:=d1,ET:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Time measurement	Input variable	BOOL
n (PT)	Delay time setting value	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d1 (Q)	Output	Output variable	BOOL
d2 (ET)	Elapsed time	Output variable	TIME

Processing details

■Operation processing

1. Output

- When (s) turns on, (d1) turns on after the time that was set by (n). The delay time elapsed after (d1) turns on is set to (d2).
- When (s) turns off, (d1) turns off and the delay elapsed time is also reset.
- Use the long timer to count the elapsed time.

2. Delay time setting

The valid setting range of (n) is T#1ms to T#2147483ms. Note that the valid setting range will be as follows by changing the timer limit setting using the engineering tool.

Minimum value	Maximum value
Identical to the long timer setting value [ms] in the timer limit setting. Note that if the long timer setting value is smaller than 1ms, the minimum value will be 1ms.	The time satisfying the following condition is used. Note that the maximum value is a value that can be included within the range of time type because the delay time setting value is of time type (32-bit value). • Delay time setting value [ms] ≤ 2147483647 [ms] × Long timer setting value of in the timer limit setting [ms] [Example] • If the long timer setting value is 0.001ms: T#1ms to T#2147483ms • If the long timer setting value is 1000ms: T#1000ms to T#2147483000ms

The value at the rising edge (off to on) of (d) is used for the setting value of (n). When the (n) value is changed while (s) is on, the new value will be enabled at the next rising edge of (s).

■Operation result

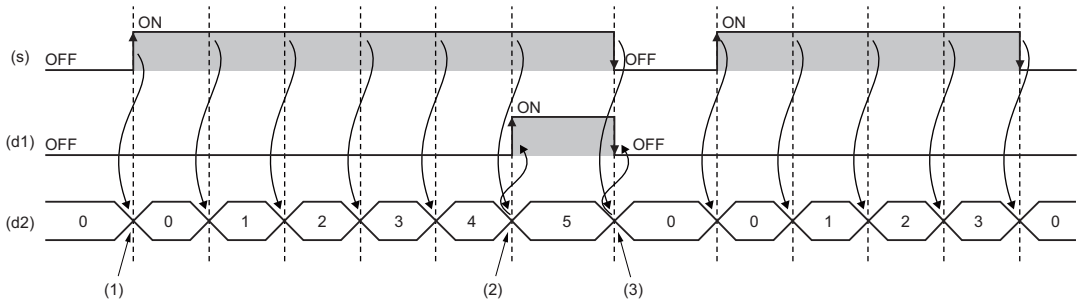
1. Function block without EN/ENO

The operation result will be as follows.

Operation result	(d1), (d2)
No operation error	Operation result output value
Operation error	Undefined value

• Timing chart

When n=T#5s (5s)



(1) When (s) turns on, (d2) starts measuring time.

(2) When the time measured in (d2) reaches the time set in (n), (d1) turns on.

(3) When both (s) and (d1) turn off, the value in (d2) is initialized.

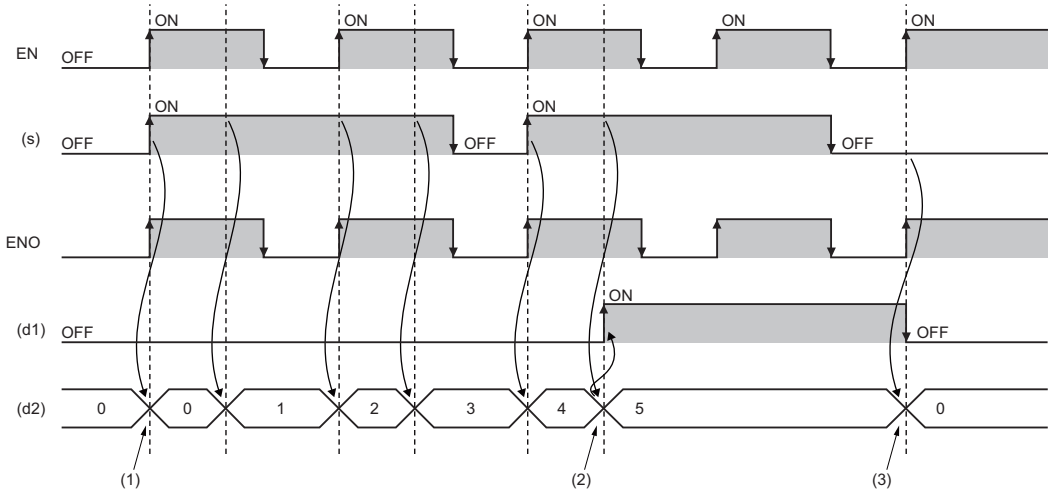
2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d1), (d2)**
TRUE (executed)	TRUE (no operation error)	Operation result output value
	FALSE (operation error)	Previous output value
FALSE (not executed)	FALSE	Previous output value

• Timing chart

When n=T#5s (5s)



(1) When (s) turns on while EN is on, (d2) starts measuring time.

(2) When the time measured in (d2) reaches the time set in (n), (d1) turns on.

(3) When both (s) and (d1) turn off while EN is on, the value in (d2) is initialized.

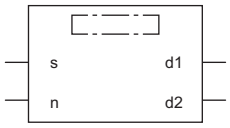
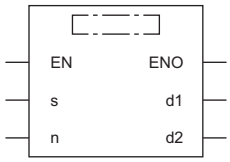
Operation error

Error code (SD0)	Description
3281H	The output time setting value exceeds the valid range.

38.7 Off Delay Timer

TOF(_E)

These function blocks turn off a signal after the specified period of time.

Ladder, FBD/LD		ST
[Without EN/ENO]  	[With EN/ENO]  	[Without EN/ENO] Instance name(IN:=s,PT:=n,Q:=d1,ET:=d2); [With EN/ENO] Instance name(EN:=en,ENO:=eno,IN:=s,PT:=n,Q:=d1,ET:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Time measurement	Input variable	BOOL
n (PT)	Delay time setting value	Input variable	TIME
ENO	Output status (TRUE: Normal, FALSE: Abnormal or operation stop)	Output variable	BOOL
d1 (Q)	Output	Output variable	BOOL
d2 (ET)	Elapsed time	Output variable	TIME

Processing details

■Operation processing

1. Output

- When (s) turns on, (d1) turns on.
- When (s) changes from on to off, (d1) turns off after the time that was set by (n). The delay time elapsed after (d1) turns off is set to (d2).
- Use the long timer to count the elapsed time.

2. Delay time setting

The valid setting range of (n) is T#1ms to T#2147483ms. Note that the valid setting range will be as follows by changing the timer limit setting using the engineering tool.

Minimum value	Maximum value
Identical to the long timer setting value [ms] in the timer limit setting. Note that if the long timer setting value is smaller than 1ms, the minimum value will be 1ms.	The time satisfying the following condition is used. Note that the maximum value is a value that can be included within the range of time type because the delay time setting value is of time type (32-bit value). • Delay time setting value [ms] ≤ 2147483647 [ms] × Long timer setting value of in the timer limit setting [ms] [Example] • If the long timer setting value is 0.001ms: T#1ms to T#2147483ms • If the long timer setting value is 1000ms: T#1000ms to T#2147483000ms

The value at the falling edge (on to off) of (s) is used for the setting value of (n). When the (n) value is changed when (s) is off, the new value will be enabled at the next falling edge of (s).

■Operation result

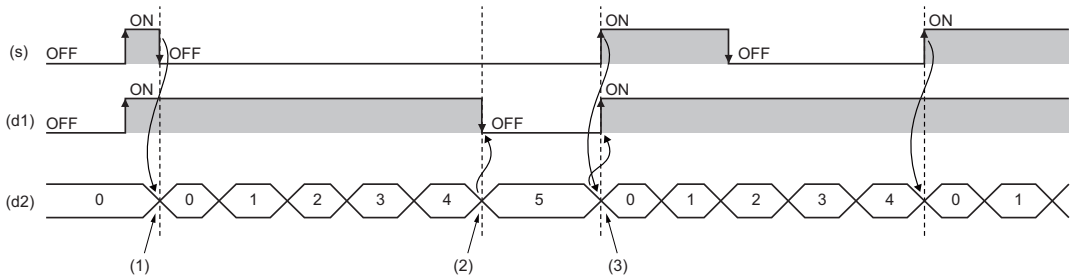
1. Function block without EN/ENO

The operation result will be as follows.

Operation result	(d1), (d2)
No operation error	Operation result output value
Operation error	Undefined value

• Timing chart

When n=T#5s (5s)



(1) When (s) turns off, (d2) starts measuring time.

(2) When the time measured in (d2) reaches the time set in (n), (d1) turns on.

(3) When (s) turns on, the value in (d2) is initialized.

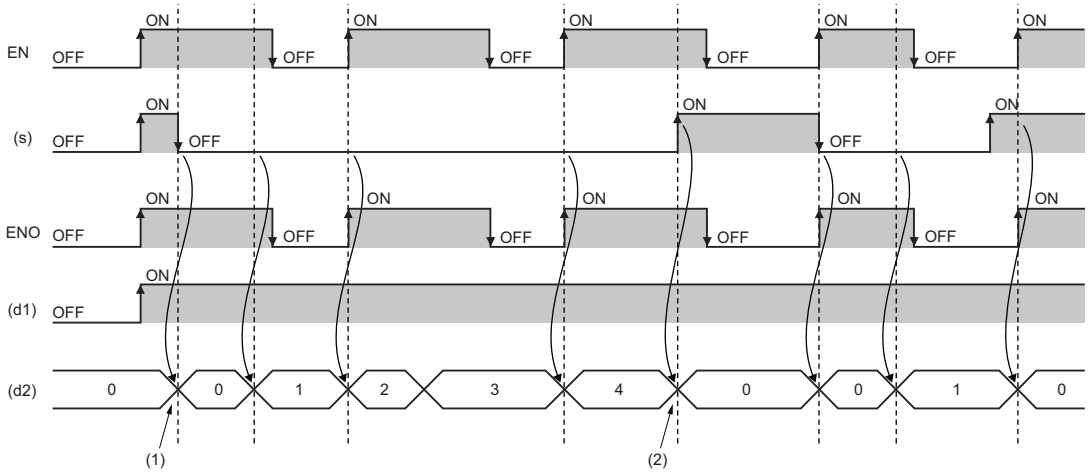
2. Function block with EN/ENO

The execution conditions and operation results will be as follows.

Execution condition	Operation result	
EN	**ENO**	**(d)**
TRUE (executed)	TRUE (no operation error)	Operation result output value
	FALSE (operation error)	Previous output value
FALSE (not executed)	FALSE	Previous output value

• Timing chart

When n=T#5s (5s)



(1) When (s) turns off while EN is on, (d2) starts measuring time.

(2) When (s) turns on while EN is on, the value in (d2) is initialized.

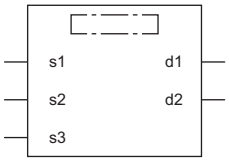
Operation error

Error code (SD0)	Description
3281H	The output time setting value exceeds the valid range.

38.8 Timer Function Block

TIMER_□_M

These function blocks start counting a timer when the execution condition is satisfied, and continue counting until the timer reaches the set value.

Ladder, FBD/LD	ST
 (□ is to be replaced by any of the following: TIMER_10_FB_M, TIMER_100_FB_M, TIMER_HIGH_FB_M, TIMER_LOW_FB_M, TIMER_CONT_FB_M, or TIMER_CONTHFB_M.)	Instance name(Coil:=s1,Preset:=s2,ValueIn:=s3,ValueOut:=d1,Status:=d2);

Setting data

■Description, type, data type

Argument	Description	Type	Data type
s1 (Coil)	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s2 (Preset)	Timer setting value	Input variable	INT
s3 (ValueIn)	Initial timer value	Input variable	INT
d1 (ValueOut)	Current timer value	Output variable	ANY16
d2 (Status)	Output	Output variable	BOOL

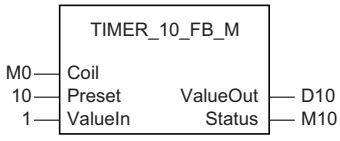
Processing details

■TIMER_10_FB_M

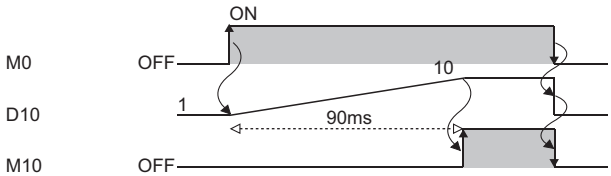
- When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×10ms. When the value reaches (s2)×10ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to the initial value (s3), and (d2) also turns off.
- If the unit of measurement of the high-speed timer (in the timer limit setting) is changed from the default value using the engineering tool, a warning will be issued during conversion of modified or newly added programs or all programs in a project.
- The valid setting range of (s2) is 0 to 32767.
- The valid setting range of (s3) is -32768 to 32767. Note that if a negative value is specified, 0 will be used as the initial value.

Ex.

[Ladder example]



[Timing chart]

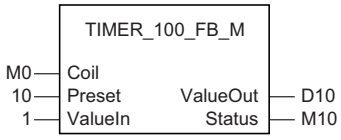


■TIMER_100_FB_M

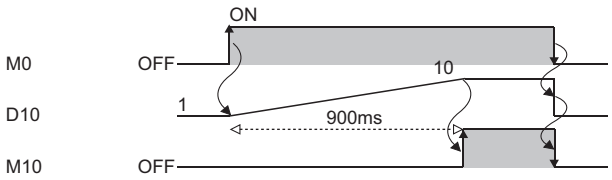
- When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×100ms. When the value reaches (s2)×100ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to the initial value (s3), and (d2) also turns off.
- If the unit of measurement of the low-speed timer (in the timer limit setting) is changed from the default value using the engineering tool, a warning will be issued during conversion of modified or newly added programs or all programs in a project.
- The valid setting range of (s2) is 0 to 32767.
- The valid setting range of (s3) is -32768 to 32767. Note that if a negative value is specified, 0 will be used as the initial value.

Ex.

[Ladder example]



[Timing chart]

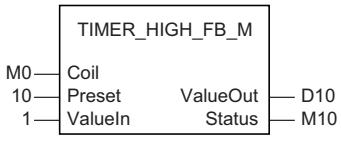


■TIMER_HIGH_FB_M

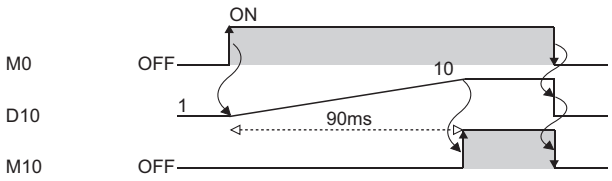
- This is a high-speed timer whose unit of measurement is 0.1 to 100ms. When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×0.1 to 100ms (variable; set in parameter). When the value reaches (s2)×0.1 to 100ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to the initial value (s3), and (d2) also turns off.
- The unit of measurement of the high-speed timer is 10ms by default. The unit can be changed in the range from 0.01 to 100ms.
- The valid setting range of (s2) is 0 to 32767.
- The valid setting range of (s3) is -32768 to 32767. Note that if a negative value is specified, 0 will be used as the initial value.

Ex.

[Ladder example]



[Timing chart]

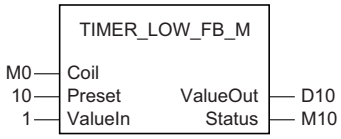


■TIMER_LOW_FB_M

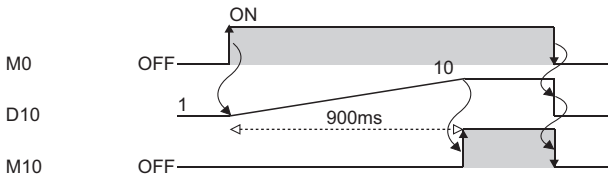
- This is a low-speed timer whose unit of measurement is 1 to 1000ms. When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×1 to 1000ms (variable; set in parameter). When the value reaches (s2)×1 to 1000ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to the initial value (s3), and (d2) also turns off.
- The unit of measurement of the low-speed timer is 100ms by default. The unit can be changed in the range from 1 to 1000ms (in increments of 1ms).
- The valid setting range of (s2) is 0 to 32767.
- The valid setting range of (s3) is -32768 to 32767. Note that if a negative value is specified, 0 will be used as the initial value.

Ex.

[Ladder example]



[Timing chart]



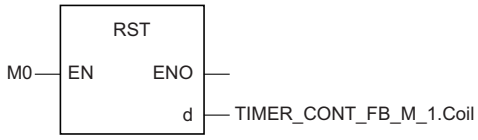
■TIMER_CONT_FB_M/TIMER_CONTHFB_M

- This is a retentive timer that measures the on time of a variable. When (s1) turns on, measurement of the current value starts. There are two retentive timers: low-speed (TIMER_CONT_FB_M) and high-speed (TIMER_CONTHFB_M) retentive timers.
- The measurement starts from (s3)×1 to 1000ms (0.1 to 100ms for the high-speed retentive timer) (variable; set in parameter). When the value reaches (s2)×1 to 1000ms (0.1 to 100ms for the high-speed retentive timer), (d2) turns on. The measured current value is output to (d1).
- Even when (s1) is off, the on/off states of (d1) and (d2) are held. When (s1) turns on again, the measurement resumes with the measured value that has been held.
- The unit of measurement (time limit) for the retentive timers is common to both the low-speed timer (TIMER_LOW_FB_M) and high-speed timer (TIMER_HIGH_FB_M).
  - Low-speed retentive timer: Low-speed timer
  - High-speed retentive timer: High-speed timer
- The valid setting range of (s2) is 0 to 32767.
- The valid setting range of (s3) is -32768 to 32767. Note that if a negative value is specified, 0 will be used as the initial value.
- To reset (d1) of a retentive timer, reset (s1) of FB directly.

Ex.

Label name: TIMER_CONT_FB_M_1

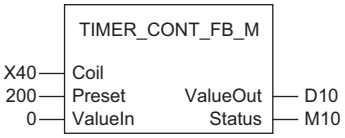
[Ladder program]



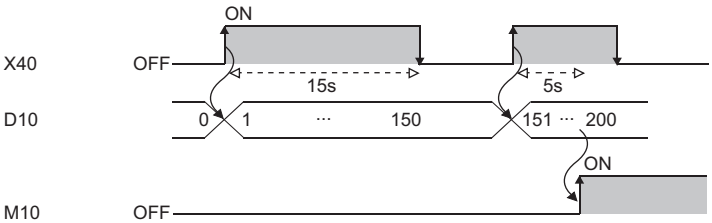
[ST program]

RST(M0,TIMER_CONT_FB_M_1.Coil)

[Ladder example]



[Timing chart]



Operation error

There is no operation error.

PART 8 MOTION CONTROL FUNCTION BLOCK

This part consists of the following chapters.

39 MOTION CONTROL FB OVERVIEW

40 LIST OF MOTION CONTROL FUNCTION BLOCK VARIABLES

41 ADMINISTRATIVE FUNCTION BLOCKS

42 MOTION FUNCTION BLOCK

43 STANDARD FUNCTION BLOCK

39 MOTION CONTROL FB OVERVIEW

This chapter describes how to use motion control FBs and the specifications.
The motion control FB includes FBs established by PLCopen®. The basic specifications of I/O signals follow the PLCopen® motion control FBs.
For the common specifications of the function block, refer to the following.
☞ Page 65 Function Blocks (FB)

39.1 Motion Control FBs

This section describes the motion control FBs.

How to use motion control FBs

The procedure for using motion control FBs is described below.

1. Create the motion control FB instance (global label or local label) using the engineering tool.
2. Create a program which sets the I/O argument for the motion control FB instance.
3. The control will be executed when Execution command (Execute) or Enable (Enable) of the motion control FB is set to TRUE.

Point

- When one motion control FB instance is being executed (including during buffering), the instance cannot be executed on multiple axes by changing the axis input (AXIS_REF input or AXES_GROUP_REF input). The axis input of the motion control FB can only be changed when an instance is not being executed. When the axis input is changed while an instance is being executed, a warning will occur and the change will be ignored. Therefore, ensure that the number of motion control FB instances created is equal to the number of axes being controlled at the same time.
- Do not allow one motion control FB instance to be called from multiple programs (programs to be executed in parallel, including interrupt programs) at the same time. Operation when an instance is called at the same time cannot be guaranteed.

Operation of motion control FBs for each system status

Motion control FBs can only be executed during RUN.
○: Available, ×: Not available

System status	Operation availability
STOP	×
RUN	○
Moderate error	×
Major error	×

Motion control FB types

Motion control FBs are classified according the operation details and the execution method.

Administrative FBs/Motion related FBs/Standard FBs

Motion control FBs are classified into the following types according to the operation details.

Type	Operation details
Administrative FB	• A motion control FB that takes an axis or an axes group for the argument and does not change the axis status or the axes group status by execution. (There are some exceptions.) • In most cases, an administrative FB can execute multiple instances to an axis or an axes group at the same time. • When multiple instances are executed for an axis or an axes group at the same time, control is preferentially performed from the one whose priority by which to execute the motion control FB is high.
Motion related FB	• A motion control FB that takes an axis or an axes group for the argument and changes the axis status or the axes group status by execution. • In most cases, a motion FB can be executed to only one axis or axes group. However, some motion control FBs can be executed at the same time. • In most cases, the axis status or the axes group status will not be changed even if an administrative FB is executed while a motion related FB is being executed. However, some motion control FBs can cause a specific state transition. • When multiple instances are executed for an axis or an axes group at the same time, control is preferentially performed from the one whose priority by which to execute the motion control FB is high.
Standard FB	• A motion control FB that does not take an axis or axes group for the argument. • A standard FB can execute multiple instances at the same time. Since it is not related to the axis, it does not affect either motion related FBs or administrative FBs.

The following table lists the priority by which to execute motion control FBs.

—: No priority assignment

Priority		Motion control FB
High ↑ ↓ Low	1	• MC_Stop (Forced Stop) • MC_GroupStop (Group Forced Stop) • MCv_MoveWait (Axis Control Start Wait) • MCv_GroupMoveWait (Axis Group Control Start Wait)
	2	• MC_Home (OPR) • MC_MoveAbsolute (Absolute Value Positioning) • MC_MoveRelative (Relative Value Positioning) • MCv_Jog (JOG) • MC_MoveVelocity (Speed Control) • MC_TorqueControl (Torque Control) • MCv_SpeedControl (Speed Control (Including Position Loop)) • MCv_PressureControl (Pressure Control) • MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) • MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) • MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) • MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) • MC_SetPosition (Current Position Change) • MCv_SetTorqueLimit (Torque Limit Value) • MC_CamIn (Cam Operation Start) • MC_GearIn (Gear Operation Start) • MC_CombineAxes (Addition/Subtraction Positioning) • MCv_BacklashCompensationFilter (Backlash Compensation Filter) • MCv_SmoothingFilter (Smoothing Filter) • MCv_DirectionFilter (Moving Direction Restriction Filter) • MCv_SpeedLimitFilter (Speed Limit Filter) • MCv_AdvancedSync (Advanced Synchronous Control) • MCv_MovePositioningData (Multiple Axes Positioning Data Operation) • MCv_PositionControl (Absolute Position Tracking Control) • MCv_CyclicPosition (Motion Cyclic Position Control) • MCv_CyclicVelocity (Motion Cyclic Speed Control) • MCv_CyclicTorque (Motion Cyclic Torque Control)
	3	• MC_TouchProbe (Touch Probe Enabled) • MCv_ChangeCycle (Current Value Change per Cycle)
	—	• MC_GroupEnable (Axes Group Enabled) • MC_GroupDisable (Axes Group Disabled) • MC_Power (Operation Available) • MC_SetOverride (Override Value Setting) • MC_ReadParameter (Parameter Read) • MC_WriteParameter (Parameter Write) • MC_ReadStatus (Single Axis Status Read) • MC_Reset (Axis Error Reset) • MC_GroupReset (Axes Group Error Reset) • MC_AbortTrigger (Touch Probe Disabled) • MC_CamTableSelect (Cam Table Selection) • MCv_ReadProfileData (Profile Read) • MCv_WriteProfileData (Profile Write) • MCv_AllPower (All Axes Operation Available) • MC_GroupSetOverride (Axes Group Override Value Setting) • MCv_MotionErrorReset (Motion Error Reset) • MCv_SyncOperationCycles (Control Operation Cycle Synchronization) • MCv_AdvPositionPerCycleCalc (Advanced Synchronous Control Current Position per Cycle Calculation) • MCv_AdvCamSetPositionCalc (Advanced Synchronous Control Cam Set Position Calculation) • MC_DigitalCamSwitch (Digital Cam Switch Output)

Execution command (Execute) type/Enable (Enable) type

Some motion control FBs are executed with Execution command (Execute), while others are executed with Enable (Enable).

Type	Execution command (Execute) type	Enable (Enable) type	Other types
Administrative FB	• MC_GroupEnable (Axes Group Enabled) • MC_GroupDisable (Axes Group Disabled) • MC_SetPosition (Current Position Change) • MCv_SetTorqueLimit (Torque Limit Value) • MC_WriteParameter (Parameter Write) • MC_Reset (Axis Error Reset) • MC_GroupReset (Axes Group Error Reset) • MC_TouchProbe (Touch Probe Enabled) • MC_AbortTrigger (Touch Probe Disabled) • MC_CamTableSelect (Cam Table Selection) • MCv_ChangeCycle (Current Value Change per Cycle) • MCv_MotionErrorReset (Motion Error Reset) • MCv_AdvPositionPerCycleCalc (Advanced Synchronous Control Current Position per Cycle Calculation) • MCv_AdvCamSetPositionCalc (Advanced Synchronous Control Cam Set Position Calculation)	• MC_Power (Operation Available) • MC_SetOverride (Override Value Setting) • MC_ReadParameter (Parameter Read) • MCv_AllPower (All Axes Operation Available) • MC_GroupSetOverride (Axes Group Override Value Setting) • MCv_SyncOperationCycles (Control Operation Cycle Synchronization) • MC_DigitalCamSwitch (Digital Cam Switch Output) • MC_ReadStatus (Single Axis Status Read)	—
Motion related FB	• MC_Home (OPR) • MC_Stop (Forced Stop) • MC_GroupStop (Group Forced Stop) • MC_MoveAbsolute (Absolute Value Positioning) • MC_MoveRelative (Relative Value Positioning) • MC_MoveVelocity (Speed Control) • MC_TorqueControl (Torque Control) • MCv_SpeedControl (Speed Control (Including Position Loop)) • MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) • MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) • MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) • MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) • MC_CamIn (Cam Operation Start) • MC_GearIn (Gear Operation Start) • MC_CombineAxes (Addition/Subtraction Positioning) • MCv_MoveWait (Axis Control Start Wait) • MCv_GroupMoveWait (Axis Group Control Start Wait) • MCv_MovePositioningData (Multiple Axes Positioning Data Operation) • MCv_PositionControl (Absolute Position Tracking Control) • MCv_CyclicPosition (Motion Cyclic Position Control) • MCv_CyclicVelocity (Motion Cyclic Speed Control) • MCv_CyclicTorque (Motion Cyclic Torque Control) • MCv_PressureControl (Pressure Control)	• MCv_BacklashCompensationFilter (Backlash Compensation Filter) • MCv_SmoothingFilter (Smoothing Filter) • MCv_DirectionFilter (Moving Direction Restriction Filter) • MCv_SpeedLimitFilter (Speed Limit Filter) • MCv_AdvancedSync (Advanced Synchronous Control)	• MCv_Jog (JOG)
Standard FB	• MCv_ReadProfileData (Profile Read) • MCv_WriteProfileData (Profile Write)	—	—

The basic operation of each motion control FB by Execution command (Execute command) and Enable (Enable) is shown below. Note that some motion control FBs have different specifications.

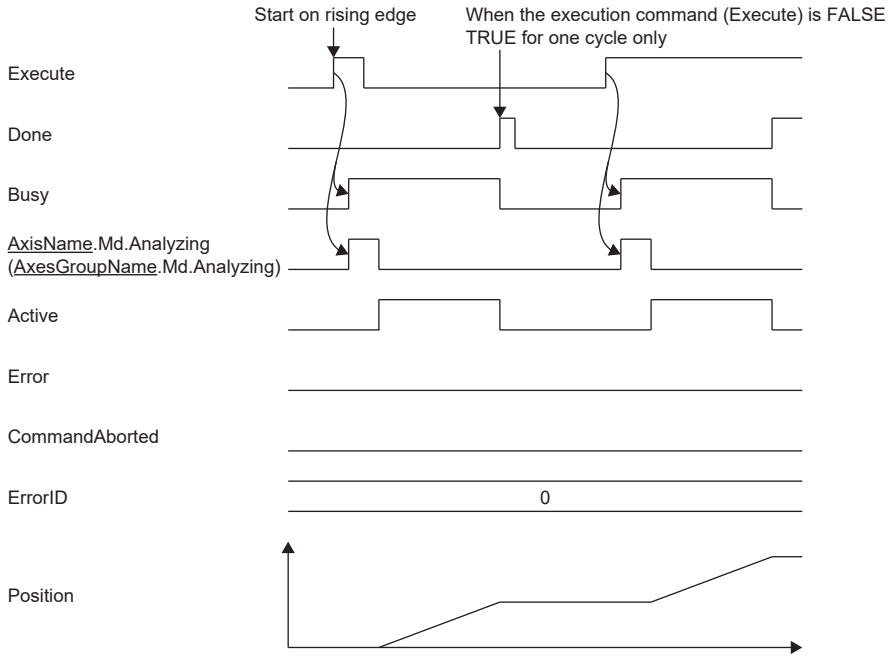
■Basic operation of Execution command (Execute) type motion control FBs

- Execution command (Execute) type FBs read the input parameters at the rising edge of Execution command (Execute) and then start operation. Once operation has started, the operation will be continued until it is completed even if Execution command (Execute) is set to FALSE.
- When operation has started, only one output variable among Executing (Busy), Execution completion (Done), Error (Error), and Abortion of execution (CommandAborted) becomes TRUE.
- Execution completion (Done), Error (Error), Error code (ErrorID), and Abortion of execution (CommandAborted) are reset at the falling edge of Execution command (Execute). Executing (Busy) and Controlling (Active) are not affected.
- When changing the input parameter during operation, the change is reflected at restart (retrigger) of Execution command (Execute) or by continuous update using Continuous update (ContinuousUpdate).
- Analyzing (AxisName.Md.Analyzing/AxesGroupName.Md.Analyzing) becomes TRUE from the rising edge of Executing (Busy), and Analyzing (AxisName.Md.Analyzing/AxesGroupName.Md.Analyzing) becomes FALSE after starting the operation.
- When Execution command (Execute) is used in pulse, Execution completion (Done) becomes TRUE for one cycle only.

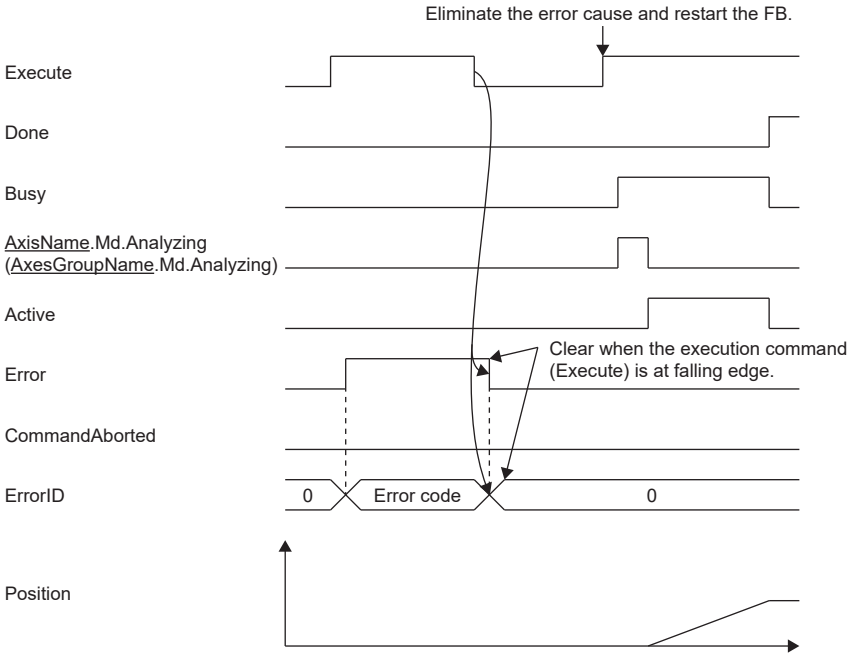
Ex.

The timing chart for Execution command (Execute) type motion control FBs is shown below.

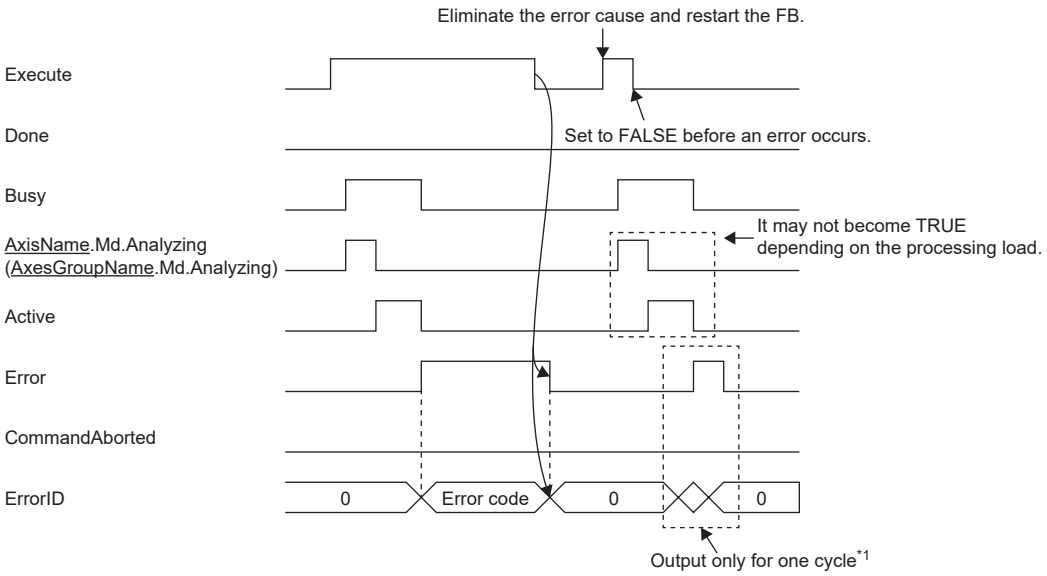
- When the FB is normal



• When the I/O variables are abnormal



• When the input variables are abnormal



*1 The following operation is performed when the termination condition of the motion control FB (Execution command (Execute) is FALSE) is met. For motion control FBs not related to the axis or motion control FBs which do not stop with decelerating, Error (Error) becomes TRUE for one cycle only and Error code (ErrorID) is output. For motion control FBs that need to perform a deceleration stop, Error (Error) becomes TRUE until the stop with decelerating and Error code (ErrorID) is maintained. At stop completion of the axis, Error (Error) becomes FALSE and Error code (ErrorID) is cleared.

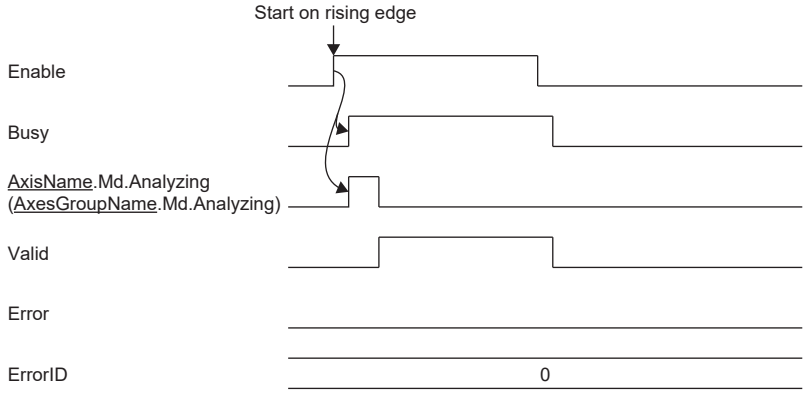
■Basic operation of Enable (Enable) type motion control FBs

- Enable (Enable) type motion control FBs are continuously executed while Enable (Enable) is TRUE.
- Output value valid (Valid) indicates that the output is a valid value. After Output value valid (Valid) becomes FALSE, each output does not change.
- Only one output variable among Output value valid (Valid)/Enabled (Enabled)/Executing (Busy), Error (Error), and Abortion of execution (CommandAborted) becomes TRUE.
- Analyzing (AxisName.Md.Analyzing/AxesGroupName.Md.Analyzing) becomes TRUE from the rising edge of Executing (Busy), and Analyzing (AxisName.Md.Analyzing/AxesGroupName.Md.Analyzing) becomes FALSE after starting the operation.
- Output variables not listed above are reset on the falling edge of Enable (Enable). (The reset timing is the same as Output value valid (Valid)/Enabled (Enabled)/Executing (Busy), Error (Error), and Abortion of execution (CommandAborted).)

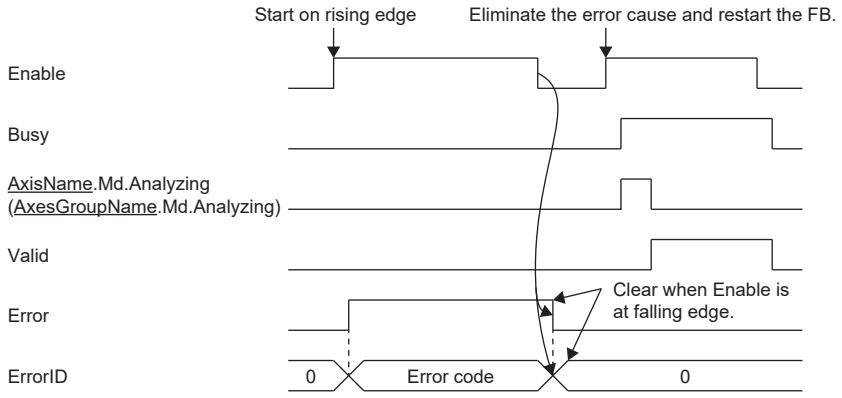
Ex.

The timing chart for Enable (Enable) type motion control FBs is shown below.

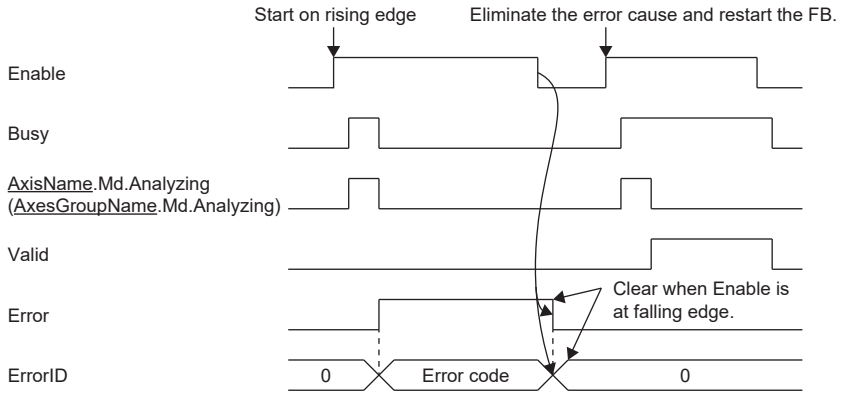
- When the FB is normal



- When the I/O variables are abnormal



- When the input variables are abnormal



Motion control types

The following types of axis and axes group control can be executed by motion related motion control FBs.

Large classification	Middle classification	Small classification	Description
Axis control	Single axis control	Positioning control	Control in which Axis status (AxisName.Md.AxisStatus) is set to "5: During positioning operation (DiscreteMotion)" and the axis is moved to the target position.
		Continuous control	Control in which Axis status (AxisName.Md.AxisStatus) is set to "6: During continuous operation (ContinuousMotion)" and continuous control is performed for the axis.
		Synchronous control	Control which has Master axis (Master) and Slave axis (Slave) as I/O variables and performs synchronous control for the axis with the Axis status (AxisName.Md.AxisStatus) of Slave axis (Slave) set to "7: During synchronous operation (SynchronizedMotion)".
		Manual control	Positioning control FBs, continuous control FBs, and synchronous control FBs whose type is Enable (Enable) and whose output variable is Execution completion (Done)
		Homing control	Control in which Axis status (AxisName.Md.AxisStatus) is set to "3: During home position return (Homing)" and continuous control is performed for the axis.
Axes group control	Multiple axes control	Positioning control	Control in which Axes group status (AxesGroupName.Md.GroupStatus) is set to "5: Operating (GroupMoving)" and the axis is moved to the target position.

Error processing

If an error occurs while the motion control FB is executed, Error (Error) becomes TRUE and the error code is output to Error code (ErrorID). At this time, for an axis, Axis status (AxisName.Md.AxisStatus) transitions to "1: Stopping on error (ErrorStop)", and for an axes group, Axes group status (AxesGroupName.Md.GroupStatus) transitions to "1: Stopping on error (GroupErrorStop)".

When the axis status of an available axis transits to the "1: Stopping on error (ErrorStop)" status, all buffering FBs are aborted. Error (Error) of the aborted motion control FBs becomes TRUE.

After that, an error reset needs to be executed to start the axis or axes group.

For how to check errors of the axis or axes group and how to reset the error, refer to the following.

User's manual for the controller used

Point

One of the following values is output to Error code (ErrorID). (The value output depends on the control.) However, event codes are not output.

- Axis error code (AxisName.Md.ErrorID)
- Axes group error code (AxesGroupName.Md.ErrorID)
- Latest motion system error code (MotionSystem.Md.ErrorID)

Units used in control

Units for the position, velocity, acceleration/deceleration, and jerk that are used in the motion system follow the unit of the axis to be used.

The following types exist for the control values, such as the position and the velocity, that are used in the motion system.

For details on the axis technical units and control values related to the position and the velocity, refer to the following.

User's manual for the controller used

Type	Description
Commanded value	A value (target value) based on an input to the motion control FB. (Such as commanded position and commanded velocity)
Set value	The current control value that is generated by motion operation. (Such as set position and set velocity)
Actual value	A value obtained by converting the actual value received from the device on the real axis to the unit of the axis. (Such as actual position and actual velocity)

I/O variables used in motion control FBs

This section describes the I/O variables used in motion control FBs.
For motion control FBs, the I/O variables, input variables, and output variables must be defined.

I/O variables

These variables set the axis and axes group of the drive unit to be controlled, such as Axis information (Axis) and Axes group information (AxesGroup).
Axes and axes groups are generated/initialized by setting axis settings and the axes group setting in the engineering tool.
The set axis and axes group are assigned as an axis variable or an axes group variable in the global label data.
For details on the axis and axes groups, refer to the following.
📖 User's manual for the controller used

Input variable

These variables set operation conditions such as the target position and the command velocity.

Output variable

These variables output the motion control FB status, drive unit status, whether there is an error, or other status.

Class

The following are the classes of the I/O variables, input variables, and output variables.

Variable	Class
I/O variable	VAR_IN_OUT
Input variable	VAR_INPUT
Output variable	VAR_OUTPUT

Data types

Variables are classified into different types based on their bit length, processing method, value range, or others.
For details on data types, refer to the following.
☞ Page 28 Data Specification Method

Omission of input arguments

When omitting motion control FB inputs, the default value that is defined for each motion control FB is applied. For details on the default values, refer to the section for the motion control FB to be used.
When omitting inputs such as the velocity in the multiple start motion control FB, the input value of the previous motion control FB is carried over.

Refresh timing of inputs/outputs

Each argument of the motion control FB is refreshed at the call timing of the motion control FB. To control the input/output of the motion control FB synchronizing with the operation cycle, call the motion control FB from a fixed cycle program whose cycle is the same as the operation cycle.

Precautions for the motion control FB

- I/O arguments of the motion control FB are refreshed at the call timing of the motion control FB, but the cycle that the control is actually executed in differs depending on the motion control FB. For example, motion FBs perform the control at the operation cycle regardless of the motion control FB call task (scan/fixed cycle/event). For details, refer to the section for the motion control FB to be used.
- The timing for refreshing the output arguments of the motion control FBs differs from the timing for refreshing axis variables and other related labels. For this reason, the timing chart provided for each function may not always match the actual signal changes. Create a program considering the refresh cycle of each signal.
- For motion control FBs in operation (Executing (Busy) is TRUE), call the motion control FB. If the motion control FB is not called into the program, control continues. However, changes to the input variable values will not be reflected in control, and it will no longer be possible to update output variables such as errors and Execution Completion.
- Do not use the motion control FB in an initial execution type program as an FB call may stop while the motion control FB is in operation.
- When used between the FOR and NEXT instructions or in combination with a control syntax (IF statement, FOR statement, and CASE statement), check Executing (Busy) and Execution completion (Done) so that the motion control FB is called before the execution of the motion control FB is complete.
- The motion control FB imports input values when Execution command (Execute) or Enable (Enable) is set to TRUE. Therefore, set input values before setting Execution command (Execute) or Enable (Enable) to TRUE. In addition, when changing multiple input arguments by a retrigger/continuous update of the motion control FB, change them by the same task as the motion control FB call task to match the import timing of each argument.
- When changing Execution command (Execute) to FALSE after it has been set to TRUE, wait until after Executing (Busy) has changed to the TRUE status before setting Execution command (Execute) to FALSE.
- The motion control uses real number data (floating-point data) for the axis current position, motion control FB target position, or others. Therefore, it may include a truncation error. For example, when repeatedly executing positioning in relative position specification, such as with MC_MoveRelative (Relative Value Positioning), the truncation error may cumulate between the set position and the specified movement value. If the truncation error is a problem, execute positioning in absolute position specification, such as with MC_MoveAbsolute (Absolute Value Positioning). In addition, the operation error that occurs in the real number data processing within the function can be suppressed by setting real number type parameters related to motion control so that there is no number after the decimal point. For details on the real number data (floating-point data) used in motion control FBs, refer to the following.

☞ Page 38 Real number data (floating-point data)

- Do not allow one motion control FB instance to be called from multiple programs (programs to be executed in parallel, including interrupt programs) at the same time. Operation when an instance is called at the same time cannot be guaranteed.

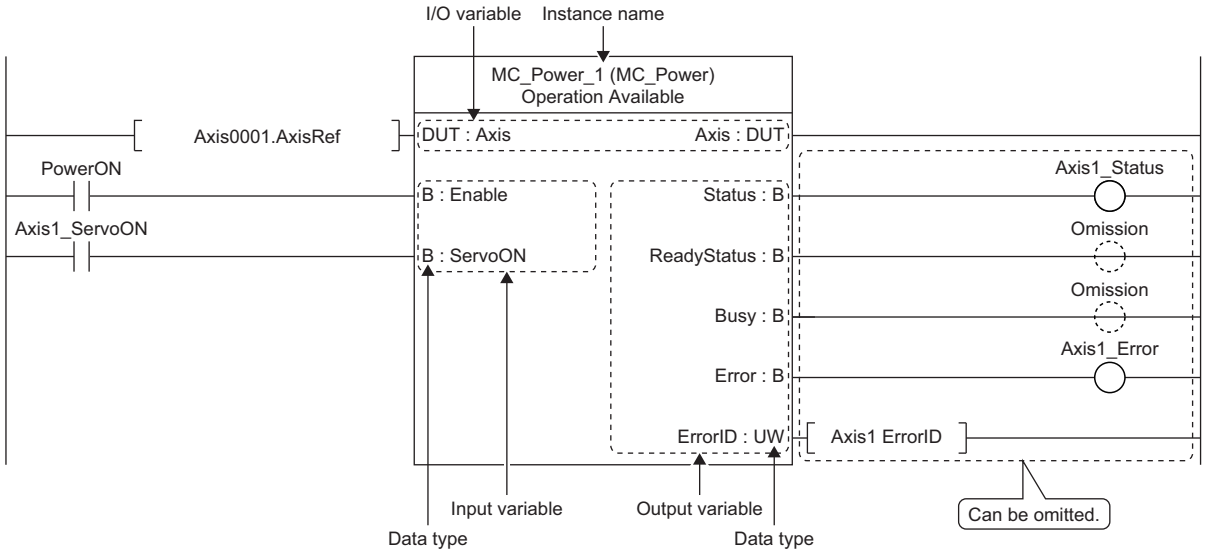
39.2 Motion Control FB Configuration

This section describes how to write each program language for the motion control FB.

Ladder

In the ladder language, the motion control FB is arranged as follows.

Ex.
If MC_Power (Operation Available)



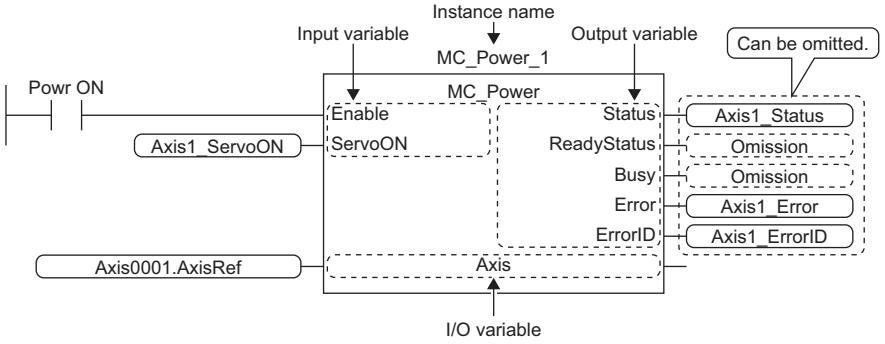
Name	Description
Instance name	The instance name assigned to each FB. The instance name can be changed.
I/O variable	These variables set the axis variable names of the drive unit (or other devices) to be controlled, such as Axis information (Axis) and Axes group information (AxesGroup).
Input variable	These variables set operation conditions such as the target position and the command velocity. The setting of these variables can be omitted. When omitted, the default value is used.
Output variable	These variables output the FB status and drive unit status.
Data type	Displays the data type of I/O variables, input variables, and output variables. For details on the data type of I/O variables, input variables, and output variables, refer to the following. ☞ Page 28 Data Specification Method

FBD/LD (Function block diagram/ladder)

In the FBD/LD language, the motion control FB is arranged as follows.

Ex.

If MC_Power (Operation Available)



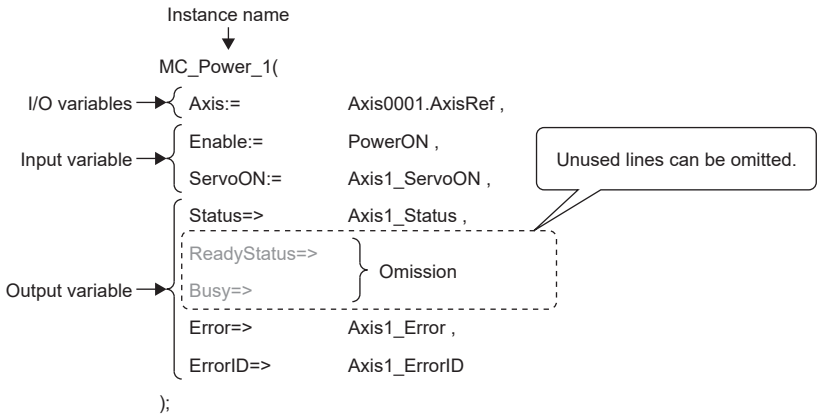
Name	Description
Instance name	The instance name assigned to each FB. The instance name can be changed.
I/O variable	These variables set the axis variable names of the drive unit (or other devices) to be controlled, such as Axis information (Axis) and Axes group information (AxesGroup).
Input variable	These variables set operation conditions such as the target position and the command velocity. The setting of these variables can be omitted. When omitted, the default value is used.
Output variable	These variables output the FB status and drive unit status.

ST (Structured text)

In the ST language, the motion control FB is arranged as follows.

Ex.

If MC_Power (Operation Available)



Name	Description
Instance name	The instance name assigned to each FB. The instance name can be changed.
I/O variable	These variables set the axis variable names of the drive unit (or other devices) to be controlled, such as Axis information (Axis) and Axes group information (AxesGroup).
Input variable	These variables set operation conditions such as the target position and the command velocity. The setting of these variables can be omitted. When omitted, the default value is used.
Output variable	These variables output the FB status and drive unit status.

40 LIST OF MOTION CONTROL FUNCTION BLOCK VARIABLES

This chapter describes the list of variables used in motion control FBs.

40.1 List of Variables

The following types of variables are handled in motion control.

Variable		Description	Reference
Axis variable		Variables for axes configured with parameter data such as the axis type or monitor data such as the axis current position.	☞ Page 1398 Axis variables
Axes group variable		Variables for axes groups configured with parameter data such as structuring axes or monitor data such as the axis group status.	☞ Page 1423 Axes group variables
System variable		Variables for systems configured with parameter data relating to add-ons or monitor data such as the operation cycle.	☞ Page 1430 System variables
Advanced synchronous control variable	Input axis variable	Variables for input axes configured with parameter data for advanced synchronous control input axes or monitor data such as the axis current position	☞ Page 1434 Input axis variable
	Output axis variable	Variables for output axes configured with parameter data for advanced synchronous control output axes or monitor data such as the axis current position	☞ Page 1436 Output axis variable
	System variable	Variables for systems configured with parameter data relating to advanced synchronous control add-ons or monitor data	☞ Page 1441 System variable (Advanced synchronous control)
	Other structure variable	Variables for structures used in advanced synchronous control that are classified outside those listed above	☞ Page 1462 Other structure variable (Advanced synchronous control)
Other structure variable		Variables for structures in a classification not listed above.	☞ Page 1442 Other structure variable

Point

In this manual, variables listed with a portion of the variable underlined indicates that the variable name is defined by the user.

AxisName.AxisRef.AxisNo

↓

A variable to be defined by the user

How to read the list of variables

Each item in the list of variables has the following meaning.

■Import

Item	Description
Operation cycle	The timing for executing each operation cycle
Normal	The timing for executing normal tasks of the motion system
Service	The timing for executing the motion service processing control
Immediately	When an event occurs
At start	When the FB starts
At stop	When the FB stops
At system start	When the motion system starts
When connecting device	When a device is connected
Ready ON	When "PLC Ready (MotionSystem.Cd.SequenceReady)" is turned on
System basic cycle	The timing for executing the system basic cycle processing
Operation cycle (GroupEnable)	The timing for executing each operation cycle (GroupEnable)
When axes group is valid	The timing when the axes group becomes valid
Operation cycle (FB is starting)	The timing for executing operation cycle processing (FB is starting)
When the 0th point of cam data is passed	When the current position per cycle passes the 0th point of the cam data currently being executed (example) The timing of when the 0th point position was passed for cam resolution 512

■Attribute

Display	Description
LIST_WRITE_BACK	Label list (Write back to initial value possible)
LIST_READ_ONLY	Label list (Write back to initial value not possible)

Axis variables

Axis variables are assigned the following data types based on the axis type.

Axis type	Data type
Real drive axis	AXIS_REAL
Real encoder axis	AXIS_ENCODER
Virtual drive axis	AXIS_VIRTUAL
Virtual encoder axis	AXIS_VIRTUAL_ENCODER
Virtual link axis	AXIS_VIRTUAL_LINK

Point

The axis variables that can be set differ depending on the axis type used. (☞ Page 1418 Variables settable as the axis variable for each axis type)

Each axis data type has the following members.

Member name	Data type*1*2	Description	Reference
AxisRef	AXIS_REF	The data structure for input/output of the motion control FBs. The type is fixed regardless of the axis type.	☞ Page 1398 AxisName.AxisRef. (Axis information)
PrConst	AXIS_□_PRM_CONST	Stores the axis parameter data (constant). Opens the setting value at the axis variable initialization. Reload to the control is not executed after the axis variable initialization.	☞ Page 1399 AxisName.PrConst. (Axis parameter constant)
Pr	AXIS_□_PRM	Stores the axis parameter data. Opens the default value at the axis variable initialization. Reload to the control is executed after the axis variable initialization. The fetch timing to the control changes depending on the parameter.	☞ Page 1403 AxisName.Pr. (Axis parameter)
Md	AXIS_□_MONI	Stores the axis monitor data. Executes refresh in the fixed cycle for each monitor data.	☞ Page 1408 AxisName.Md. (Axis monitor data)
Cd	AXIS_□_CMD	Stores the axis control command data. Acquires the latest value every control operation cycle and uses it for the control.	☞ Page 1417 AxisName.Cd. (Axis control data)

*1 □: The data type of each axis type

*2 The members of the data type differ depending on the axis type.

AxisName.AxisRef. (Axis information)

Variable name	Name	Import	Data type	Attribute	Description
AxisNo	Axis No.	—	WORD(UINT)	LIST_WRITE_BACK	The axis No. is set. • 0: Not set • 1 to 10000: Set axis No.
StartIO	I/O No.	—	WORD(HEX)	LIST_WRITE_BACK	The I/O No. is set. By default, 3E0H is automatically input. • 000H to 0FFH: The start I/O number (the first three digits when expressed in four-digit hexadecimal)

AxisName.PrConst. (Axis parameter constant)

Variable name	Name	Import	Data type	Attribute	Description
AddressOfStation	Station address setting	At system start	WSTRING(63)	LIST_WRITE_BACK	The network address (IP address) of the device station is set with a string. When multiple logic axes are included in one station such as with a multi-axis drive unit, set multi-drop numbers to identify certain logic axes. ■For a single axis device 192.168.3.1 IP address ■For a multi-axis device Example: C-axis of MR-J5W-G 192.168.3.1#2 Multidrop number IP address *: The multi-drop No. is set as "# + No. (decimal)". • #0: A-axis • #1: B-axis • #2: C-axis *: The multi-drop No. can be omitted. When it is omitted, it is regarded as "#0".
AxisType	Axis type setting	At system start	INT (MC_AXIS_TYPE) ☞ Page 1477 MC_AXIS_TYPE	LIST_WRITE_BACK	The axis type is set. • 0: Real Drive Axis (DriveAxis) • 2: Real Encoder Axis (EncoderAxis) • 3: Virtual drive axis (VirtualDriveAxis) • 4: Virtual encoder axis (VirtualEncoderAxis) • 5: Virtual link axis (VirtualLinkAxis)
Encoder_AxisType	Real encoder axis type setting	At system start	INT (MC_ENCODER_AXIS_TYPE) ☞ Page 1477 MC_ENCODER_AXIS_TYPE	LIST_WRITE_BACK	The type of the real encoder axis is set. • 1: Via Drive Unit (Drive)
Encoder_CounterDisableSignal	Counter disabling signal	At system start	SIGNAL_SELECT ☞ Page 1447 SIGNAL_SELECT (Signal select)	LIST_WRITE_BACK	The signal to switch to the counter disable is set. Settings and operations specific to Counter disabling signal (Encoder_CounterDisableSignal) are shown below. If a value outside the range is set, Out of Parameter Range (Axis) (error code: 1A60H) occurs. ■Target (Target) • Data type can only be set to [VAR], [DEV], and [CONST]. • For the network synchronous communication setting of the device station referred to when [VAR] or [DEV] is specified, only "Synchronize" can be specified. ■Signal detection method (Detection) Allows detection at the following setting level only. • 0: Detection at TRUE (HighLevel) • 1: Detection at FALSE (LowLevel) ■Compensation time (CompensationTime) Can only be set to 0.0[s]. ■Filter time (FilterTime) The setting range is "0.0 to 5.0 [s]".
Encoder_RingCout_LowerValue	Encoder ring counter lower limit value	At system start	DINT	LIST_WRITE_BACK	The encoder ring counter lower limit value is set. ■One-word PosActualValue • -32768 to 32767 If a value outside the one-word range is set, Out of Parameter Range (Axis) (error code: 1A60H) occurs. ■Two-word PosActualValue, or if omitted • -2147483648 to 2147483647
Encoder_RingCout_UpperValue	Encoder ring counter upper limit value	At system start	DINT	LIST_WRITE_BACK	The encoder ring upper limit value is set. ■One-word PosActualValue • -32768 to 32767 *:If a value outside the one-word range is set, Out of Parameter Range (Axis) (error code: 1A60H) occurs. ■Two-word PosActualValue, or if omitted • -2147483648 to 2147483647

Variable name	Name	Import	Data type	Attribute	Description
HwStrokeLimit_FlsSignal	Upper limit signal	At system start	SIGNAL_SELECT ☞ Page 1447 SIGNAL_SELECT (Signal select)	LIST_WRITE_BACK	The signal to use the upper limit signal (FLS) is set. Depending on the variables used as external signals and the status of devices managed by the controller, the signal may be treated as detected. Settings and operations specific to Upper limit signal (HwStrokeLimit_FlsSignal) are shown below. ■I/O number (StartIO) Ignores the input value. ■Target (Target) If Target is not specified, signal disabled is determined and the signal non-detection status is always set. Moreover, Hardware Stroke Limit Override (AxisName.Md.HwStrokeLimit_Override) is not imported. The data type that can be specified is only [VAR], [DEV], or [CONST]. * If an unusable value is set, Out of Parameter Range (Axis) (error code: 1A60H) occurs. The cause can be checked with the following detail codes. • An incorrect string was set (detail code: 0001H) • An unavailable data type was set (detail code: 0002H) • An unavailable type was specified (detail code: 0003H) • A type other than the numerical value type was specified at label specification (detail code: 0004H) • A signal that does not exist was set (detail code: 0005H) • A global label where access from an external device is disabled was set (detail code: 0005H) • The number of characters exceeded the upper limit (detail code: 0007H) • "Asynchronous" was set in the network synchronous communication setting of the device station being referred to (detail code: 8002H) • An unsupported device was set (detail code: 8002H) • A device number out of range was set (detail code: 8002H) ■Signal detection method (Detection) Allows detection at the following specification level only. • 0: Detection at TRUE (HighLevel) • 1: Detection at FALSE (LowLevel) * If a value outside the range is specified, Out of Parameter Range (Axis) (error code: 1A60H) (detail code: 8003H) occurs. ■Compensation time (CompensationTime) Ignores the input value. ■Filter time (FilterTime) The setting range is 0.0 to +5.0 [s]. * If a value outside the range is specified, Out of Filter Time Setting Range of Each Axis Signal Warning (event code: 00D24H) occurs, and the axis operates with 0.0 of the filter time.

Variable name	Name	Import	Data type	Attribute	Description
HwStrokeLimit_RlsSignal	Lower limit signal	At system start	SIGNAL_SELECT ☞ Page 1447 SIGNAL_SELECT (Signal select)	LIST_WRITE_BACK	The signal to use the lower limit signal (RLS) is set. Depending on the variables used as external signals and the status of devices managed by the controller, the signal may be treated as detected. Settings and operations specific to Lower limit signal (HwStrokeLimit_RlsSignal) are shown below. ■I/O number (StartIO) Ignores the input value. ■Target (Target) If Target is not specified, signal disabled is determined and the signal non-detection status is always set. Also, control data is not imported. The data type that can be specified is only [VAR], [DEV], or [CONST]. * If an unusable value is set, Out of Parameter Range (Axis) (error code: 1A60H) occurs. The cause can be checked with the following detail codes. • An incorrect string was set (detail code: 0001H) • An unavailable data type was set (detail code: 0002H) • An unavailable type was specified (detail code: 0003H) • A type other than the numerical value type was specified at label specification (detail code: 0004H) • A signal that does not exist was set (detail code: 0005H) • A global label where access from an external device is disabled was set (detail code: 0005H) • The number of characters exceeded the upper limit (detail code: 0007H) • "Asynchronous" was set in the network synchronous communication setting of the device station being referred to (detail code: 8002H) • An unsupported device was set (detail code: 8002H) • A device number out of range was set (detail code: 8002H) ■Signal detection method (Detection) Allows detection at the following specification level only. • 0: Detection at TRUE (HighLevel) • 1: Detection at FALSE (LowLevel) * If a value outside the range is specified, Out of Parameter Range (Axis) (error code: 1A60H) (detail code: 8003H) occurs. ■Compensation time (CompensationTime) Ignores the input value. ■Filter time (FilterTime) The setting range is 0.0 to +5.0 [s]. * If a value outside the range is specified, Out of Filter Time Setting Range of Each Axis Signal Warning (event code: 00D24H) occurs, and the axis operates with 0.0 of the filter time.
OperationCycle	Control cycle setting	At system start	INT	LIST_WRITE_BACK	■Real axis An operation cycle equal to the communication cycle of the device station assigned to the axis is automatically selected. If no applicable operation cycle exists, Cycle assignment incorrect (axis) (error code: 1B2BH) occurs. • 0: Automatic setting ■Virtual axis or real axis where no station address has been set • 0: Operate in the first operation cycle. • 1: Operate in the first operation cycle. • 2: Operate in the second operation cycle • 3: Operate in the third operation cycle
PosRestoration_AbsPosEnable	Absolute position control setting	At system start	INT (MC_ABS_SYSTEM) ☞ Page 1477 MC_ABS_SYSTEM	LIST_WRITE_BACK	The absolute position control is set. • 0: Disable absolute position system (ABSDisabled) • 1: Enable absolute position system (Enabled) • -1: Automatic Setting (Acquire from Connected Device) (Auto)

Variable name	Name	Import	Data type	Attribute	Description
PosRestoration_RestorationBase	Current position restoration reference position setting	At system start	INT (MC_POS_RESTORATION_BASE) ☞ Page 1479 MC_POS_RESTORATION_BASE	LIST_WRITE_BACK	When using the absolute position system, the current position restoration reference position is set. • 0: Backup position • 1: Home position • -1: Automatic determination
RingCount_Enable	Ring counter enabled selection	At system start	BOOL	LIST_WRITE_BACK	Whether to enable/disable the ring counter is set. • FALSE: Disabled • TRUE: Enabled
RingCount_LowerValue	Ring counter lower limit value	At system start	LREAL	LIST_WRITE_BACK	The ring counter lower limit value is set. • -10000000000.0 to 10000000000.0 Not imported when the ring counter is disabled
RingCount_UpperValue	Ring counter upper limit value	At system start	LREAL	LIST_WRITE_BACK	The ring counter upper limit value is set. • -10000000000.0 to 10000000000.0 Not imported when the ring counter is disabled
SlaveEmulate_Enable	Axis emulation enabled	At system start	BOOL	LIST_WRITE_BACK	Whether the axis is used as an emulate axis or not is set. • FALSE: Disabled • TRUE: Enabled
SlaveObject	Object data	At system start	SLAVE_OBJECT_REAL ☞ Page 1460 SLAVE_OBJECT_REAL	LIST_WRITE_BACK	The object data is set to the real drive axis.
			SLAVE_OBJECT_VIRTUAL_ENCODER ☞ Page 1461 SLAVE_OBJECT_VIRTUAL_ENCODER		The object data is set to the virtual encoder axis.
TorqueLimit_Max	Torque limit maximum value	At system start	LREAL	LIST_WRITE_BACK	The maximum value that can be specified as the torque limit value is set. (Common in the positive direction and the negative direction) • 0.0 to 1000.0 [%]
TorqueLimit_NegativeInitial	Negative direction torque limit initial value	At system start	LREAL	LIST_WRITE_BACK	The initial value of Negative direction torque limit value (AxisName.Cd.TorqueLimit_Negative) is set. • 0.0 to 1000.0 [%]
TorqueLimit_PositiveInitial	Positive direction torque limit initial value	At system start	LREAL	LIST_WRITE_BACK	The initial value of Positive direction torque limit value (AxisName.Cd.TorqueLimit_Positive) is set. • 0.0 to 1000.0 [%]

AxisName.Pr. (Axis parameter)

Variable name	Name	Import	Data type	Attribute	Description
AccelerationLimit	Acceleration limit value	Ready ON	LREAL	LIST_WRITE_BACK	The acceleration limit value is set. • 0.0000, 0.0001 to 2147483647.0 [U/s^2] The behavior will be as follows according to this setting. • When the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)", limiting of the acceleration is not performed. • When "0.0000" is set, limiting of the acceleration is not performed. • When a positive number less than 0.0001 is set, the value is imported as "0.0000".
AccelerationZeroBehavior	Operation selection at start acceleration/ deceleration 0	Ready ON	INT (MC_ACC_ZERO_MODE) ☞ Page 1478 MC_ACC_ZERO_MODE	LIST_WRITE_BACK	The behavior when "0.0" is set as the acceleration, the deceleration, or the acceleration/deceleration time at start is set. • -1: Error (Not started) (ACCError) • 1: Maximum acceleration/deceleration (MaximumAcceleration)
CmdInPos_Width	Command in-position width	Ready ON	LREAL	LIST_WRITE_BACK	The command in-position width is set. • 0.0: Function disabled • 0.000000001 to 10000000000.0: Function enabled When a positive number less than 0.000000001 is set, the value is imported as "0.0".
DecelerationLimit	Deceleration limit value	Ready ON	LREAL	LIST_WRITE_BACK	The deceleration limit value is set. • 0.0000, 0.0001 to 2147483647.0 [U/s^2] The behavior will be as follows according to this setting. • When the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)", limiting of the acceleration is not performed. • When "0.0000" is set, limiting of the deceleration is not performed. • When a positive number less than 0.0001 is set, the value is imported as "0.0000".
Drive_UnitConvRatioNum	Driver unit conversion numerator	Ready ON	DWORD(UDINT)	LIST_WRITE_BACK	The numerator used to convert the command unit of the motion system to the command unit of the drive unit is set. • 1 to 2147483647
Drive_UnitConvRatioDen	Driver unit conversion denominator	Ready ON	DWORD(UDINT)	LIST_WRITE_BACK	The denominator used to convert the command unit of the motion system to the command unit of the drive unit is set. • 1 to 2147483647

Variable name	Name	Import	Data type	Attribute	Description
ForcedStop_Signal	Forced stop signal	Ready ON	SIGNAL_SELECT ☞ Page 1447 SIGNAL_SELECT (Signal select)	LIST_WRITE_BACK	The signal to use the forced stop is set. Depending on the variables used as external signals and the status of devices managed by the controller, the signal may be treated as detected. For details, refer to the following. Settings and operations specific to Forced stop signal (ForcedStop_Signal) are shown below. ■I/O number (StartIO) Ignores the input value. ■Target (Target) If Target is not specified, signal disabled is determined and the signal non-detection status is always set. The data type that can be specified is only [VAR], [DEV], or [CONST]. * If an unusable value is set, Out of Parameter Range (Axis) (error code: 1A60H) occurs. The cause can be checked with the following detail codes. • An incorrect string was set (detail code: 0001H) • An unavailable data type was set (detail code: 0002H) • An unavailable type was specified (detail code: 0003H) • A type other than the numerical value type was specified at label specification (detail code: 0004H) • A signal that does not exist was set (detail code: 0005H) • A global label where access from an external device is disabled was set (detail code: 0005H) • The number of characters exceeded the upper limit (detail code: 0007H) • "Asynchronous" was set in the network synchronous communication setting of the device station being referred to (detail code: 8002H) • An unsupported device was set (detail code: 8002H) • A device number out of range was set (detail code: 8002H) ■Signal detection method (Detection) Allows detection at the following specification level only. • 0: Detection at TRUE (HighLevel) • 1: Detection at FALSE (LowLevel) * If edge detection is specified, Out of Parameter Range (Axis) (error code: 1A60H) (detail code: 8003H) occurs. ■Compensation time (CompensationTime) Ignores the input value. ■Filter time (FilterTime) The setting range is "0.0 to 5.0 [s]". * If a value outside the range is specified, Out of Filter Time Setting Range of Each Axis Signal Warning (event code: 00D24H) occurs, and the axis operates with 0.0 of the filter time.
Homing_Required	Home position return required or not	Ready ON	BOOL	LIST_WRITE_BACK	Whether homing is required or not is set. When homing is not required, set this variable to FALSE. When set to FALSE, the homing request will not become TRUE at the current position restoration. • FALSE: Home position return not required • TRUE: Home position return required
JerkLimit	Jerk limit value	Ready ON	LREAL	LIST_WRITE_BACK	The jerk limit value is set. • 0.0000, 0.0001 to 2147483647.0 [U/s^3] The behavior will be as follows according to this setting. • When "0.0000" is set, limiting of the jerk is not performed. • When a positive number less than 0.0001 is set, the value is imported as "0.0000".
OverrunOperation	Operation setting at overrun	Ready ON	INT (MC_DECEL_STOP_MODE) ☞ Page 1477 MC_DECEL_STOP_MODE	LIST_WRITE_BACK	The operation performed when the target position is exceeded during positioning operation is set. • 1: Immediate stop (ImmediateStop) • 2: Keep current acceleration/deceleration (KeepCurrentAcc)
StartableAtUnhomed	Start permission at home position return uncompleted	Ready ON	BOOL	LIST_WRITE_BACK	Whether axis start is allowed or not when homing is incomplete is set. • FALSE: Disabled • TRUE: Enabled

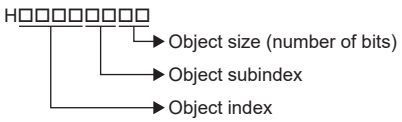
Variable name	Name	Import	Data type	Attribute	Description
StopMode_Deceleration	Deceleration at stop	Ready ON	LREAL	LIST_WRITE_BACK	The deceleration/deceleration time at deceleration stop by stop cause occurrence is set. ■When using acceleration/deceleration method specifying acceleration/deceleration (acceleration/deceleration specification method) • 0.0000, 0.0001 to 2147483647.0 [U/s^2] ■When using acceleration/deceleration method specifying acceleration/deceleration time (acceleration/deceleration time-fixed method) • 0.000000, 0.000001 to 8400.0 [s] The behavior will be as follows according to this setting. • If this variable is set to "0.0", the axis stops immediately regardless of the setting of Operation selection at start acceleration/deceleration 0 (AccelerationZeroBehavior). • If a stop cause occurs, clamping by the upper limit value is performed when the axis goes out of the range beyond the upper limit value, and the event is treated as "0 (Immediate stop)" when the axis goes out of the range beyond the lower limit value.
StopMode_DecelerationCurve	Stop selection at deceleration to stop	Ready ON	INT (MC_STOP_CURVE_MODE) ☞ Page 1477 MC_STOP_CURVE_MODE	LIST_WRITE_BACK	The operation performed when a stop cause occurs during deceleration (including a stop cause and automatic deceleration) is set. • 1: Recreate deceleration curve (OverrideCurve)
StopMode_General	Stop selection at stop by factors	Ready ON	INT (MC_DECEL_STOP_MODE) ☞ Page 1477 MC_DECEL_STOP_MODE	LIST_WRITE_BACK	The operation performed when a stop cause occurs is set. • 1: Immediate stop (ImmediateStop) • 2: Keep current acceleration/deceleration (KeepCurrentAcc) • 3: Alternative acceleration/deceleration (AlternativeAcc) Immediate stop will be executed when executing an FB without deceleration specification.
StopMode_HwStrokeLimit	Stop selection at hardware stroke limit error occurrence	Ready ON	INT (MC_DECEL_STOP_MODE) ☞ Page 1477 MC_DECEL_STOP_MODE	LIST_WRITE_BACK	The operation performed when a hardware stroke limit error occurs is set. • 1: Immediate stop (ImmediateStop) • 2: Keep current acceleration/deceleration (KeepCurrentAcc) • 3: Alternative acceleration/deceleration (AlternativeAcc) Immediate stop will be executed when executing an FB without deceleration specification.
StopMode_ServoOff	Process selection at servo OFF command during operation	Ready ON	INT (MC_DECEL_STOP_MODE) ☞ Page 1477 MC_DECEL_STOP_MODE	LIST_WRITE_BACK	The operation performed when the servo OFF state is established during operation is set. • 0: Ignore (Ignore) • 4: Servo OFF after immediate stop (ServoOffAfterImmediateStop) • 5: Servo OFF after deceleration to stop (ServoOffAfterDecelStop)
StopMode_SwStrokeLimit	Stop selection at software stroke limit error occurrence	Ready ON	INT (MC_DECEL_STOP_MODE) ☞ Page 1477 MC_DECEL_STOP_MODE	LIST_WRITE_BACK	The operation performed when a software stroke limit error occurs is set. • 1: Immediate stop (ImmediateStop) • 2: Keep current acceleration/deceleration (KeepCurrentAcc) • 3: Alternative acceleration/deceleration (AlternativeAcc) Immediate stop will be executed when executing an FB without deceleration specification.
StopOption_DriverTargetIgnored	Driver command discard detection setting	Ready ON	BOOL	LIST_WRITE_BACK	Whether to detect the command discard status of the driver module and stop at an error during operation of the axis or not is set. • FALSE: Detection disabled • TRUE: Detection enabled

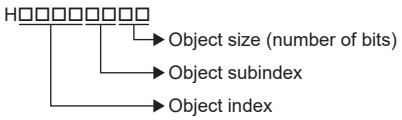
Variable name	Name	Import	Data type	Attribute	Description
StopSignal	Stop signal	Ready ON	SIGNAL_SELECT ☞ Page 1447 SIGNAL_SELECT (Signal select)	LIST_WRITE_BACK	The signal to use the stop signal (STOP) of the external input signal is set. Depending on the variables used as external signals and the status of devices managed by the controller, the signal may be treated as detected. Settings and operations specific to Stop signal (StopSignal) are shown below. ■I/O number (StartIO) Ignores the input value. ■Target (Target) If Target is not specified, signal disabled is determined and the signal non-detection status is always set. The data type that can be specified is only [VAR], [DEV], or [CONST]. * If an unusable value is set, Out of Parameter Range (Axis) (error code: 1A60H) occurs. The cause can be checked with the following detail codes. • An incorrect string was set (detail code: 0001H) • An unavailable data type was set (detail code: 0002H) • An unavailable type was specified (detail code: 0003H) • A type other than the numerical value type was specified at label specification (detail code: 0004H) • A signal that does not exist was set (detail code: 0005H) • A global label where access from an external device is disabled was set (detail code: 0005H) • The number of characters exceeded the upper limit (detail code: 0007H) • "Asynchronous" was set in the network synchronous communication setting of the device station being referred to (detail code: 8002H) • An unsupported device was set (detail code: 8002H) • A device number out of range was set (detail code: 8002H) ■Signal detection method (Detection) Allows detection at the following specification level only. · 0: Detection at TRUE (HighLevel) · 1: Detection at FALSE (LowLevel) * If edge detection is specified, Out of Parameter Range (Axis) (error code: 1A60H) (detail code: 8003H) occurs. ■Compensation time (CompensationTime) Ignores the input value. ■Filter time (FilterTime) The setting range is "0.0 to 5.0 [s]". * If a value outside the range is specified, Out of Filter Time Setting Range of Each Axis Signal Warning (event code: 00D24H) occurs, and the axis operates with 0.0 of the filter time.
SwStrokeLimit_Lower	Software stroke limit lower value	Ready ON	LREAL	LIST_WRITE_BACK	The software stroke limit lower value is set. • -10000000000.0 to 10000000000.0
SwStrokeLimit_Target	Software stroke limit target	Ready ON	INT (MC_POS_SOURCE) ☞ Page 1478 MC_POS_SOURCE	LIST_WRITE_BACK	The target of the software stroke limit is set. • -1: Invalid (Invalid) • 1: Set position (SetPosition) • 3: Feed machine position (FeedMachinePosition)
SwStrokeLimit_Upper	Software stroke limit upper value	Ready ON	LREAL	LIST_WRITE_BACK	The software stroke limit upper value is set. • -10000000000.0 to 10000000000.0
Unit_Position	Position command unit	Ready ON	DWORD(HEX)	LIST_WRITE_BACK	The position command unit to be used in the motion control is set. • 00000000H: pulse • 00010000H: m • FD010000H: mm • FA010000H: μm • F7010000H: nm • 00410000H: degree • FD410000H: $\times 10^{-3}$degree • FA410000H: $\times 10^{-6}$degree • 00B40000H: Revolution • 00C00000H: inch • 00FF0000H: An arbitrary unit

Variable name	Name	Import	Data type	Attribute	Description
Unit_PositionString	Position command unit string	Ready ON	WSTRING(31)	LIST_WRITE_BACK	The command unit to be used in the motion control is set as a string. This variable is set when Position command unit (AxisName.Pr.Unit_Position) is set to "00FF0000H: arbitrary unit".
Unit_Velocity	Velocity command unit	Ready ON	DWORD(HEX)	LIST_WRITE_BACK	The velocity command unit to be used in the motion control is set. • 00000300H: U/s • 00004700H: U/min • FD000300H: U/ms • FA000300H: U/μs • F7000300H: U/ns
VelocityBias	Speed bias value	Ready ON	LREAL	LIST_WRITE_BACK	The speed bias value is set for each axis. • 0.0000, 0.0001 to 2500000000.0 When "0.0000" is set, the speed bias is invalid.
VelocityLimit_Negative	Negative direction speed limit value	Ready ON	LREAL	LIST_WRITE_BACK	The negative direction speed limit value is set. • 0.0001 to 2500000000.0
VelocityLimit_OverOperation	Operation setting at speed limit value exceeded	Ready ON	INT (MC_VELOCITY_LIMIT_MODE) ☞ Page 1476 MC_VELOCITY_LIMIT_MODE	LIST_WRITE_BACK	The operation performed when the speed limit value is exceeded is set. • 0: Ignore (Ignore) • 3: Immediate stop (ImmediateStop) The behavior will be as follows according to this setting. • When "0: Ignore (Ignore)" is set, Speed Limit Value Over Warning (event code: 00D03H) occurs at the time of detecting the command velocity exceeding the speed limit value, and the command exceeding the speed limit value is output as it is. • When "3: Immediate stop (ImmediateStop)" is set, Speed Limit Value Over during Controlling (error code: 1AA4H) occurs at the time of detecting the command velocity exceeding the speed limit value, and the positioning control in execution stops.
VelocityLimit_Positive	Positive direction speed limit value	Ready ON	LREAL	LIST_WRITE_BACK	The positive direction speed limit value is set. • 0.0001 to 2500000000.0

AxisName.Md. (Axis monitor data)

Variable name	Name	Import	Data type	Attribute	Description
AccelerationLimit	Acceleration limit value	Ready ON	LREAL	LIST_READ_ONLY	The acceleration limit value is stored.
AccelerationOverride	Acceleration override factor	Operation cycle	LREAL	LIST_READ_ONLY	The acceleration override factor is stored.
AccelerationZeroBehavior	Operation selection at start acceleration/deceleration 0	Ready ON	INT (MC_ACC_ZERO_MODE) ☞ Page 1478 MC_ACC_ZERO_MODE	LIST_READ_ONLY	The behavior when "0.0" is set as the acceleration, the deceleration, or the acceleration/deceleration time at start is displayed. • -1: Error (Not started) (ACCError) • 1: Maximum acceleration/deceleration (MaximumAcceleration)
ActualPosition	Actual position	Operation cycle	LREAL	LIST_READ_ONLY	The actual current position is stored. The value of "The actual position = the current position × driver unit conversion denominator/numerator" is set. The actual position is a value which is rounded with the ring counter range.
ActualVelocity	Actual velocity	Operation cycle	LREAL	LIST_READ_ONLY	The actual velocity is stored. This velocity is calculated from the difference of the actual position. The value is positive when moving to the positive direction (Address increase direction), and the value is negative when moving to the negative direction (Address decrease direction).
Analyzing	Analyzing	Operation cycle	BOOL	LIST_READ_ONLY	The status of positioning analyzing is displayed. • FALSE: Not analyzing • TRUE: Analyzing
AutoDeceleration	Automatically decelerating	Operation cycle	BOOL	LIST_READ_ONLY	The auto deceleration processing status is displayed. TRUE is stored while the auto deceleration processing is performed. When the multiple start is performed, TRUE is set while the auto deceleration processing is executed during the last positioning point execution. FALSE is set when the control change is performed. During axes group operation, FALSE is stored in Automatically decelerating (AxisName.Md.AutoDeceleration) of the structuring axis. • FALSE: Not during auto deceleration • TRUE: Automatically decelerating
AxisName	Axis name	At system start	WSTRING(127)	LIST_READ_ONLY	The axis name is displayed.
AxisStatus	Axis status	Operation cycle	INT (MC_AXIS_STATUS) ☞ Page 1475 MC_AXIS_STATUS	LIST_READ_ONLY	The axis status is displayed. • -1: Axis variable not initialized/axis parameter error (Invalid) • 0: Axis disabled (Disabled) • 1: Stopping on error (ErrorStop) • 2: Decelerating to stop (Stopping) • 3: During homing (Homing) • 4: Standby (Standstill) • 5: During positioning operation (DiscreteMotion) • 6: During continuous operation (ContinuousMotion) • 7: During synchronous operation (SynchronizedMotion)
BufferingFBs	Number of buffering FBs	At start/at stop	INT	LIST_READ_ONLY	The number of buffering FBs (0 to 2) is displayed.
CmdInPos	Command in-position	Operation cycle	BOOL	LIST_READ_ONLY	This signal turns on when the remaining distance to the target position becomes equal to or less than Command in-position width (AxisName.Md.CmdInPos_Width). • TRUE: Equal to or less than Command in-position width • FALSE: More than Command in-position width
CmdInPos_Width	Command in-position width	Ready ON	LREAL	LIST_READ_ONLY	The command in-position width is displayed. Imports the setting value of Command in-position width (AxisName.Pr.CmdInPos_Width). When Command in-position width (AxisName.Md.CmdInPos_Width) is set to "0.0", the command in-position function of the axis is disabled.

Variable name	Name	Import	Data type	Attribute	Description
CommandedAcceleration	Specified acceleration	Operation cycle	LREAL	LIST_READ_ONLY	The acceleration specified by the user is stored. (Unit: U/s^2) When the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)", this variable stores the specified acceleration/deceleration time. (Unit: s) During axes group operation, "0.0" is stored in Specified acceleration (AxisName.Md.CommandedAcceleration) of the structuring axis.
CommandedDeceleration	Specified deceleration	Operation cycle	LREAL	LIST_READ_ONLY	The deceleration specified by the user is stored. (Unit: U/s^2) When the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)", it is an illegal value. During axes group operation, "0.0" is stored in Specified deceleration (AxisName.Md.CommandedDeceleration) of the structuring axis.
CommandedJerk	Specified jerk	Operation cycle	LREAL	LIST_READ_ONLY	The jerk specified by the user is stored. (Unit: U/s^3) During axes group operation, "0.0" is stored in Specified jerk (AxisName.Md.CommandedJerk) of the structuring axis.
CommandedPosition	Commanded position	Operation cycle	LREAL	LIST_READ_ONLY	The Target position (Position) and Movement amount (Distance) set in the positioning motion control FB being controlled is stored. "0.0" is stored after the positioning is completed.
CommandedVelocity	Commanded velocity	Operation cycle	LREAL	LIST_READ_ONLY	The commanded velocity imported to the motion related motion control FB being controlled is stored. "0.0" is stored after the control is completed. During axes group operation, "0.0" is stored in Commanded velocity (AxisName.Md.CommandedVelocity) of the structuring axis.
Cspr_ForwardSidePosition	Forward-side position at cspr	Operation cycle	LREAL	LIST_READ_ONLY	The forward-side position for cyclic synchronous pressure mode (cspr) is stored.
Cspr_SetPressure	Command pressure at cspr	Operation cycle	DINT	LIST_READ_ONLY	The command pressure for cyclic synchronous pressure mode (cspr) is stored.
Cst_SetTorque	Command current torque at cst/ct	Operation cycle	LREAL	LIST_READ_ONLY	The command torque for cyclic torque mode and continuous operation to torque control mode is stored.
Cst_SetLimitVelocity	Command limit velocity at cst/ct/cspr	Operation cycle	LREAL	LIST_READ_ONLY	The command limit velocity in cyclic synchronous torque mode (cst), continuous operation to torque control mode (ct), or cyclic synchronous pressure mode (cspr) is stored.
Cst_TargetTorque	Target torque at cst/ct	Operation cycle	LREAL	LIST_READ_ONLY	The target torque for cyclic torque mode and continuous operation to torque control mode is stored.
CumulativePosition	Cumulative current position	Operation cycle	LREAL	LIST_READ_ONLY	The current position address is stored. When the current position is changed, the address is changed to the current position change value. The ring address is "-10000000000.0 ≤ cumulative current position < 10000000000.0".
DecelerationLimit	Deceleration limit value	Ready ON	LREAL	LIST_READ_ONLY	The deceleration limit value is stored.
Drive_RPDO[1..64]	RPDO (Motion system → Device) mapping	When connecting device	DWORD(HEX)	LIST_READ_ONLY	In the RPDO mapping settings of the network parameter, the object index specified to the nth place is displayed.  • The object size displays 1 byte, 2 bytes, 4 bytes, and 8 bytes in bits. • When the size of the nth object is 1 byte, a padding object is inserted in the nth+1 place.

Variable name	Name	Import	Data type	Attribute	Description
Drive_TPDO[1..64]	TPDO (Device → Motion system) mapping	When connecting device	DWORD(HEX)	LIST_READ_ONLY	In the TPDO mapping settings of the network parameter, the object index specified to the nth place is displayed.  • The object size displays 1 byte, 2 bytes, 4 bytes, and 8 bytes in bits. • When the size of the nth object is 1 byte, a padding object is inserted in the nth+1 place.
Drive_UnitConvRatioNum	Driver unit conversion numerator	Ready ON	DWORD(UDINT)	LIST_READ_ONLY	The numerator used to convert to the command unit of the drive unit is stored.
Drive_UnitConvRatioDen	Driver unit conversion denominator	Ready ON	DWORD(UDINT)	LIST_READ_ONLY	The denominator used convert to the command unit of the drive unit is stored.
Driver_Mode	Driver control mode	Operation cycle	INT (MC_DRIVE_MODE) ☞ Page 1476 MC_DRIVE_MODE	LIST_READ_ONLY	The drive unit control mode status is displayed. • 6: Homing mode (hm) • 8: Cyclic synchronous position mode (csp) • 9: Cyclic synchronous velocity mode (csv) • 10: Cyclic synchronous torque mode (cst) • -104: Continuous operation to torque control mode (ct) • -111: Cyclic synchronous pressure mode (cspr)
Driver_ReadyOn	Driver ready ON status	Operation cycle	BOOL	LIST_READ_ONLY	The connected drive unit status is displayed. This variable becomes TRUE when Driver status (AxisName.Md.Driver_State) is set to one of the following. • 2: Fault Reaction Active • 5: Switched On • 6: Operation Enable • 7: Quick Stop Active
Driver_ServoOn	Driver servo ON status	Operation cycle	BOOL	LIST_READ_ONLY	The connected drive unit status is displayed. This variable becomes TRUE when Driver status (AxisName.Md.Driver_State) is set to one of the following. • 6: Operation Enable • 7: Quick Stop Active
Driver_State	Driver status	Operation cycle	INT (MC_DRIVE_STATE) ☞ Page 1477 MC_DRIVE_STATE	LIST_READ_ONLY	The drive unit status is displayed. • 0: Not Ready To Switch On • 1: Fault • 2: Fault Reaction Active • 3: Switch On Disabled • 4: Ready To Switch On • 5: Switched On • 6: Operation Enable • 7: Quick Stop Active • -1: Invalid
DriverError	Drive unit error detection	Immediately	BOOL	LIST_READ_ONLY	The drive unit error status is displayed. • FALSE: No drive unit error • TRUE: Drive unit error present
DriverErrorID	Drive unit error code	Immediately	WORD(HEX)	LIST_READ_ONLY	The upper 16 bits of "Current alarm (2A41H)" of the object data is displayed when a drive unit error occurs.
DriverErrorDetailID	Drive unit error detail code	Immediately	WORD(HEX)	LIST_READ_ONLY	The lower 16 bits of "Current alarm (2A41H)" of the object data is stored when a drive unit error occurs.
Encoder_Connected	Connection status	Operation cycle	BOOL	LIST_READ_ONLY	The virtual encoder axis connection status is displayed. • FALSE: Not connected • TRUE: Connected
Encoder_CounterDisable	Counter disabled	Operation cycle	BOOL	LIST_READ_ONLY	The status of the input from the encoder is displayed. • FALSE: Counter enabled • TRUE: Counter disabled
Error	Axis error detection	Immediately	BOOL	LIST_READ_ONLY	The axis error status is displayed. • FALSE: No axis error • TRUE: Axis error present
ErrorID	Axis error code	Immediately	WORD(HEX)	LIST_READ_ONLY	The error code of the axis error occurred is displayed.

Variable name	Name	Import	Data type	Attribute	Description
FeedMachinePosition	Feed machine position	Operation cycle	LREAL	LIST_READ_ONLY	The current position address by the machine coordinate is stored. The cumulative value from the homing complete position is set. The address is not changed even if the current position is changed. The ring address is "-10000000000.0 ≤ feed function position < 10000000000.0".
FollowupDisable	Follow-up disabled	Operation cycle	BOOL	LIST_READ_ONLY	The follow-up disable status is displayed. When it is disabled, the follow-up is not performed. • FALSE: Follow-up enabled • TRUE: Follow-up disabled
ForcedStop_Released	Forced stop cancelling	Operation cycle	BOOL	LIST_READ_ONLY	The forced stop cancel status is displayed. • FALSE: Forced stop • TRUE: Forced stop release
ForcedStop_Signal	Forced stop signal	Ready ON	SIGNAL_SELECT ☞ Page 1447 SIGNAL_SELECT (Signal select)	LIST_READ_ONLY	The input status of the forced stop signal is displayed. Outputs specific to Forced stop signal (ForcedStop_Signal) are shown below. ■I/O number (StartIO) "0" is always displayed. ■Target (Target) The fetch result of the parameter is displayed. ■Signal detection method (Detection) The fetch result of the parameter is displayed. ■Compensation time (CompensationTime) "0.0" is always displayed. ■Filter time (FilterTime) The fetch result of the parameter is displayed.
Homing_Complete	Home position return completed	Immediately	BOOL	LIST_READ_ONLY	Whether homing is completed normally or not is displayed. This variable becomes FALSE when operation starts or when homing is required. • FALSE: Homing uncompleted • TRUE: Home position return completed
Homing_Request	Home position return request	Immediately	BOOL	LIST_READ_ONLY	Whether homing is required or not is displayed. This variable becomes FALSE when homing is completed. • FALSE: No homing request • TRUE: Homing request
Homing_Required	Home position return required or not	Ready ON	BOOL	LIST_READ_ONLY	Whether homing is required for the axis or not is displayed. If set to FALSE, the homing request will not become TRUE at the current position restoration. • FALSE: Home position return not required • TRUE: Home position return required
Homing_Status	Home position return operation status	Operation cycle	WORD(HEX)	LIST_READ_ONLY	The homing status of the drive unit is displayed. • 0000H: Homing procedure in progress • 0001H: Homing procedure interrupted or not started • 0002H: Homing attained (target not reached) • 0003H: Homing procedure completed successfully • 0004H: Homing error occurred (velocity other than 0) • 0005H: Homing error occurred (velocity is 0) • FFFFH: Homing procedure is not in progress
HwStrokeLimit_FlsStatus	Upper limit signal status	Operation cycle	BOOL	LIST_READ_ONLY	The detection status of the upper limit signal (FLS) is displayed. • FALSE: Not detecting the upper limit signal (FLS) • TRUE: Detecting the upper limit signal (FLS) The detection state, which reflects the signal detection method or the filter time but not the signal input state, is displayed.
HwStrokeLimit_Override	Hardware stroke limit override	At start	STRING(15)	LIST_READ_ONLY	The acceptance status of the enabled/disabled switching command of the hardware stroke limit check is displayed. • DISABLE: Check disabled • ONLY_INSIDE: Check disabled for movement returning to the range • Others: No disable request

Variable name	Name	Import	Data type	Attribute	Description
HwStrokeLimit_RlsStatus	Lower limit signal status	Operation cycle	BOOL	LIST_READ_ONLY	The detection status of the lower limit signal (RLS) is displayed. • FALSE: Not detecting the lower limit signal (RLS) • TRUE: Detecting the lower limit signal (RLS) The detection state, which reflects the signal detection method or the filter time but not the signal input state, is displayed.
InVelocity	Target velocity reached	Operation cycle	BOOL	LIST_READ_ONLY	Whether the set velocity has been reached or not is displayed. During axes group operation, FALSE is displayed in Target velocity reached (AxisName.Md.InVelocity) of the structuring axis. • FALSE: Not reached • TRUE: Reached
Io_PosActualValue	Object data_PosActualValue	Operation cycle	DINT	LIST_READ_ONLY	The value of object data Position actual value (6064H) is displayed. (The feedback of position to be received from the device) For a real encoder axis or virtual encoder axis, the input pulse value [encoder pulse unit] acquired from the encoder is displayed.
Io_PosEncoderResolution	Object data_PosEncoderResolution	Operation cycle	DWORD(UDINT)	LIST_READ_ONLY	The value of object data Position encoder resolution (608FH) is displayed. For a real encoder axis or virtual encoder axis, the synchronous encoder resolution is displayed.
Io_PressureActualValue	Object data_PressureActualValue	Operation cycle	DINT	LIST_READ_ONLY	The value of object data Pressure actual value (2E52H) is displayed.
Io_Statusword	Object data_Statusword	Operation cycle	WORD(HEX)	LIST_READ_ONLY	The value of object data Statusword (6041H) is displayed. For a real encoder axis, the encoder status is emulated and displayed on the motion system side.
Io_TargetPos	Object data_TargetPos	Operation cycle	DINT	LIST_READ_ONLY	The value of object data Target position (607AH) is displayed. (The position command to be sent to the device station) For a virtual drive axis, a value which is converted into the command value in a driver unit from the cumulative current position is displayed.
Io_TargetVelocity	Object data_TargetVelocity	Operation cycle	DINT	LIST_READ_ONLY	The value of object data Target Velocity (60FFH) is displayed. (The velocity command to be sent to the device station)
Io_TorqueActualValue	Object data_TorqueActualValue	Operation cycle	INT	LIST_READ_ONLY	The value of object data Torque actual value (6077H) is displayed. (The feedback torque to be received from the device station)
Io_VelActualValue	Object data_VelActualValue	Operation cycle	DINT	LIST_READ_ONLY	The value of object data Velocity actual value (606CH) is displayed. (The feedback velocity to be received from the device station)
JerkLimit	Jerk limit value	Ready ON	LREAL	LIST_READ_ONLY	The jerk limit value is stored.
OperationCycle	Control cycle	At system start	INT	LIST_READ_ONLY	The control operation cycle of the axis is stored. • 0: Axis not initialized/Control cycle setting out-of-range • 1: Operate in the first operation cycle • 2: Operate in the second operation cycle • 3: Operate in the third operation cycle
OverrunOperation	Operation setting at overrun	Ready ON	INT (MC_DECEL_STOP_MODE) ☞ Page 1477 MC_DECEL_STOP_MODE	LIST_READ_ONLY	The input status of the overrun operation setting is displayed. • 1: Immediate stop (ImmediateStop) • 2: Keep current acceleration/deceleration (KeepCurrentAcc)

Variable name	Name	Import	Data type	Attribute	Description
PosRestoration_Status	Current position restoration status	Service	INT (MC_POS_RESTORATION_STATUS) ☞ Page 1478 MC_POS_RESTORATION_STATUS	LIST_READ_ONLY	The current position restoration status is displayed. When the current position restoration request needs to be executed, the status becomes "1: Waiting for Current Position Restoration Request (WaitingRequest)" when a device station is connected. The status becomes "0: Not Executed (NotExecute)" when the device is disconnected. • 0: Current Position Restoration Not Executed (NotExecute) • 1: Waiting for Current Position Restoration Request (WaitingRequest) • 2: Restoration Completed in Incremental System (RestoredInIncSystem) • 3: Restoration Completed in Absolute Position System (Homing Uncompleted) (RestoredInAbsSystemUnHomed) • 4: Restoration Completed in Absolute Position System (RestoredInAbsSystem)
ProfileID	Execution profile ID No.	Operation cycle	WORD(UINT)	LIST_READ_ONLY	The profile ID currently being executed is stored. The value is updated with the following timing. ■The specified CamTableID (cam table ID) is stored. • Timing when MC_CamIn is executed and the slave axis status becomes "7: During synchronous operation (SynchronizedMotion)" • When CamTableID (cam table ID) is changed ■The value is cleared to 0. • When Periodic (Periodic) is "FALSE: Non periodic", Options bit 16 (Specifying operation after one cycle operation) is off (end), and EndOfProfile is TRUE • When a stop cause occurs
SetAcceleration	Set current acceleration	Operation cycle	LREAL	LIST_READ_ONLY	The set output acceleration (the acceleration/deceleration calculated from the difference of the set velocity) is stored.*1 (Unit: U/s^2) The acceleration/deceleration status differs depending on the movement direction and the value of Set acceleration (AxisName.Md.SetAcceleration). ■Positive movement direction (address increasing direction) • 0.0: During stop or fixed-speed • Plus sign: During acceleration • Minus sign: During deceleration ■Negative movement direction (address decreasing direction) • 0.0: During stop or fixed-speed • Plus sign: During deceleration • Minus sign: During acceleration "0.0" is stored when the target velocity is reached. During axes group operation, the set acceleration of the structuring axis is stored.
SetPosition	Set position	Operation cycle	LREAL	LIST_READ_ONLY	The current position address is stored.*1 The set position is a value equal to the cumulative current value rounded with the ring counter range.
SetVelocity	Set velocity	Operation cycle	LREAL	LIST_READ_ONLY	The set output velocity is stored.*1 This velocity is calculated from the difference of the set position. The value is positive when moving to the positive direction (Address increase direction), and the value is negative when moving to the negative direction (Address decrease direction). During axes group operation, the set velocity of the structuring axis is stored.
SlaveEmulate_Enable	Axis emulating	Service	BOOL	LIST_READ_ONLY	Whether driver emulate operation is valid or invalid is displayed. • FALSE: Disabled • TRUE: Enabled
StartableAtUnhomed	Start permission at home position return uncompleted	Ready ON	BOOL	LIST_READ_ONLY	The start status when homing is incomplete is displayed. • FALSE: Disabled • TRUE: Enabled
StopMode_Deceleration	Deceleration at stop	Ready ON	LREAL	LIST_READ_ONLY	The input status of deceleration at stop is displayed.

Variable name	Name	Import	Data type	Attribute	Description
StopMode_DecelerationCurve	Stop selection at deceleration to stop	Ready ON	INT (MC_STOP_CURVE_MODE) ☞Page 1477 MC_STOP_CURVE_MODE	LIST_READ_ONLY	The input status of stop selection at deceleration to stop is displayed. • 1: Recreate deceleration curve (OverrideCurve)
StopMode_General	Stop selection at stop by factors	Ready ON	INT (MC_DECEL_STOP_MODE) ☞Page 1477 MC_DECEL_STOP_MODE	LIST_READ_ONLY	The input status of stop selection at stop by factors is displayed. • 1: Immediate stop (ImmediateStop) • 2: Keep current acceleration/deceleration (KeepCurrentAcc) • 3: Alternative acceleration/deceleration (AlternativeAcc)
StopMode_HwStrokeLimit	Stop selection at hardware stroke limit error occurrence	Ready ON	INT (MC_DECEL_STOP_MODE) ☞Page 1477 MC_DECEL_STOP_MODE	LIST_READ_ONLY	The input status of stop selection at the hardware stroke limit error occurrence is displayed. • 1: Immediate stop (ImmediateStop) • 2: Keep current acceleration/deceleration (KeepCurrentAcc) • 3: Alternative acceleration/deceleration (AlternativeAcc)
StopMode_ServoOff	Process selection at servo OFF command during operation	Ready ON	INT (MC_DECEL_STOP_MODE) ☞Page 1477 MC_DECEL_STOP_MODE	LIST_READ_ONLY	The input status of process selection at servo OFF command during operation is displayed. • 0: Ignore (Ignore) • 4: Servo OFF after immediate stop (ServoOffAfterImmediateStop) • 5: Servo OFF after deceleration to stop (ServoOffAfterDecelStop)
StopMode_SwStrokeLimit	Stop selection at software stroke limit error occurrence	Ready ON	INT (MC_DECEL_STOP_MODE) ☞Page 1477 MC_DECEL_STOP_MODE	LIST_READ_ONLY	The input status of stop selection at software stroke limit error occurrence is displayed. • 1: Immediate stop (ImmediateStop) • 2: Keep current acceleration/deceleration (KeepCurrentAcc) • 3: Alternative acceleration/deceleration (AlternativeAcc)
StopOption_DriverTargetIgnored	Driver command discard detection setting	Ready ON	BOOL	LIST_READ_ONLY	The status of driver command discard detection setting is displayed. • FALSE: Detection disabled • TRUE: Detection enabled
StopSignal	Stop signal	Ready ON	SIGNAL_SELECT ☞Page 1447 SIGNAL_SELECT (Signal select)	LIST_READ_ONLY	The input status of the stop signal (STOP) of the external input signal is displayed. Settings and operations specific to Stop signal (StopSignal) are shown below. ■I/O number (StartIO) "0" is always displayed. ■Target (Target) The fetch result of the parameter is displayed. ■Signal detection method (Detection) The fetch result of the parameter is displayed. ■Compensation time (CompensationTime) "0.0" is always displayed. ■Filter time (FilterTime) The fetch result of the parameter is displayed.
StopStatus	Stop status	Operation cycle	BOOL	LIST_READ_ONLY	The input status of the stop signal (STOP) of the external input signal is displayed. • FALSE: Stop processing released • TRUE: During stop processing
SwStrokeLimit_Lower	Software stroke limit lower value	Ready ON	LREAL	LIST_READ_ONLY	The software stroke limit lower value is displayed.
SwStrokeLimit_Override	Software stroke limit override	At start	STRING(15)	LIST_READ_ONLY	The software stroke limit valid/invalid status is displayed. • DISABLE: Check disabled • ONLY_INSIDE: Check disabled for movement returning to the range • Others: Check enabled (no disable request)
SwStrokeLimit_Target	Software stroke limit target	Ready ON	INT (MC_POS_SOURCE) ☞Page 1478 MC_POS_SOURCE	LIST_READ_ONLY	The software stroke limit target is displayed. • -1: Invalid (Invalid) • 1: Set position (SetPosition) • 3: Feed machine position (FeedMachinePosition)

Variable name	Name	Import	Data type	Attribute	Description
SwStrokeLimit_Upper	Software stroke limit upper value	Ready ON	LREAL	LIST_READ_ONLY	The software stroke limit upper value is displayed.
TargetAcceleration	Target acceleration	Operation cycle	LREAL	LIST_READ_ONLY	The maximum acceleration used during acceleration is stored. (Unit: U/s^2) During axes group operation, "0.0" is stored in Target acceleration (AxisName.Md.TargetAcceleration) of the structuring axis.
TargetDeceleration	Target deceleration	Operation cycle	LREAL	LIST_READ_ONLY	The maximum deceleration used during deceleration is stored. (Unit: U/s^2) During axes group operation, "0.0" is stored in Target deceleration (AxisName.Md.TargetDeceleration) of the structuring axis.
TargetVelocity	Target velocity	Operation cycle	LREAL	LIST_READ_ONLY	The actual target velocity whose override and velocity limit value are considered is stored. "0.0" is stored after the control is completed. During axes group operation, "0.0" is stored in Target velocity (AxisName.Md.TargetVelocity) of the structuring axis. ■For positioning control The value becomes more than "0.0" regardless of the movement direction. ■For other than positioning control The value is positive when moving to the positive direction (Address increase direction), and the value is negative when moving to the negative direction (Address decrease direction).
TorqueLimit_Negative	Negative direction torque limit value	Operation cycle	LREAL	LIST_READ_ONLY	The negative direction torque limit value that is currently valid is displayed.
TorqueLimit_Positive	Positive direction torque limit value	Operation cycle	LREAL	LIST_READ_ONLY	The positive direction torque limit value that is currently valid is displayed.
Unit_Position	Position command unit	Ready ON	DWORD(HEX)	LIST_READ_ONLY	The position command unit to be used in the motion control is displayed. • 00000000H: pulse • 00010000H: m • FD010000H: mm • FA010000H: μm • F7010000H: nm • 00410000H: degree • FD410000H: ×10^{-3}degree • FA410000H: ×10^{-6}degree • 00B40000H: Revolution • 00C00000H: inch • 00FF0000H: An arbitrary unit
Unit_PositionDisplay	Position command unit display	Ready ON	WSTRING(31)	LIST_READ_ONLY	The currently controlled position command unit is displayed as a string.
Unit_PositionString	Position command unit string	Ready ON	WSTRING(31)	LIST_READ_ONLY	When Position command unit (AxisName.Pr.Unit_Position) is set to "00FF0000H: arbitrary unit", the currently controlled position command unit set in Position command unit string (AxisName.Pr.Unit_PositionString) is displayed as a string.
Unit_Velocity	Velocity command unit	Ready ON	DWORD(HEX)	LIST_READ_ONLY	The velocity command unit to be used in the motion control is displayed. • 00000300H: U/s • 00004700H: U/min • FD000300H: U/ms • FA000300H: U/μs • F7000300H: U/ns
Unit_VelocityDisplay	Velocity command unit display	Ready ON	WSTRING(31)	LIST_READ_ONLY	The currently controlled velocity command unit is output as a string.
UseInGroup	Using axes group	Operation cycle	BOOL	LIST_READ_ONLY	Whether the specified axis is being used in the axes group or not is displayed. • FALSE: Not used • TRUE: Used
VelocityBias	Speed bias value	Ready ON	LREAL	LIST_READ_ONLY	The setting value for Speed bias value (AxisName.Pr.VelocityBias) that was imported during Ready ON is stored.

Variable name	Name	Import	Data type	Attribute	Description
VelocityLimit_Negative	Negative direction speed limit value	Ready ON	LREAL	LIST_READ_ONLY	The address decreasing direction speed limit value is stored.
VelocityLimit_OverOperation	Operation setting at speed limit value exceeded	Ready ON	INT (MC_VELOCITY_LIMIT_MODE) ☞ Page 1476 MC_VELOCITY_LIMIT_MODE	LIST_READ_ONLY	The operation performed when the speed limit value is exceeded during control is displayed. • 0: Ignore (Ignore) • 3: Immediate stop (ImmediateStop)
VelocityLimit_Positive	Positive direction speed limit value	Ready ON	LREAL	LIST_READ_ONLY	The address increasing direction speed limit value is stored.
VelocityOverride	Velocity override factor	Operation cycle	LREAL	LIST_READ_ONLY	The velocity override factor is displayed.
Warning	Axis warning detection	Immediately	BOOL	LIST_READ_ONLY	The axis warning status is displayed. • FALSE: No axis warning • TRUE: Axis warning present
WarningID	Axis warning code	Immediately	WORD(HEX)	LIST_READ_ONLY	The event code is stored when an axis warning occurs.

*1 The stored value includes an error because a floating-point error occurs.

AxisName.Cd. (Axis control data)

Variable name	Name	Import	Data type	Attribute	Range
AccelerationOverride	Acceleration override factor	Operation cycle	LREAL	LIST_READ_ONLY	The acceleration override factor is set. • 0.01 to 10.00
Encoder_Connect	Connection command	Operation cycle	BOOL	LIST_READ_ONLY	The connection status of the virtual encoder axis input is switched between valid and invalid. • FALSE: Disabled • TRUE: Enabled
Encoder_CounterDisable	Counter disabled	Operation cycle	BOOL	LIST_READ_ONLY	The signal to switch to the counter disable is set. • FALSE: Not executed • TRUE: Execute the counter disable request
Encoder_InputValue	Encoder input value	Operation cycle	DINT	LIST_READ_ONLY	The values used as the virtual encoder axis input values are sequentially set.
ErrorReset	Axis error reset	Service	BOOL	LIST_READ_ONLY	Axis errors/warnings and drive unit errors are reset. This variable becomes FALSE when the error reset is completed. • FALSE: Not executed • TRUE: Executes the error reset
FollowupDisable	Follow-up disabled	Operation cycle	BOOL	LIST_READ_ONLY	The follow-up is disabled. • FALSE: Not executed • TRUE: Executes the followup disable request
Homing_ClearRequest	Home position return request clear	Operation cycle	BOOL	LIST_READ_ONLY	The homing request is cleared. Home position return request clear (Homing_ClearRequest) becomes FALSE automatically after Home position return request (Homing_Request) is set to FALSE. • FALSE: Not executed • TRUE: Executes the homing request clear
HwStrokeLimit_Override	Hardware stroke limit override	At start	STRING(15)	LIST_READ_ONLY	The hardware stroke limit valid/invalid status is temporarily switched. • DISABLE: Check disabled • ONLY_INSIDE: Check disabled for movement returning to the range • Others: No disable request
SwStrokeLimit_Override	Software stroke limit override	At start	STRING(15)	LIST_READ_ONLY	The software stroke limit valid/invalid status is temporarily switched. • DISABLE: Check disabled • ONLY_INSIDE: Check disabled for movement returning to the range • Others: Check enabled (no disable request)
TorqueLimit_Negative	Negative direction torque limit value	Operation cycle	LREAL	LIST_READ_ONLY	The negative direction torque limit value is set. • 0.0 to 1000.0 [%]
TorqueLimit_Positive	Positive direction torque limit value	Operation cycle	LREAL	LIST_READ_ONLY	The positive direction torque limit value is set. • 0.0 to 1000.0 [%]
VelocityOverride	Velocity override factor	Operation cycle	LREAL	LIST_READ_ONLY	The velocity override factor is set. • 0.00 to 10.00

Variables settable as the axis variable for each axis type

The variables that can be set and used as the axis variable differ for each axis type.

The variables that can be set for each axis type are shown below.

■AxisName.AxisRef. (Axis information)

○: Can be set, ×: Cannot be set

Variable name	Name	Axis type				
		Real drive axis	Real encoder axis	Virtual drive axis	Virtual encoder axis	Virtual link axis
AxisNo	Axis No.	○	○	○	○	○
StartIO	I/O No.	○	○	○	○	○

■AxisName.PrConst. (Axis parameter constant)

○: Can be set, ×: Cannot be set

Variable name	Name	Axis type				
		Real drive axis	Real encoder axis	Virtual drive axis	Virtual encoder axis	Virtual link axis
AddressOfStation	Station address setting	○	○	×	×	×
AxisType	Axis type setting	○	○	○	○	○
Encoder_AxisType	Real encoder axis type setting	×	○	×	×	×
Encoder_CounterDisableSignal	Counter disabling signal	×	○	×	○	×
Encoder_RingCout_LowerValue	Encoder ring counter lower limit value	×	×	×	○	×
Encoder_RingCout_UpperValue	Encoder ring counter upper limit value	×	×	×	○	×
HwStrokeLimit_FlsSignal	Upper limit signal	○	×	○	×	×
HwStrokeLimit_RlsSignal	Lower limit signal	○	×	○	×	×
OperationCycle	Control cycle setting	○	○	○	○	○
PosRestoration_AbsPosEnable	Absolute position control setting	○	○	○	○	○
PosRestoration_RestorationBase	Current position restoration reference position setting	○	×	×	×	×
RingCount_Enable	Ring counter enabled selection	○	○	○	○	○
RingCount_LowerValue	Ring counter lower limit value	○	○	○	○	○
RingCount_UpperValue	Ring counter upper limit value	○	○	○	○	○
SlaveEmulate_Enable	Axis emulation enabled	○	×	×	×	×
SlaveObject	Object data	○	×	×	○	×
TorqueLimit_Max	Torque limit maximum value	○	×	×	×	×
TorqueLimit_NegativeInitial	Negative direction torque limit initial value	○	×	×	×	×
TorqueLimit_PositiveInitial	Positive direction torque limit initial value	○	×	×	×	×

■AxisName.Pr. (Axis parameter)

○: Can be set, ×: Cannot be set

Variable name	Name	Axis type				
		Real drive axis	Real encoder axis	Virtual drive axis	Virtual encoder axis	Virtual link axis
AccelerationLimit	Acceleration limit value	○	×	○	×	×
AccelerationZeroBehavior	Operation selection at start acceleration/deceleration 0	○	×	○	×	×
CmdInPos_Width	Command in-position width	○	×	○	×	×
DecelerationLimit	Deceleration limit value	○	×	○	×	×
Drive_UnitConvRatioNum	Driver unit conversion numerator	○	○	×	○	×
Drive_UnitConvRatioDen	Driver unit conversion denominator	○	○	×	○	×
ForcedStop_Signal	Forced stop signal	○	×	○	×	×
Homing_Required	Home position return required or not	○	○	○	○	○
JerkLimit	Jerk limit value	○	×	○	×	×
OverrunOperation	Operation setting at overrun	○	×	○	×	×
StartableAtUnhomed	Start permission at home position return uncompleted	○	○	○	○	○
StopMode_Deceleration	Deceleration at stop	○	×	○	×	×
StopMode_DecelerationCurve	Stop selection at deceleration to stop	○	×	○	×	×
StopMode_General	Stop selection at stop by factors	○	×	○	×	×
StopMode_HwStrokeLimit	Stop selection at hardware stroke limit error occurrence	○	×	○	×	×
StopMode_ServoOff	Process selection at servo OFF command during operation	○	×	○	×	×
StopMode_SwStrokeLimit	Stop selection at software stroke limit error occurrence	○	×	○	×	×
StopOption_DriverTargetIgnored	Driver command discard detection setting	○	×	×	×	×
StopSignal	Stop signal	○	×	○	×	×
SwStrokeLimit_Lower	Software stroke limit lower value	○	×	○	×	×
SwStrokeLimit_Target	Software stroke limit target	○	×	○	×	×
SwStrokeLimit_Upper	Software stroke limit upper value	○	×	○	×	×
Unit_Position	Position command unit	○	○	○	○	○
Unit_PositionString	Position command unit string	○	○	○	○	○
Unit_Velocity	Velocity command unit	○	○	○	○	○
VelocityBias	Speed bias value	○	×	○	×	○
VelocityLimit_Negative	Negative direction speed limit value	○	×	○	×	×
VelocityLimit_OverOperation	Operation setting at speed limit value exceeded	○	×	○	×	×
VelocityLimit_Positive	Positive direction speed limit value	○	×	○	×	×

■AxisName.Md. (Axis monitor data)

○: Can be set, ×: Cannot be set

Variable name	Name	Axis type				
		Real drive axis	Real encoder axis	Virtual drive axis	Virtual encoder axis	Virtual link axis
AccelerationLimit	Acceleration limit value	○	×	○	×	×
AccelerationOverride	Acceleration override factor	○	×	○	×	×
AccelerationZeroBehavior	Operation selection at start acceleration/deceleration 0	○	×	○	×	×
ActualPosition	Actual position	○	○	×	×	×
ActualVelocity	Actual velocity	○	○	×	×	×
Analyzing	Analyzing	○	○	○	○	○
AutoDeceleration	Automatically decelerating	○	×	○	×	×
AxisName	Axis name	○	○	○	○	○
AxisStatus	Axis status	○	○	○	○	○
BufferingFBs	Number of buffering FBs	○	×	○	×	×
CmdInPos	Command in-position	○	×	○	×	×
CmdInPos_Width	Command in-position width	○	×	○	×	×
CommandedAcceleration	Specified acceleration	○	×	○	×	×
CommandedDeceleration	Specified deceleration	○	×	○	×	×
CommandedJerk	Specified jerk	○	×	○	×	×
CommandedPosition	Commanded position	○	×	○	×	×
CommandedVelocity	Commanded velocity	○	×	○	×	×
Cspr_ForwardSidePosition	Forward-side position at cspr	○	×	×	×	×
Cspr_SetPressure	Command pressure at cspr	○	×	×	×	×
Cst_SetTorque	Command current torque at cst/ct	○	×	×	×	×
Cst_TargetTorque	Target torque at cst/ct	○	×	×	×	×
Cst_SetLimitVelocity	Command limit velocity at cst/ct/ cspr	○	×	×	×	×
CumulativePosition	Cumulative current position	○	○	○	○	○
DecelerationLimit	Deceleration limit value	○	×	○	×	×
Drive_RPDO[1..64]	RPDO (Motion system → Device) mapping	○	×	×	×	×
Drive_TPDO[1..64]	TPDO (Device → Motion system) mapping	○	×	×	×	×
Drive_UnitConvRatioNum	Driver unit conversion numerator	○	○	×	×	×
Drive_UnitConvRatioDen	Driver unit conversion denominator	○	○	×	×	×
Driver_Mode	Driver control mode	○	×	×	×	×
Driver_ReadyOn	Driver ready ON status	○	×	×	×	×
Driver_ServoOn	Driver servo ON status	○	×	×	×	×
Driver_State	Driver status	○	×	×	×	×
DriverError	Drive unit error detection	○	×	×	×	×
DriverErrorID	Drive unit error code	○	×	×	×	×
DriverErrorDetailID	Drive unit error detail code	○	×	×	×	×
Encoder_Connected	Connection status	×	×	×	○	×
Encoder_CounterDisable	Counter disabled	×	○	×	○	×
Error	Axis error detection	○	○	○	○	○
ErrorID	Axis error code	○	○	○	○	○
FeedMachinePosition	Feed machine position	○	×	×	×	×
FollowupDisable	Follow-up disabled	○	×	×	×	×
ForcedStop_Released	Forced stop cancelling	○	×	○	×	×
ForcedStop_Signal	Forced stop signal	○	×	○	×	×
Homing_Complete	Home position return completed	○	○	○	○	○
Homing_Request	Home position return request	○	○	○	○	○

Variable name	Name	Axis type				
		Real drive axis	Real encoder axis	Virtual drive axis	Virtual encoder axis	Virtual link axis
Homing_Required	Home position return required or not	○	○	○	○	○
Homing_Status	Home position return operation status	○	×	×	×	×
HwStrokeLimit_FlsStatus	Upper limit signal status	○	×	○	×	×
HwStrokeLimit_Override	Hardware stroke limit override	○	×	○	×	×
HwStrokeLimit_RlsStatus	Lower limit signal status	○	×	○	×	×
InVelocity	Target velocity reached	○	×	○	×	×
Io_PosActualValue	Object data_PosActualValue	○	○	×	○	×
Io_PosEncoderResolution	Object data_PosEncoderResolution	×	○	×	○	×
Io_PressureActualValue	Object data_PressureActualValue	○	×	×	×	×
Io_Statusword	Object data_Statusword	○	○	×	×	×
Io_TargetPos	Object data_TargetPos	○	×	○	×	×
Io_TargetVelocity	Object data_TargetVelocity	○	×	×	×	×
Io_TorqueActualValue	Object data_TorqueActualValue	○	×	×	×	×
Io_VelActualValue	Object data_VelActualValue	○	×	×	×	×
JerkLimit	Jerk limit value	○	×	○	×	×
OperationCycle	Control cycle	○	○	○	○	○
OverrunOperation	Operation setting at overrun	○	×	○	×	×
PosRestoration_Status	Current position restoration status	○	○	○	○	○
ProfileID	Execution profile ID No.	○	×	○	×	○
SetAcceleration	Set current acceleration	○	×	○	×	×
SetPosition	Set position	○	○	○	○	○
SetVelocity	Set velocity	○	×	○	○	×
SlaveEmulate_Enable	Axis emulating	○	×	×	×	×
StartableAtUnhomed	Start permission at home position return uncompleted	○	○	○	○	○
StopMode_Deceleration	Deceleration at stop	○	×	○	×	×
StopMode_DecelerationCurve	Stop selection at deceleration to stop	○	×	○	×	×
StopMode_General	Stop selection at stop by factors	○	×	○	×	×
StopMode_HwStrokeLimit	Stop selection at hardware stroke limit error occurrence	○	×	○	×	×
StopMode_ServoOff	Process selection at servo OFF command during operation	○	×	○	×	×
StopMode_SwStrokeLimit	Stop selection at software stroke limit error occurrence	○	×	○	×	×
StopOption_DriverTargetIgnored	Driver command discard detection setting	○	×	×	×	×
StopSignal	Stop signal	○	×	○	×	×
StopStatus	Stop status	○	×	○	×	×
SwStrokeLimit_Lower	Software stroke limit lower value	○	×	○	×	×
SwStrokeLimit_Override	Software stroke limit override	○	×	○	×	×
SwStrokeLimit_Target	Software stroke limit target	○	×	○	×	×
SwStrokeLimit_Upper	Software stroke limit upper value	○	×	○	×	×
TargetAcceleration	Target acceleration	○	×	○	×	×
TargetDeceleration	Target deceleration	○	×	○	×	×
TargetVelocity	Target velocity	○	×	○	×	×
TorqueLimit_Negative	Negative direction torque limit value	○	×	×	×	×
TorqueLimit_Positive	Positive direction torque limit value	○	×	×	×	×
Unit_Position	Position command unit	○	○	○	○	○

Variable name	Name	Axis type				
		Real drive axis	Real encoder axis	Virtual drive axis	Virtual encoder axis	Virtual link axis
Unit_PositionDisplay	Position command unit display	○	○	○	○	○
Unit_PositionString	Position command unit string	○	○	○	○	○
Unit_Velocity	Velocity command unit	○	○	○	○	○
Unit_VelocityDisplay	Velocity command unit display	○	○	○	○	○
UseInGroup	Using axes group	○	○	○	○	○
VelocityBias	Speed bias value	○	×	○	×	○
VelocityLimit_Negative	Negative direction speed limit value	○	×	○	×	×
VelocityLimit_OverOperation	Operation setting at speed limit value exceeded	○	×	○	×	×
VelocityLimit_Positive	Positive direction speed limit value	○	×	○	×	×
VelocityOverride	Velocity override factor	○	×	○	×	×
Warning	Axis warning detection	○	○	○	○	○
WarningID	Axis warning code	○	○	○	○	○

■AxisName.Cd. (Axis control data)

○: Can be set, ×: Cannot be set

Variable name	Name	Axis type				
		Real drive axis	Real encoder axis	Virtual drive axis	Virtual encoder axis	Virtual link axis
AccelerationOverride	Acceleration override factor	○	×	○	×	×
Encoder_Connect	Connection command	×	×	×	○	×
Encoder_CounterDisable	Counter disabled	×	○	×	○	×
Encoder_InputValue	Encoder input value	×	×	×	○	×
ErrorReset	Axis error reset	○	○	○	○	○
FollowupDisable	Follow-up disabled	○	×	×	×	×
Homing_ClearRequest	Home position return request clear	○	○	○	○	○
HwStrokeLimit_Override	Hardware stroke limit override	○	×	○	×	×
SwStrokeLimit_Override	Software stroke limit override	○	×	○	×	×
TorqueLimit_Negative	Negative direction torque limit value	○	×	×	×	×
TorqueLimit_Positive	Positive direction torque limit value	○	×	×	×	×
VelocityOverride	Velocity override factor	○	×	○	×	×

Axes group variables

The data type of axes group variables is expressed as AXES_GROUP.

Each axes group data type has the following members.

Member name	Data type	Description	Reference
AxesGroupRef	AXES_GROUP_REF	The data structure for input/output of the motion control FBs.	☞ Page 1423 AxesGroupName.AxesGroupRef. (Axes group information)
PrConst	AXES_GROUP_PRM_CONST	Stores parameter data (constant) of the axes group. Opens the setting value at the axes group variable initialization. Reload to the control is not executed after the axes group variable initialization.	☞ Page 1423 AxesGroupName.PrConst. (Axes group parameter constant)
Pr	AXES_GROUP_PRM	Stores the parameter data for the axes group. Opens the default value at axes group variable generation. Reload to the control is executed after the axes group initialization. The fetch timing to the control changes depending on the parameter.	☞ Page 1424 AxesGroupName.Pr. (Axes group parameter)
Md	AXES_GROUP_MONI	Stores the monitor data for the axes group. Executes refresh in the fixed cycle for each monitor data.	☞ Page 1426 AxesGroupName.Md. (Axes group monitor data)
Cd	AXES_GROUP_CMD	Stores the command data for axes group control. Acquires the latest value every control operation cycle and uses it for the control.	☞ Page 1429 AxesGroupName.Cd. (Axes group control data)

AxesGroupName.AxesGroupRef. (Axes group information)

Variable name	Name	Import	Data type	Attribute	Description
GroupNo	Axes group No.	—	WORD(UINT)	LIST_WRITE_BACK	The axes group No. is set. • 0: Not set • 1 to 10000: Set axes group No.
StartIO	I/O No.	—	WORD(HEX)	LIST_WRITE_BACK	The I/O No. is set.

AxesGroupName.PrConst. (Axes group parameter constant)

Variable name	Name	Import	Data type	Attribute	Description
AxesGroupType	Axes group type setting	At system start	INT (MC_AXES_GROUP_TYPE) ☞ Page 1478 MC_AXES_GROUP_TYPE	LIST_WRITE_BACK	An axes group type is set. • 0: Standard (Standard)

AxesGroupName.Pr. (Axes group parameter)

Variable name	Name	Import	Data type	Attribute	Description
AccelerationLimit	Acceleration limit value	Ready ON	LREAL	LIST_WRITE_BACK	The acceleration limit value is set. • 0.0000, 0.0001 to 2147483647.0 The behavior will be as follows according to this setting. • When the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)", limiting of the acceleration calculated from the specified acceleration/deceleration time is performed. • When "0.0000" is set, limiting of the acceleration is not performed. • When a positive number less than 0.0001 is set, the value is imported as "0.0000".
AccelerationZeroBehavior	Operation selection at start acceleration/deceleration 0	Ready ON	INT (MC_ACC_ZERO_MODE) ☞ Page 1478 MC_ACC_ZERO_MODE	LIST_WRITE_BACK	The behavior when "0.0" is set as the acceleration, the deceleration, or the acceleration/deceleration time at start is set. • -1: Error (Not started) (ACCError) • 1: Maximum acceleration/deceleration (MaximumAcceleration)
Axis[1..16]	Structuring axis	Ready ON	AXIS_REF ☞ Page 1398 AxisName.AxisRef. (Axis information)	LIST_WRITE_BACK	The Axis No. (AxisNo) of Axis information (AxisName.AxisRef) that configures the axes group is set.
CmdInPos_Width	Command in-position width	Ready ON	LREAL	LIST_WRITE_BACK	The command in-position width on the composite axis of the axes group is set. • 0.0: Function disabled • 0.000000001 to 10000000000.0: Function enabled When a positive number less than 0.000000001 is set, the value is imported as "0.0".
DecelerationLimit	Deceleration limit value	Ready ON	LREAL	LIST_WRITE_BACK	The deceleration limit value is set. • 0.0000, 0.0001 to 2147483647.0 The behavior will be as follows according to this setting. • When the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)", limiting of the deceleration calculated from the specified acceleration/deceleration time is performed. • When "0.0000" is set, limiting of the deceleration is not performed. • When a positive number less than 0.0001 is set, the value is imported as "0.0000".
JerkLimit	Jerk limit value	Ready ON	LREAL	LIST_WRITE_BACK	The jerk limit value is set. • 0.0000, 0.0001 to 2147483647.0 The behavior will be as follows according to this setting. • When "0.0000" is set, limiting of the jerk is not performed. • When a positive number less than 0.0001 is set, the value is imported as "0.0000".
OverrunOperation	Operation setting at overrun	Ready ON	INT (MC_DECEL_STOP_MODE) ☞ Page 1477 MC_DECEL_STOP_MODE	LIST_WRITE_BACK	The operation performed when the target position is exceeded during operation is set. • 1: Immediate stop (ImmediateStop)

Variable name	Name	Import	Data type	Attribute	Description
StopMode_Deceleration	Deceleration at stop	Ready ON	LREAL	LIST_WRITE_BACK	The deceleration/deceleration time at deceleration stop by stop cause occurrence is set. ■When using acceleration/deceleration method specifying acceleration/deceleration (acceleration/deceleration specification method) • 0.0000, 0.0001 to 2147483647.0 ■When using acceleration/deceleration method specifying acceleration/deceleration time (acceleration/deceleration time-fixed method) • 0.000000, 0.000001 to 8400.0 The behavior will be as follows according to this setting. • If this variable is set to "0.0", the axis stops immediately regardless of the setting of Operation selection at start acceleration/deceleration 0 (AccelerationZeroBehavior). • If a stop cause occurs, clamping by the upper limit value is performed when the axis goes out of the range beyond the upper limit value, and the event is treated as "0 (Immediate stop)" when the axis goes out of the range beyond the lower limit value.
StopMode_DecelerationCurve	Stop selection at deceleration to stop	Ready ON	INT (MC_STOP_CURVE_MODE) ☞ Page 1477 MC_STOP_CURVE_MODE	LIST_WRITE_BACK	The operation performed when a stop cause occurs during deceleration (including a stop cause and automatic deceleration) is set. • 1: Recreate deceleration curve (OverrideCurve)
StopMode_ErrorInGroup	Structuring axis operation selection at axis stop by factors	Ready ON	INT (MC_DECEL_STOP_MODE) ☞ Page 1477 MC_DECEL_STOP_MODE	LIST_WRITE_BACK	During axes group operation, when the servo is turned off via the drive unit on the structuring axis and an immediate stop axis error occurs, the operation of the axis on which an error has not occurred is set. • 1: Immediate stop (ImmediateStop)
StopMode_General	Stop selection at stop by factors	Ready ON	INT (MC_DECEL_STOP_MODE) ☞ Page 1477 MC_DECEL_STOP_MODE	LIST_WRITE_BACK	The operation performed when a stop cause occurs is set. • 1: Immediate stop (ImmediateStop) • 2: Keep current acceleration/deceleration (KeepCurrentAcc) • 3: Alternative acceleration/deceleration (AlternativeAcc) Immediate stop will be executed when executing an FB without deceleration specification.
Unit_Position	Position command unit	When axes group is valid	DWORD(HEX)	LIST_WRITE_BACK	The position command unit to be used in the motion control is set. • 00000000H: pulse • 00010000H: m • FD010000H: mm • FA010000H: μm • F7010000H: nm • 00410000H: degree • FD410000H: $\times 10^{-3}$degree • FA410000H: $\times 10^{-6}$degree • 00B40000H: Revolution • 00C00000H: inch • 00FF0000H: An arbitrary unit
Unit_PositionString	Position command unit string	When axes group is valid	WSTRING(31)	LIST_WRITE_BACK	The command unit to be used in the motion control is set as a string. This variable is set when Position command unit (Unit_Position) is set to "00FF0000H: arbitrary unit".
Unit_Velocity	Velocity command unit	When axes group is valid	DWORD(HEX)	LIST_WRITE_BACK	The velocity command unit to be used in the motion control is set. • 00000300H: U/s • 00004700H: U/min • FD000300H: U/ms • FA000300H: U/μs • F7000300H: U/ns
VelocityBias	Speed bias value	Ready ON	LREAL	LIST_WRITE_BACK	The speed bias value is set for each axes group. When "0.0000" is set, the speed bias is invalid. • 0.0000, 0.0001 to 2500000000.0

Variable name	Name	Import	Data type	Attribute	Description
VelocityLimit	Velocity limit value	Ready ON	LREAL	LIST_WRITE_BACK	The velocity limit value (maximum velocity during control) of the axes group is set. • 0.0001 to 2500000000.0

AxesGroupName.Md. (Axes group monitor data)

Variable name	Name	Import	Data type	Attribute	Description
AccelerationLimit	Acceleration limit value	Ready ON	LREAL	LIST_READ_ONLY	The acceleration limit value is stored.
AccelerationOverride	Acceleration override factor	Operation cycle	LREAL	LIST_READ_ONLY	The acceleration override factor is stored.
AccelerationZeroBehavior	Operation selection at start acceleration/deceleration 0	Ready ON	INT (MC_ACC_ZERO_MODE) ☞ Page 1478 MC_ACC_ZERO_MODE	LIST_READ_ONLY	The behavior when "0.0" is set as the acceleration, the deceleration, or the acceleration/deceleration time at start is displayed. • -1: Error (Not started) (ACCError) • 1: Maximum acceleration/deceleration (MaximumAcceleration)
ActualVelocity	Actual velocity	Operation cycle (GroupEnable)	LREAL	LIST_READ_ONLY	The actual velocity is stored. This velocity is the vector speed of the actual velocity of the structuring axes. The value becomes more than "0.0" regardless of the movement direction of the structuring axis. "0.0" is stored when the axes group is invalid.
Analyzing	Analyzing	Operation cycle	BOOL	LIST_READ_ONLY	The status of positioning analyzing of the axes group is displayed. • FALSE: Not analyzing • TRUE: Analyzing
AutoDeceleration	Automatically decelerating	Operation cycle (GroupEnable)	BOOL	LIST_READ_ONLY	The auto deceleration processing status is displayed. TRUE is stored while the auto deceleration processing is being performed. When the multiple start is executed, this variable becomes TRUE while the auto deceleration processing is being performed during the last positioning point execution. This variable becomes FALSE when the control change is performed. • FALSE: Not during auto deceleration • TRUE: Automatically decelerating
Axis[1..16]	Structuring axis	Immediately	AXIS_REF ☞ Page 1398 AxisName.AxisRef. (Axis information)	LIST_READ_ONLY	The Axis No. (AxisNo) of Axis information (AxisName.AxisRef) that configures the axes group is stored.
BufferingFBs	Number of buffering FBs	Operation cycle (GroupEnable)	INT	LIST_READ_ONLY	The number of buffering FBs (0 to 2) is displayed.
CmdInPos	Command in-position	Operation cycle	BOOL	LIST_READ_ONLY	Whether the remaining distance to the target position is equal to or less than Command in-position width (AxesGroupName.Pr.CmdInPos_Width) is displayed. • FALSE: Equal to or more than Command in-position width • TRUE: Less than Command in-position width
CmdInPos_Width	Command in-position width	Ready ON	LREAL	LIST_READ_ONLY	The command in-position width on the composite axis of the axes group is displayed. The value displayed is the imported setting value of Command in-position width (AxesGroupName.Pr.CmdInPos_Width). When the value of Command in-position width (AxesGroupName.Md.CmdInPos_Width) is "0.0", the command in-position function of the axes group is disabled.
CommandedAcceleration	Specified acceleration	Operation cycle (GroupEnable)	LREAL	LIST_READ_ONLY	The acceleration specified by the user is stored. (Unit: U/s^2) When the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)", this variable stores the specified acceleration/deceleration time. (Unit: s)

Variable name	Name	Import	Data type	Attribute	Description
CommandedDeceleration	Specified deceleration	Operation cycle (GroupEnable)	LREAL	LIST_READ_ONLY	The deceleration specified by the user is stored. (Unit: U/s^2) When the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)", it is an illegal value.
CommandedJerk	Specified jerk	Operation cycle (GroupEnable)	LREAL	LIST_READ_ONLY	The jerk specified by the user is stored. (Unit: U/s^3)
CommandedVelocity	Commanded velocity	Operation cycle (GroupEnable)	LREAL	LIST_READ_ONLY	The commanded velocity imported to the motion related motion control FB being controlled is stored. "0.0" is stored after the control is completed. When Speed mode (VelocityMode) is set to Reference axis speed (ReferenceAxisSpeed) or Long axis speed (LongAxisSpeed) for linear interpolation control, this variable stores the commanded velocity of the applicable axis.
DecelerationLimit	Deceleration limit value	Ready ON	LREAL	LIST_READ_ONLY	The deceleration limit value is stored.
Error	Axes group error detection	Immediately	BOOL	LIST_READ_ONLY	The axes group error status is displayed. • FALSE: No axes group error • TRUE: Axes group error present
ErrorID	Axes group error code	Immediately	WORD(HEX)	LIST_READ_ONLY	The error code of the axes group error occurred is displayed.
GroupName	Axes group name	Immediately	WSTRING(127)	LIST_READ_ONLY	The axes group name is stored.
GroupStatus	Axes group status	Operation cycle	INT (MC_AXES_GROUP_STATUS) ☞ Page 1475 MC_AXES_GROUP_STATUS	LIST_READ_ONLY	The axes group status is displayed. • -1: Axes group variable not initialized/Axes group parameter error (Invalid) • 0: Axes group disabled (GroupDisabled) • 1: Stopping on error (GroupErrorStop) • 2: Decelerating to stop (GroupStopping) • 4: Standby (GroupStandby) • 5: Operating (GroupMoving)
InterpolationAxes	Interpolation axes	Operation cycle (GroupEnable)	DWORD(HEX)	LIST_READ_ONLY	The structuring axis that is executing the interpolation control of the axes group is displayed as a bit.
InVelocity	Target velocity reached	Operation cycle (GroupEnable)	BOOL	LIST_READ_ONLY	Whether the set velocity has been reached or not is displayed. • FALSE: Not reached • TRUE: Reached
JerkLimit	Jerk limit value	Ready ON	LREAL	LIST_READ_ONLY	The jerk limit value is stored.
NumberOfAxes	Number of structuring axes	Immediately	INT	LIST_READ_ONLY	The number of structuring axes in the axes group is displayed.
OverrunOperation	Operation setting at overrun	Ready ON	INT (MC_DECEL_STOP_MODE) ☞ Page 1477 MC_DECEL_STOP_MODE	LIST_READ_ONLY	The input status of the overrun operation setting is displayed. • 1: Immediate stop (ImmediateStop)
SetAcceleration	Set current acceleration	Operation cycle (GroupEnable)	LREAL	LIST_READ_ONLY	The set output acceleration (the acceleration/deceleration calculated from the difference of the set velocity) is stored. (Unit: U/s^2) The value is positive when acceleration changes in the positive direction (address increasing direction), and the value is negative when acceleration changes in the negative direction (address decreasing direction). The stored value includes an error because a floating-point error occurs.
SetVelocity	Set velocity	Operation cycle (GroupEnable)	LREAL	LIST_READ_ONLY	During axes group operation, the set output velocity is stored. The value becomes more than "0.0" regardless of the movement direction of the structuring axis. "0.0" is stored after the control is completed. When Speed mode (VelocityMode) is set to Reference axis speed (ReferenceAxisSpeed) or Long axis speed (LongAxisSpeed) for linear interpolation control, this variable stores the absolute value of the set velocity of the applicable axis.

Variable name	Name	Import	Data type	Attribute	Description
StopMode_Deceleration	Deceleration at stop	Ready ON	LREAL	LIST_READ_ONLY	The input status of deceleration at stop is displayed.
StopMode_DecelerationCurve	Stop selection at deceleration to stop	Ready ON	INT (MC_STOP_CURVE_MODE) ☞ Page 1477 MC_STOP_CURVE_MODE	LIST_READ_ONLY	The input status of stop selection at deceleration to stop is displayed. • 1: Recreate deceleration curve (OverrideCurve)
StopMode_ErrorInGroup	Structuring axis operation selection at axis stop by factors	Ready ON	INT (MC_DECEL_STOP_MODE) ☞ Page 1477 MC_DECEL_STOP_MODE	LIST_READ_ONLY	The input status of structuring axis operation selection at axis stop by factors is displayed. • 1: Immediate stop (ImmediateStop)
StopMode_General	Stop selection at stop by factors	Ready ON	INT (MC_DECEL_STOP_MODE) ☞ Page 1477 MC_DECEL_STOP_MODE	LIST_READ_ONLY	The input status of stop selection at stop by factors is displayed. • 1: Immediate stop (ImmediateStop) • 2: Keep current acceleration/deceleration (KeepCurrentAcc) • 3: Alternative acceleration/deceleration (AlternativeAcc)
TargetAcceleration	Target acceleration	Operation cycle (GroupEnable)	LREAL	LIST_READ_ONLY	The maximum acceleration used during acceleration is stored. (Unit: U/s^2)
TargetDeceleration	Target deceleration	Operation cycle (GroupEnable)	LREAL	LIST_READ_ONLY	The maximum deceleration used during deceleration is stored. (Unit: U/s^2)
TargetVelocity	Target velocity	Operation cycle (GroupEnable)	LREAL	LIST_READ_ONLY	The actual target velocity whose override and velocity limit value are considered is stored. The value becomes more than "0.0" regardless of the movement direction of the structuring axis. "0.0" is stored after the control is completed. When Speed mode (VelocityMode) is set to Reference axis speed (ReferenceAxisSpeed) or Long axis speed (LongAxisSpeed) for linear interpolation control, this variable stores the target velocity of the applicable axis.
Unit_Position	Position command unit	When axes group is valid	DWORD(HEX)	LIST_READ_ONLY	The position command unit to be used in the motion control is displayed. • 00000000H: pulse • 00010000H: m • FD010000H: mm • FA010000H: μm • F7010000H: nm • 00410000H: degree • FD410000H: ×10^{-3}degree • FA410000H: ×10^{-6}degree • 00B40000H: Revolution • 00C00000H: inch • 00FF0000H: An arbitrary unit
Unit_PositionDisplay	Position command unit display	When axes group is valid	WSTRING(31)	LIST_READ_ONLY	The currently controlled position command unit is output as a string.
Unit_PositionString	Position command unit string	When axes group is valid	WSTRING(31)	LIST_READ_ONLY	When Position command unit (AxesGroupName.Pr.Unit_Position) is set to "00FF0000H: arbitrary unit", the currently controlled position command unit set in Position command unit string (AxesGroupName.Pr.Unit_PositionString) is displayed as a string.
Unit_Velocity	Velocity command unit	When axes group is valid	DWORD(HEX)	LIST_READ_ONLY	The velocity command unit to be used in the motion control is displayed. • 00000300H: U/s • 00004700H: U/min • FD000300H: U/ms • FA000300H: U/μs • F7000300H: U/ns
Unit_VelocityDisplay	Velocity command unit display	When axes group is valid	WSTRING(31)	LIST_READ_ONLY	The currently controlled velocity command unit is output as a string.

Variable name	Name	Import	Data type	Attribute	Description
VelocityBias	Speed bias value	Ready ON	LREAL	LIST_READ_ONLY	The setting value for Speed bias value (AxisGroupName.Pr.VelocityBias) that was imported during Ready ON is stored.
VelocityLimit	Velocity limit value	Ready ON	LREAL	LIST_READ_ONLY	The velocity limit value of the axes group is stored.
VelocityOverride	Velocity override factor	Operation cycle	LREAL	LIST_READ_ONLY	The velocity override factor is displayed.
Warning	Axes group warning detection	Immediately	BOOL	LIST_READ_ONLY	The axis group warning status is displayed. • FALSE: No axes group warning • TRUE: Axes group warning present
WarningID	Axes group warning code	Immediately	WORD(HEX)	LIST_READ_ONLY	The event code is stored when an axes group warning occurs.

AxesGroupName.Cd. (Axes group control data)

Variable name	Name	Import	Data type	Attribute	Description
AccelerationOverride	Acceleration override factor	Operation cycle	LREAL	LIST_READ_ONLY	The acceleration override factor is set. • 0.01 to 10.00
ErrorReset	Axes group error reset	Service	BOOL	LIST_READ_ONLY	TA reset of axes group errors and warnings is requested. • FALSE: Not executed • TRUE: Executes the error reset
VelocityOverride	Velocity override factor	Operation cycle	LREAL	LIST_READ_ONLY	The velocity override factor is set. • 0.00 to 10.00

System variables

The data type of system variables is expressed as MT_SYSTEM type.

The system data type has the following members.

Member name	Data type	Description	Reference
PrConst	SYS_CONST	Stores the parameter data (constant) for the system. Opens the setting value at system variable initialization. Does not execute reimporting to the control after system variable initialization.	☞ Page 1430 MotionSystem.PrConst. (Motion system parameter constant)
Pr	SYS_PRM	Stores the parameter data for the system. Opens the setting value at system variable initialization. The fetch timing to the control changes depending on the parameter.	☞ Page 1431 MotionSystem.Pr. (Motion system parameter)
Md	SYS_MONI	Stores the monitor data for the system. Executes refresh in the fixed cycle for each monitor data.	☞ Page 1432 MotionSystem.Md. (Motion system monitor data)
Cd	SYS_CMD	Stores the command data for the system. Gets the newest value from each motion service processing control and uses it for control.	☞ Page 1433 MotionSystem.Cd. (Motion system control data)

MotionSystem.PrConst. (Motion system parameter constant)

Variable name	Name	Import	Data type	Attribute	Description
ExcludeWarning	Excluded warning	At system start	STRING(255)	LIST_WRITE_BACK	A warning that should not be detected is set. • "": Not set • "0x1000, 0x1001": Will not detect warnings 0x1000, 0x1001. • "0x1000-0x1010": Will not detect warnings 0x1000 to 0x1010.
OperationCycle[1]	Operation cycle setting	At system start	CYCLE_PARAM ☞ Page 1448 CYCLE_PARAM	LIST_WRITE_BACK	The first operation cycle setting value is set. Setting and operations specific to Operation cycle setting (OperationCycle[1]) are shown below. ■Cycle setting (Cycle) • 0: Synchronizes with the basic cycle of network
OperationCycle[2]	Operation cycle setting	At system start	CYCLE_PARAM ☞ Page 1448 CYCLE_PARAM	LIST_WRITE_BACK	The second operation cycle setting value is set. Setting and operations specific to Operation cycle setting (OperationCycle[2]) are shown below. ■Cycle setting (Cycle) • 0: Synchronizes with the middle speed cycle of network
OperationCycle[3]	Operation cycle setting	At system start	CYCLE_PARAM ☞ Page 1448 CYCLE_PARAM	LIST_WRITE_BACK	The third operation cycle setting value is set. Setting and operations specific to Operation cycle setting (OperationCycle[3]) are shown below. ■Cycle setting (Cycle) • 0: Synchronizes with the low speed cycle of network
ProfileRootDrive	Operation profile data root drive	At system start	WSTRING(7)	LIST_WRITE_BACK	The save destination drive of the operation profile data is set. ■Setting range • Data memory (drive 4): /rom • SD memory card (drive 2): /sdc
SequenceReadyInterlock	PLC Ready Interlock Selection	At system start	BOOL	LIST_WRITE_BACK	Selects the method to turn on "PLC Ready (MotionSystem.Cd.SequenceReady)". (☞ Page 1433 MotionSystem.Cd. (Motion system control data)) • TRUE: Interlocking with the CPU function part • FALSE: Not interlocking with the CPU function part

MotionSystem.Pr. (Motion system parameter)

Variable name	Name	Import	Data type	Attribute	Description
ForcedStop_Signal	All axes forced stop signal	Ready ON	SIGNAL_SELECT ☞ Page 1447 SIGNAL_SELECT (Signal select)	LIST_WRITE_BACK	The signal to use the all axes forced stop is set. Depending on the variables used as external signals and the status of devices managed by the controller, the signal may be treated as detected. Settings and operations specific to All axes forced stop signal (ForcedStop_Signal) are shown below. ■I/O number (StartIO) Ignores the input value. ■Target (Target) If Target is not specified, signal disabled is determined and the signal non-detection status is always set. The data type that can be specified is only [VAR], [DEV], or [CONST]. * If an unusable data type is specified, Out of Parameter Range (System) (error code: 1A62H) occurs. When [VAR] or [DEV] is specified, if "Asynchronous" is set for the network synchronous communication setting of the device station referred to, Out of Parameter Range (System) (error code: 1A62H) occurs. ■Signal detection method (Detection) Allows detection at the following specification level only. If edge detection specification is specified, Out of Parameter Range (System) (error code: 1A62H) is output. • 0: Detection at TRUE (HighLevel) • 1: Detection at FALSE (LowLevel) ■Compensation time (CompensationTime) Ignores the input value. ■Filter time (FilterTime) The setting range is "0.0 to 5.0 [s]". * If a value outside the range is specified, Out of Filter Time Setting Range of System Signal Warning (event code: 00F0FH) occurs, and the axis operates with 0.0 of the filter time.
StopMode_All	Stop selection at all axes stop by factors	Ready ON	INT (MC_DECEL_STOP_MODE) ☞ Page 1477 MC_DECEL_STOP_MODE	LIST_WRITE_BACK	Whether to stop immediately or decelerate to stop when an all axes stop cause occurs is set. • 1: Immediate stop (ImmediateStop) • 2: Keep current acceleration/deceleration (KeepCurrentAcc) • 3: Alternative acceleration/deceleration (AlternativeAcc) Immediate stop will be executed when executing an FB without deceleration specification.
StopMode_AllDeceleration	Deceleration at all axes stop	Ready ON	LREAL	LIST_WRITE_BACK	The deceleration at deceleration stop when an all axes stop cause occurred is set Immediate stop will be executed when 0.0000 is set. The setting range differs depending on the acceleration/deceleration method. • Acceleration/deceleration specification method: 0.0000, 0.0001 to 2147483647.0 • Acceleration/deceleration time-fixed method: 0.0000, 0.000001 to 8400.0

MotionSystem.Md. (Motion system monitor data)

Variable name	Name	Import	Data type	Attribute	Description
Error	Motion system error detection	Immediately	BOOL	LIST_READ_ONLY	The error status is displayed. • FALSE: No error • TRUE: Error present
ErrorID	Latest motion system error code	Immediately	WORD(HEX)	LIST_READ_ONLY	The latest error code is displayed.
ForcedStop_Released	Forced stop cancelling	Operation cycle	BOOL	LIST_READ_ONLY	The forced stop cancel status is displayed. • FALSE: All axes forced stop • TRUE: All axes forced stop release
ForcedStop_Signal	All axes forced stop signal	Ready ON	SIGNAL_SELECT ☞ Page 1447 SIGNAL_SELECT (Signal select)	LIST_READ_ONLY	The input status of the All axes forced stop signal is displayed. Outputs specific to All axes forced stop signal (ForcedStop_Signal) are shown below. ■I/O number (StartIO) "0" is always displayed. ■Target (Target) The fetch result of the parameter is displayed. ■Signal detection method (Detection) The fetch result of the parameter is displayed. ■Compensation time (CompensationTime) "0.0" is always displayed. ■Filter time (FilterTime) The fetch result of the parameter is displayed.
MotionService	Motion service processing monitor	Service	CYCLE_MONI ☞ Page 1448 CYCLE_MONI	LIST_READ_ONLY	The monitored values of motion service processing is stored.
OperationCycle[1..3]	Operation cycle monitor	Immediately	CYCLE_MONI ☞ Page 1448 CYCLE_MONI	LIST_READ_ONLY	The monitor values of the first operation cycle, second operation cycle, and third operation cycle are stored.
Ready	Ready	Service	BOOL	LIST_READ_ONLY	The on/off status of Built-in motion ready (X20) is displayed. • FALSE: Off (Not READY) • TRUE: On (READY)
StopMode_All	Stop selection at all axes stop by factors	Ready ON	INT (MC_DECEL_STOP_MODE) ☞ Page 1477 MC_DECEL_STOP_MODE	LIST_READ_ONLY	The input status of stop selection at all axes stop by factors is displayed. • 1: Immediate stop (ImmediateStop) • 2: Keep current acceleration/deceleration (KeepCurrentAcc) • 3: Alternative acceleration/deceleration (AlternativeAcc)
StopMode_AllDeceleration	Deceleration at all axes stop	Ready ON	LREAL	LIST_READ_ONLY	The input status of deceleration at all axes stop is displayed.
Sync	Synchronization flag	Service	BOOL	LIST_READ_ONLY	The on/off status of Synchronization flag (X21) is displayed. • FALSE: Off (Module access disabled) • TRUE: On (Module access enabled)
Warning	Motion system warning detection	Immediately	BOOL	LIST_READ_ONLY	The warning status is displayed. • FALSE: No warning • TRUE: Warning present
WarningID	Latest motion system warning code	Immediately	WORD(HEX)	LIST_READ_ONLY	The latest event code is stored.

MotionSystem.Cd. (Motion system control data)

Variable name	Name	Import	Data type	Attribute	Description
ErrorReset	System error reset	Service	BOOL	LIST_READ_ONLY	A reset of all errors and warnings is requested. • FALSE: Not executed • TRUE: Executes the system error reset
SequenceReady	PLC ready	Service	BOOL	LIST_READ_ONLY	The CPU function part is checked for normal operation, and the signal status is displayed to inform the motion function part. • FALSE: PLC ready OFF • TRUE: PLC ready ON ■When "PLC Ready Interlock Selection (MotionSystem.PrConst.SequenceReadyInterlock)" is TRUE When the following conditions are all satisfied, this variable becomes TRUE. • Synchronization flag (X21) turns on. • The status of the CPU function part changes from STOP to PAUSE/RUN. ■When "PLC Ready Interlock Selection (MotionSystem.PrConst.SequenceReadyInterlock)" is FALSE When the following conditions are all satisfied, this variable becomes TRUE. • Synchronization flag (X21) turns on. • The status of the CPU function part changes from STOP to PAUSE/RUN. • PLC READY (Y20) turns on.

Input axis variable

The data type of input axis variables to be used for advanced synchronous control is expressed as the ADV_INPUT type. Each input axis data type has the following members.

Member name	Data type	Description	Reference
LabelID	WORD (UINT)	The input axis label ID is set.	☞ Page 1434 AdvInputName. (Input axis setting)
Axis	AXIS_REF	The axis to be the input axis for advanced synchronous control is set. An identical axis cannot be specified for multiple input axis information.	
PrConst	ADV_INPUT_PRM_CONST	The parameter data (constant) of the axis to be the input axis is stored. The setting value at the input axis variable initialization is opened. Reload to the control is not executed after the input axis variable initialization.	☞ Page 1434 AdvInputName.PrConst. (Input axis parameter constant)
Pr	ADV_INPUT_PRM	The parameter data of the axis to be the input axis is stored. The default value at the input axis variable initialization is opened. Reload to the control is not executed after the input axis variable initialization. The fetch timing to the control changes depending on the parameter.	☞ Page 1435 AdvInputName.Pr. (Input axis parameter)
Md	ADV_INPUT_MONI	The monitor data of the axis to be the input axis is stored. Refresh in the fixed cycle is executed for each monitor data.	☞ Page 1435 AdvInputName.Md. (Input axis monitor data)

AdvInputName. (Input axis setting)

Variable name	Name	Import	Data type	Attribute	Description
LabelID	Axis label ID	At system start	WORD (UINT)	LIST_WRITE_BACK	The input axis label ID is set. • 1 to 256
Axis	Axis information	At system start	AXIS_REF ☞ Page 1398 AxisName.AxisRef. (Axis information)	LIST_WRITE_BACK	Axis information (AxisName.AxisRef) of an axis to be the input axis for advanced synchronous control is set. The maximum number of input axes that can be generated is 256. An identical axis cannot be set for the axis information on different input axes. Do not set Axis information (AxisName.AxisRef) to an axis defined with a local label. Doing so prevents the input axis from being generated. The axis types that can be set as output axes are real drive axes, virtual drive axes, and virtual linked axes. If an unavailable axis is set, Advanced Synchronous Control Input Axis Incorrect (error code: 1A0EH) occurs.

AdvInputName.PrConst. (Input axis parameter constant)

Variable name	Name	Import	Data type	Attribute	Description
SourceValue	Data source selection	At system start	INT (MC_SOURCE) ☞ Page 1474 MC_SOURCE	LIST_WRITE_BACK	The data source of the input axis to be referenced in advanced synchronous control is set. • 1: Set value (mcSetValue) • 2: Actual value (mcActualValue) • 101: Latest set value (mcLatestSetValue) • 102: Latest actual value (mcLatestActualValue)
SmoothingTimeConstant	Smoothing time constant	At system start	WORD (UINT)	LIST_WRITE_BACK	The averaging time for smoothing the input movement amount from the input axis is set. • 0 to 5000 [ms]
PhaseCompensationTimeConstant	Phase compensation time constant	At system start	WORD (UINT)	LIST_WRITE_BACK	The time constant at which the phase compensation amount during phase compensation is reflected with a primary delay is set. 63% of the phase compensation amount is reflected with the set time constant. • 0 to 65535 [ms]
DirectionRestriction	Moving direction restriction	At system start	INT (MC_INPUT_DIRECTION) ☞ Page 1480 MC_INPUT_DIRECTION	LIST_WRITE_BACK	This variable is set when the input movement amount from the input axis needs to be restricted to one direction. • 0: No moving direction restriction (NoDirectionRestriction) • 1: Allowed only when the current position is in the increasing direction (mcPositiveDirection) • 2: Allowed only when the current position is in the decreasing direction (mcNegativeDirection)

AdvInputName.Pr. (Input axis parameter)

Variable name	Name	Import	Data type	Attribute	Description
PhaseCompensationAdvanceTime	Phase compensation advancement time	Operation cycle	DINT	LIST_WRITE_BACK	This variable is set when the input axis phrase (input response) needs to be advanced or delayed. • -100000000 to 100000000 [μs]

AdvInputName.Md. (Input axis monitor data)

Variable name	Name	Import	Data type	Attribute	Description
ActiveStatus	Enable state	Immediately	INT	LIST_READ_ONLY	The input axis status is stored. • -1: Invalid • 0: Not connected • 1: Input axis invalid
CumulativePosition	Cumulative current position	Operation cycle	LREAL	LIST_READ_ONLY	The position unit of the input axis whose cumulative current position is set to Axis information (AdvInputName.Axis) on the input axis is set. The current position of the input axis is a value after smoothing processing, phase compensation processing, and moving direction restriction processing.
SetVelocity	Set velocity	Operation cycle	LREAL	LIST_READ_ONLY	The speed unit of the input axis whose speed is set to Axis information (AdvInputName.Axis) on the input axis is stored. The speed of the input axis is a value after smoothing processing, phase compensation processing, and moving direction restriction processing.
PhaseCompensationAmount	Phase compensation amount	Operation cycle	LREAL	LIST_READ_ONLY	The position command unit of the input axis whose phase compensation amount is set to Axis information (AdvInputName.Axis) on the input axis is stored. The phase compensation amount of the input axis is a value after smoothing processing and phase compensation processing.
DirectionRestrictionAmount	Moving direction restriction amount	Operation cycle	LREAL	LIST_READ_ONLY	The position unit of the axis whose cumulative value of the input movement amount in the direction opposite to the permitted direction is set to the input axis when the moving direction of the input axis is restricted is stored.

Output axis variable

The data type of output axis variables to be used for advanced synchronous control is expressed as the ADV_OUTPUT type. Each output axis data type has the following members.

Member name	Data type	Description	Reference
LabelID	WORD (UINT)	The output axis label ID is set.	☞ Page 1436 AdvOutputName. (Output axis setting)
AxisRef	AXIS_REF	The axis to be the output axis for advanced synchronous control is set. An identical axis cannot be specified for multiple output axis information.	
PrConst	ADV_OUTPUT_PRM_CONST	The parameter data (constant) of the axis to be the output axis is stored. The setting value at the output axis variable initialization is opened. Reload to the control is not executed after the output axis variable initialization.	☞ Page 1437 AdvOutputName.PrConst. (Output axis parameter constant)
Pr	ADV_OUTPUT_PRM	The parameter data of the axis to be the output axis is stored. The default value at the output axis variable initialization is opened. Reload to the control is executed after the output axis variable initialization. The fetch timing to the control changes depending on the parameter.	☞ Page 1438 AdvOutputName.Pr. (Output axis parameter)
Md	ADV_OUTPUT_MONI	The monitor data of the axis to be the output axis is stored. Refresh in the fixed cycle is executed for each monitor data.	☞ Page 1440 AdvOutputName.Md. (Output axis monitor data)
Cd	ADV_OUTPUT_CMD	The output axis control command data is stored. The latest value is acquired for every control operation cycle and used for the control.	☞ Page 1440 AdvOutputName.Cd. (Output axis control data)

AdvOutputName. (Output axis setting)

Variable name	Name	Import	Data type	Attribute	Description
LabelID	Axis label ID	At system start	WORD (UINT)	LIST_WRITE_BACK	The output axis label ID is set. • 1 to 256
Axis	Axis information	At system start	AXIS_REF ☞ Page 1398 AxisName.AxisRef. (Axis information)	LIST_WRITE_BACK	Axis information (AxisName.AxisRef) of an axis to be the output axis for advanced synchronous control is set. The maximum number of output axes that can be generated is 256. An identical axis cannot be set for the axis information on different output axes. If it is set, Advanced Synchronous Control Output Axis Incorrect (error code: 1A0FH) will occur. Do not set Axis information (AxisName.AxisRef) to an axis defined with a local label. Doing so prevents the output axis from being generated.

AdvOutputName.PrConst. (Output axis parameter constant)

Variable name	Name	Import	Data type	Attribute	Description
SmoothingTimeConstant	Output axis smoothing time constant	At system start	WORD (UINT)	LIST_WRITE_BACK	The averaging time [ms] for smoothing the output movement amount from the output axis is set. • 0 to 5000 [ms]
MasterClutchSmoothingTimeConstant	Master axis clutch smoothing time constant	At system start	WORD (UINT)	LIST_WRITE_BACK	The time constant is set when Master axis clutch smoothing method (AdvOutputName.Pr.Clutch.MasterSmoothingMethod) is set to "1: Time constant method (exponent) (TimeConstantExponent)" or "2: Time constant method (linear) (TimeConstantLinear)". The time constant to be set is common to both clutch on and off. • 0 to 5000 [ms]
AuxClutchSmoothingTimeConstant	Auxiliary axis clutch smoothing time constant	At system start	WORD (UINT)	LIST_WRITE_BACK	The time constant is set when Auxiliary axis clutch smoothing method (AdvOutputName.Pr.Clutch.AuxSmoothingMethod) is set to "1: Time constant method (exponent) (TimeConstantExponent)" or "2: Time constant method (linear) (TimeConstantLinear)". The time constant to be set is common to both clutch on and off. • 0 to 5000 [ms]
MasterSpeedChangeGearSmoothingTimeConstant	Master axis speed change gear smoothing time constant	At system start	WORD (UINT)	LIST_WRITE_BACK	The averaging time to smooth the speed change during master axis speed change processing is set. Note that the smoothing process delays the input value transmission for the setting time. When the setting value is "0", the speed changes directly. • 0 to 5000 [ms]
AuxSpeedChangeGearSmoothingTimeConstant	Auxiliary axis speed change gear smoothing time constant	At system start	WORD (UINT)	LIST_WRITE_BACK	The averaging time to smooth the speed change during auxiliary axis speed change processing is set. Note that the smoothing process delays the input value transmission for the setting time. When the setting value is "0", the speed changes directly. • 0 to 5000 [ms]
OutSpeedChangeGearSmoothingTimeConstant	Output axis speed change gear smoothing time constant	At system start	WORD (UINT)	LIST_WRITE_BACK	The averaging time to smooth the speed change during output axis speed change processing is set. Note that the smoothing process delays the input value transmission for the setting time. When the setting value is "0", the speed changes directly. • 0 to 5000 [ms]

AdvOutputName.Pr. (Output axis parameter)

Variable name	Name	Import	Data type	Attribute	Description
SubAxis	Sub-input axis target	At start	AXIS_REF ☞ Page 1398 AxisName.AxisRef. (Axis information)	LIST_WRITE_BACK	The target axis of the sub-input axis is set. This variable is invalid when set to "0". If an axis that has not been assigned to the input settings is set, Advanced synchronization control axis unset (error code: 1B22H) occurs.
AuxAxis	Auxiliary axis target	At start	AXIS_REF ☞ Page 1398 AxisName.AxisRef. (Axis information)	LIST_WRITE_BACK	The target axis of the auxiliary axis is set. This variable is invalid when set to "0". If an axis that has not been assigned to the input settings is set, Advanced synchronization control axis unset (error code: 1B22H) occurs.
Gear	Gear parameter	At start	ADV_GEAR_PARAM ☞ Page 1462 ADV_GEAR_PARAM (Gear parameter)	LIST_WRITE_BACK	The gear function parameters are set.
MasterOnClutchSignal	Master axis ON clutch signal setting	At start	SIGNAL_SELECT ☞ Page 1447 SIGNAL_SELECT (Signal select)	LIST_WRITE_BACK	The external signal of the clutch is set when Master axis ON clutch control setting (AdvOutputName.Pr.Clutch.MasterOnControl) is "15: I/O data specification (ClutchSignal)". Settings and operations specific to Master axis ON clutch signal setting (MasterOnClutchSignal) are shown below. ■Target (Target) Data type can only be set to [OBJ], [VAR], [DEV], and [LINK]. ■Signal detection method (Detection) Allows detection at the following setting level only. • 2: Detection at FALSE → TRUE (rising edge) (RisingEdge) • 3: Detection at TRUE → FALSE (falling edge) (FallingEdge) • 4: Detection at rising edge/falling edge (BothEdges) ■Compensation time (CompensationTime) The setting range is "-5.0 to 5.0 [s]". ■Filter time (FilterTime) The setting range is "0.0".
MasterOffClutchSignal	Master axis OFF clutch signal setting	At start	SIGNAL_SELECT ☞ Page 1447 SIGNAL_SELECT (Signal select)	LIST_WRITE_BACK	The external signal of the clutch is set when Master axis OFF clutch control setting (AdvOutputName.Pr.Clutch.MasterOffControl) is "15: I/O data specification (ClutchSignal)". Settings and operations specific to Master axis OFF clutch signal setting (MasterOffClutchSignal) are shown below. ■Target (Target) Data type can only be set to [OBJ], [VAR], [DEV], and [LINK]. ■Signal detection method (Detection) Allows detection at the following setting level only. • 2: Detection at FALSE → TRUE (rising edge) (RisingEdge) • 3: Detection at TRUE → FALSE (falling edge) (FallingEdge) • 4: Detection at rising edge/falling edge (BothEdges) ■Compensation time (CompensationTime) The setting range is "-5.0 to 5.0 [s]". ■Filter time (FilterTime) The setting range is "0.0".
AuxOnClutchSignal	Auxiliary axis ON clutch signal setting	At start	SIGNAL_SELECT ☞ Page 1447 SIGNAL_SELECT (Signal select)	LIST_WRITE_BACK	The external signal of the clutch is set when Auxiliary axis ON clutch control setting (AdvOutputName.Pr.Clutch.AuxOnControl) is "15: I/O data specification (ClutchSignal)". Settings and operations specific to Auxiliary axis ON clutch signal setting (AuxOnClutchSignal) are shown below. ■Target (Target) Data type can only be set to [OBJ], [VAR], [DEV], and [LINK]. ■Signal detection method (Detection) Allows detection at the following setting level only. • 2: Detection at FALSE → TRUE (rising edge) (RisingEdge) • 3: Detection at TRUE → FALSE (falling edge) (FallingEdge) • 4: Detection at rising edge/falling edge (BothEdges) ■Compensation time (CompensationTime) The setting range is "-5.0 to 5.0 [s]". ■Filter time (FilterTime) The setting range is "0.0".

Variable name	Name	Import	Data type	Attribute	Description
AuxOffClutchSignal	Auxiliary axis OFF clutch signal setting	At start	SIGNAL_SELECT ☞ Page 1447 SIGNAL_SELECT (Signal select)	LIST_WRITE_BACK	The external signal of the clutch is set when Auxiliary axis OFF clutch control setting (AdvOutputName.Pr.Clutch.AuxOffControl) is "15: I/O data specification (ClutchSignal)". Settings and operations specific to Auxiliary axis OFF clutch signal setting (AuxOffClutchSignal) are shown below. ■Target (Target) Data type can only be set to [OBJ], [VAR], [DEV], and [LINK]. ■Signal detection method (Detection) Allows detection at the following setting level only. • 2: Detection at FALSE → TRUE (rising edge) (RisingEdge) • 3: Detection at TRUE → FALSE (falling edge) (FallingEdge) • 4: Detection at rising edge/falling edge (BothEdges) ■Compensation time (CompensationTime) The setting range is "-5.0 to 5.0 [s]". ■Filter time (FilterTime) The setting range is "0.0".
Clutch	Clutch parameter	—	ADV_CLUTCH_PARAM ☞ Page 1463 ADV_CLUTCH_PARAM (Clutch parameter)	LIST_WRITE_BACK	The clutch function parameters are set.
SpeedChangeGear	Speed change gear parameter	—	ADV_SPEEDCHANGEGEAR_PARAM ☞ Page 1468 ADV_SPEEDCHANGEGEAR_PARAM (Speed change gear parameter)	LIST_WRITE_BACK	The speed change gear function parameters are set.
Cam	Cam parameter	—	ADV_CAM_PARAM ☞ Page 1469 ADV_CAM_PARAM (Cam parameter)	LIST_WRITE_BACK	The cam function parameters are set.
Restore	Synchronous control initial position parameter	—	ADV_RETORE_PARAM ☞ Page 1472 ADV_RESTORE_PARAM (Synchronous control initial position parameter)	LIST_WRITE_BACK	The synchronous control initial position parameters are set.

AdvOutputName.Md. (Output axis monitor data)

Variable name	Name	Import	Data type	Attribute	Description
SyncStatus	Synchronous status	Immediately	INT	LIST_READ_ONLY	The output axis synchronous status is stored. • -1: Invalid • 0: Unsynchronized • 1: Synchronized
MasterAxisNo	Master axis	At start	WORD (UINT)	LIST_READ_ONLY	The axis number of the master axis that is set to Axis information (AdvInputName.Axis) on the input axis that is in synchronization with the output axis is stored. If no connection is established, "0" is stored.
SubAxisNo	Sub-input axis	At start	WORD (UINT)	LIST_READ_ONLY	The axis number of the sub-input axis that is set to Axis information (AdvInputName.Axis) on the input axis that is in synchronization with the output axis is stored. If no connection is established, "0" is stored.
AuxAxisNo	Auxiliary axis	At start	WORD (UINT)	LIST_READ_ONLY	The axis number of the auxiliary axis that is set to Axis information (AdvInputName.Axis) on the input axis that is in synchronization with the output axis is stored. If no connection is established, "0" is stored.
Clutch	Clutch monitor	—	ADV_CLUTCH_MONI ☞ Page 1466 ADV_CLUTCH_MONI (Clutch monitor)	LIST_READ_ONLY	The monitor data of the clutch function is stored.
Cam	Cam monitor	—	ADV_CAM_MONI ☞ Page 1470 ADV_CAM_MONI (Cam monitor)	LIST_READ_ONLY	The monitor data of the cam function is stored.

AdvOutputName.Cd. (Output axis control data)

Variable name	Name	Import	Data type	Attribute	Description
Clutch	Clutch control data	—	ADV_CLUTCH_CMD ☞ Page 1466 ADV_CLUTCH_CMD (Clutch control data)	LIST_READ_ONLY	The clutch function control data is set.
SpeedChangeGear	Speed change gear control data	—	ADV_SPEEDCHANGEGEAR_CMD ☞ Page 1468 ADV_SPEEDCHANGEGEAR_CMD (Speed change gear control data)	LIST_WRITE_BACK	The speed change gear function control data is set.
Cam	Cam control data	—	ADV_CAM_CMD ☞ Page 1471 ADV_CAM_CMD (Cam control data)	LIST_READ_ONLY	The control data of the cam function is set.

System variable (Advanced synchronous control)

The data type of system variables is expressed as SYSTEM type.

The system data type has the following members.

Member name	Data type	Description	Reference
PrConst	SYS_CONST	Stores the parameter data (constant) for the system. Opens the setting value at system variable initialization. Does not execute reimporting to the control after system variable initialization.	☞ Page 1441 AddonSystem.PrConst. (Add-on system parameter constant)
Md	SYS_MONI	Stores the monitor data for the system. Executes refresh in the fixed cycle for each monitor data.	☞ Page 1441 AddonSystem.Md. (Add-on system monitor data)

AddonSystem.PrConst. (Add-on system parameter constant)

Variable name	Name	Import	Data type	Attribute	Description
MT_MotionControl_AdvancedSync	Add-on MotionControl_AdvancedSync parameter	At system start	ADDON_PARAM ☞ Page 1448 ADDON_PARAM	LIST_WRITE_BACK	The maximum usage of the memory used by the add-on MotionControl_AdvancedSync parameter is set.

AddonSystem.Md. (Add-on system monitor data)

Variable name	Name	Import	Data type	Attribute	Description
MT_MotionControl_AdvancedSync	Add-on MotionControl_AdvancedSync monitor	Service	ADDON_MONI ☞ Page 1448 ADDON_MONI	LIST_READ_ONLY	The free space of the memory used by the add-on MotionControl_AdvancedSync parameter is stored.

Other structure variable

MC_CAM_REF

Variable name	Name	Data type	Description
ProfileData	Profile	PROFILE_DATA ☞ Page 1447 PROFILE_DATA	The profile of the cam data is set.
CamID	Cam ID	MC_CAM_ID ☞ Page 1442 MC_CAM_ID	The profile ID of the cam data is set.

MC_POSITIONING_DATA_REF

Variable name	Name	Data type	Description
ProfileData	Profile	PROFILE_DATA ☞ Page 1447 PROFILE_DATA	The profile of the positioning data is set.
PositioningData	Positioning data	MC_POSITIONING_DATA_ID ☞ Page 1442 MC_POSITIONING_DATA_ID	The ID number of the positioning data is set.

MC_CAM_ID

Variable name	Name	Data type	Description
ProfileID	Profile ID	PROFILE_ID ☞ Page 1447 PROFILE_ID	The profile ID of the cam data is set.

MC_POSITIONING_DATA_ID

Variable name	Name	Data type	Description
ProfileData	Profile ID	PROFILE_ID ☞ Page 1447 PROFILE_ID	The operation profile ID of the positioning data is set.

MC_TRIGGER_REF

Variable name	Name	Data type	Description
Signal	Trigger signal	SIGNAL_SELECT ☞ Page 1447 SIGNAL_SELECT (Signal select)	The trigger signal is set.

MC_INPUT_REF

Variable name	Name	Data type	Description
Signal	Input signal	SIGNAL_SELECT ☞ Page 1447 SIGNAL_SELECT (Signal select)	The input signal is set.

MC_OUTPUT_REF

Variable name	Name	Data type	Description
Signal	Output signal	SIGNAL_SELECT ☞ Page 1447 SIGNAL_SELECT (Signal select)	The output signal is set.

MC_CAMSWITCH_REF

Variable name	Name	Data type	Description
ProfileData	Profile	PROFILE_DATA ☞ Page 1447 PROFILE_DATA	The profile of the digital cam switch is set.
CamSwitchData	Cam switch data	MC_CAMSWITCH_DATA_REF ☞ Page 1443 MC_CAMSWITCH_DATA_REF	The ID number of the digital cam switch is set.

MC_TRACK_REF

Variable name	Name	Data type	Description
OnCompensation	ON time compensation	LREAL	The timing when the output signal turns on is compensated.
OffCompensation	OFF time compensation	LREAL	The timing when the output signal turns off is compensated.
Hysteresis	Switching disabled distance	LREAL	The distance within which switching is prohibited by the data source is set. When the data source increases or decreases near the switching point, the chattering of output signals is prevented.

MC_CAMSWITCH_DATA_REF

Variable name	Name	Data type	Description
ProfileData	Profile ID	PROFILE_ID ☞ Page 1447 PROFILE_ID	The operation profile ID of the digital cam switch is specified.

MC_CAMSWITCH_SOURCE_REF

Variable name	Name	Data type	Description
Source	Data source	TARGET_REF ☞ Page 1444 TARGET_REF (Input signal)	A data source is specified.
RingCount_LowerValue	Data source ring counter lower limit value	LREAL	The data source ring counter lower limit value is set.
RingCount_UpperValue	Data source ring counter upper limit value	LREAL	The data source ring counter upper limit value is set.
SpeedAverageNum	Number of speed averaging times	WORD (UINT)	To calculate the on/off timing of the switch, set until which previous operation cycles to include for average speed calculation, counting from the present.

TARGET_REF (Input signal)

Variable name	Name	Data type	Description
StartIO	IO No.	WORD(HEX)	The IO No. is set.
Target	Target	WSTRING(63)	The signal used in control is set with the following string formats. The setting is not case-sensitive. In addition, spaces are ignored. ■[Classification](Type)Data name.Bit position@Target modification • [Classification]*1 • (Type)*2 • Data name*1 • .Bit position*3 • @Target modification*4 ■[Classification](Data type)WSTRING type label^ • [Classification]*1 • (Type)*2 • WSTRING type label^*8

*1 The data classification is set in "[Classification]", and the target data is set in "Data name". The method for setting the data name differs depending on the set classification.

Classification	Target	Data name		Setting example
		Format	Description	
[OBJ]	CANopen object of the device (which has been mapped to cyclic data)	[OBJ]0x□□□□**##	The index, subindex, and size of the object are set. • □□□□: Index • **: Subindex • ##: Size (in bits) *: Match the (type) setting with the object size. *: Some of the objects can be used as an external signal high-accuracy input. *: Objects with a size of "1 byte" cannot be used.	[OBJ]0x607A0020 *: For an MR-J5(W)-G Target position (Obj.607AH: 00H) 4 byte object (20H)
[VAR]	Label of the motion system	[VAR]Label name	A label in the motion system is set. *: When setting a local label, set the POU name in "@Target modification".	[VAR]ADunit10.OutputEnable
[AXIS]	Axis data	[AXIS]MC_SOURCE	An MC_SOURCE enumeration type enumerator is set. Only the data type "LREAL" can be set. • 1: Set value (mcSetValue) • 2: Actual value (mcActualValue) • 101: Latest set value (mcLatestSetValue) • 102: Latest actual value (mcLatestActualValue) *: This data cannot be written.	[AXIS]mcSetValue
[DEV]	Device in the motion system (Including link device of the module itself)	[DEV]Data name	A device in the module itself is set. For details on settable devices, refer to the following. ☞ Page 1446 List of supported devices for [DEV] type	[DEV]X40 [DEV]U0\G11500000.1
[CONST]	Constant	[CONST]Constant [CONST]0xConstant	A floating decimal (E format is also available), decimal integer, or hexadecimal integer is used for the setting. *: This data cannot be written.	[CONST]1000 [CONST]0x100

*2 This is written when explicitly setting the data type. The following data types can be set.
When "(BOOL)" is specified for data set in Classification having a data type other than BOOL type, it will be handled as though 0 was specified for the specification target bit number.

Type
(BOOL)
(INT)
(DINT)
(WORD)
(DWORD)
(REAL)
(LREAL)

*3 When the bit position is added to data of the [Classification] shown below, the data is handled as though it is BOOL type. When (Type) and the bit position are both set, the setting for (Type) is ignored and the data becomes BOOL type data.

Classification	Bit position setting range
[OBJ]	0 to F
[VAR] (only WORD type, DWORD type, INT type, DINT type)	
[DEV]	

*4 The auxiliary information used for specifying the data is set. The details differ depending on the set [Classification]. Certain functions that use signals require the target modification to be set. When specifying the target modification for data types and functions that do not require it, the specification will be ignored.

Classification	Target modification		Setting example
	Format	Description	
[OBJ]	@Station address	The station address (IP address to identify the device) is set. When multiple logic axes are included in one station such as with a multi-axis drive unit, set multi-drop numbers to identify certain logic axes. @192.168.3.10#2 Multidrop number IP address *: The multi-drop No. is set as "# + No. (decimal)". • #0: A-axis • #1: B-axis • #2: C-axis *: The multi-drop No. can be omitted. When it is omitted, it is regarded as "#0".	@192.168.3.10 @192.168.3.10#2
[VAR]	@POU name	The POU name is set when setting local labels. When @POU is not written, the data is handled as a global label regardless of the setting which uses the same label name for a global label and a local label.	
[AXIS]	@Position@CumulativePos	The target data (position) is set. @Position and @CumulativePos are set with the MC_SOURCE type enumerator set in "Data name".*5	@Position @CumulativePos
[DEV]	—	The target modification cannot be set. It is ignored if set.	
[CONST]	@+adding value @-subtracting value	Used to change the read value each time the value is referred. Setting the target modification allows the value to be changed each cycle. When the data type is REAL or LREAL, this setting is invalid.	[CONST](INT)0@-10*6 [CONST](BOOL)0x00.3@+1*7

*5 The following enumerators can be set.

○: Can be set, ×: Cannot be set

MC_SOURCE type enumerator	Description	Modification	
		@Position	@CumulativePos
1: Set value (mcSetValue)	Set value of previous operation cycle	○	○
2: Actual value (mcActualValue)	Actual value of previous operation cycle	○	×
101: Latest set value (mcLatestSetValue)	Set value of latest operation cycle	○	○
102: Latest actual value (mcLatestActualValue)	Actual value of latest operation cycle	○	×

*6 The referred value will be as "0, -10, -20, -30, ...".

*7 The referred value switches between TRUE/FALSE every eight (2^3) cycles. (Adds 1 byte value "+1" every cycle, then retrieves the status of bit 3.)
Values from 0 to 7 can be used for bit specification for [CONST].

*8 When setting to a format with more than 63 characters, use WSTRING type labels and use the following method to perform the setting.
(1) Declare the WSTRING type global label in the motion system, and store the following portion: Data name.Bit position@Target modification.
(2) Set the label name from (1) as the WSTRING type label of TARGET_REF. (Add "^" to the end of the name to indicate reference.)

■List of supported devices for [DEV] type

A list of supported devices for [DEV] type is shown below.

R: Only read is available, RW: Read and write are available, —: Not supported

Device type	Device name	Symbol	Supported
User device	Input	X	RW*1*2
	Direct access input	DX	—
	Output	Y	RW*1*2
	Direct access output	DY	—
	Internal relay	M	RW*2
	Latch relay	L	RW*2
	Link relay	B	RW*2
	Link special relay	SB	RW*2
	Annunciator	F	R
	Edge relay	V	—
	Timer	T	R*3
	Retentive timer	ST	R*3
	Long timer	LT	R*3
	Long retentive timer	LST	R*3
	Counter	C	R*3
	Long counter	LC	R*3
	Data register	D	RW*2
	Link register	W	RW*2
	Link special register	SW	RW*2
System device	Special relay	SM	RW
	Special register	SD	RW
	Function input	FX	—
	Function output	FY	—
	Function register	FD	—
File register	File register	R	—
		ZR	—
Index register	Index register	Z	—
	(32-bit index modification)	(ZZ)	—
	Long index register	LZ	—
Pointer	Pointer	P	—
	Interrupt pointer	I	—
Module access device	Module access device	Un\G	RW*4
CPU buffer memory access device	CPU buffer memory access device	U3E0\G	RW*5
Refresh data register	Refresh data register	RD	—
Nesting	Nesting	N	—
Others	Network No. specification device	J	—
	Intelligent module No. specification device	U	—

*1 Only unassigned devices can be used.

*2 When the device is set as a refresh target in refresh settings, whether the device station is connected is monitored. For the operation when the device station is disconnected, refer to each of the following functions.
MELSEC MX Controller MX-R Model User's Manual
When the device is set as a refresh target in "Refresh Settings", the signals available are only those of devices whose "Network Synchronous Communication" is set to "Synchronous".

*3 Timer, retentive timer, long timer, long retentive timer, counter, and long counter have a contact (S), coil (C), and current value (N). When specifying them as the data name, specify as follows: Contact: "Device symbol S", Coil: "Device symbol C", Current value: "Device symbol N"
To specify the contact of retentive timer ST10 to the data name: [DEV]STS10
To specify the current value of long counter LC10 to the data name: [DEV]LCN10
When using timer, retentive timer, long timer, long retentive timer, counter, or long counter, specify the type in accordance with the device size.

*4 Supported only when the buffer memory of the motion function part or CC-Link IE TSN function part of the controller is specified.

*5 Supported only when the CPU buffer memory of the controller is specified.

SIGNAL_SELECT (Signal select)

Variable name	Name	Data type	Description
Source	Signal	TARGET_REF ☞ Page 1444 TARGET_REF (Input signal)	BOOL type signals used in control are set. For Source (Signal), only the following data classifications can be set. • [OBJ] (which has been mapped to cyclic data) • [VAR] • [DEV] • [CONST]
Detection	Signal detection method	INT (MC_SIGNAL_LOGIC) ☞ Page 1478 MC_SIGNAL_LOGIC	The signal logic is set. • 0: Detection at TRUE (HighLevel) • 1: Detection at FALSE (LowLevel) • 2: Detection at FALSE → TRUE (rising edge) (RisingEdge) • 3: Detection at TRUE → FALSE (falling edge) (FallingEdge) • 4: Detection at rising edge/falling edge (BothEdges)
CompensationTime	Compensation time	LREAL	The compensation time of the I/O signal is set in units of seconds. *: The setting range differs for each function.
FilterTime	Filter time	LREAL	The filter time for removing chattering of the I/O signal is set in units of seconds. *: The setting range differs for each function.

PROFILE_DATA

Variable name	Name	Data type	Description
Location	Operation profile data storage location	FILE_LOCATION ☞ Page 1447 FILE_LOCATION	The file name and storage location of the operation profile data is set.
ID	Profile ID	PROFILE_ID ☞ Page 1447 PROFILE_ID	The profile ID No. of the operation profile data is set.

FILE_LOCATION

Variable name	Name	Data type	Description
FileName	File name	WSTRING(63)	The file name is set. (Up to 63 characters including the extension can be set.)*1
Path	Folder specification	WSTRING(63)	The folder path in which the file is stored is set. (Up to 63 characters can be set.)*1

*1 Set the total character amount for File name (FileName) and Folder specification (Path) to within 127 characters (including null).

PROFILE_ID

Variable name	Name	Data type	Description
Number	Profile ID No.	WORD(UINT)	The profile ID No. is set. • 0 to 60000
StartIO	I/O number	WORD	The I/O No. is set.

CYCLE_PARAM

Variable name	Name	Data type	Description
NumOfCycleOverWngDetectTimes	Number of cycle over warning detections	WORD(UINT)	The number of cycles until the cycle over warning of each cycle is detected can be set. Warning is not detected when 0 is set. When a value outside the range is set, operation is performed with the default value. • 0 to 5 times
NumOfCycleOverErrDetectTimes	Number of cycle over error detections	WORD(UINT)	The number of cycles until the cycle over error of each cycle is detected can be set. When a value outside the range is set, operation is performed with the default value. • 1 to 5 times
CycleOverErrorType	Cycle over error selection	INT (MC_ERROR_CLASS) ☞ Page 1478 MC_ERROR_CLASS	The assignment of the over error for each cycle is set. • 2: Minor Error (MinorError) • 3: Moderate Error (ModerateError)

CYCLE_MONI

Variable name	Name	Data type	Description
ProcessingTime	Processing time	DWORD(UDINT)	The processing time of each cycle is stored in units of [ns].
MaximumProcessingTime	Maximum processing time	DWORD(UDINT)	The maximum processing time of each cycle is stored in units of [ns].
Cycle	Setting cycle	DWORD(UDINT)	The cycle setting of each cycle is stored in units of [ns].
CycleOver	Cycle over	BOOL	This variable becomes TRUE when the cycle over is detected for each cycle.

ADDON_PARAM

Variable name	Name	Data type	Description
RamSizeMax	Maximum RAM size	DWORD(UDINT)	The maximum amount [k byte] of system memory (RAM) used in the add-ons is set.
BackupRamSizeMax	Maximum backup RAM size	DWORD(UDINT)	The maximum amount [k byte] of system memory (backup RAM) being used in the add-ons is set.

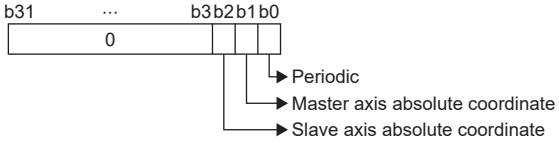
ADDON_MONI

Variable name	Name	Data type	Description
RamUsage	RAM usage	DWORD(UDINT)	The current amount [k byte] of system memory (RAM) being used in the add-ons is stored.
RamMaxUsage	RAM usage maximum	DWORD(UDINT)	The maximum amount [k byte] of system memory (RAM) being used in the add-ons is stored.
BackupRamUsage	Backup RAM usage	DWORD(UDINT)	The current amount [k byte] of system memory (backup RAM) being used in the add-ons is stored.
BackupRamMaxUsage	Backup RAM usage maximum	DWORD(UDINT)	The maximum amount [k byte] of system memory (backup RAM) being used in the add-ons is stored.
Version	Version	STRING(15)	The version information of the add-ons is stored.

INSTANCE_ID

Variable name	Name	Data type	Description
StartIO	IO No.	WORD(HEX)	The IO No. is set.
Number	Instance ID	WORD(UINT)	The instance ID number is set.

PROFILE_CAM_DATA

Variable name	Name (device offset)	Data type	Description
Interpolate	Specifying interpolation method (+0)	INT	The specifying interpolation method for the cam data is set. • 0: Linear interpolation • 1: Specifying for each section • 2: Spline interpolation
Resolution	Resolution (+2)	DWORD(UDINT)	The cam resolution amount for one cycle is set. • 8 to 65535
InputUnitString	Input unit string (+4)	WSTRING(31)	The unit for input data is set as a string. When omitted, the setting is handled as "pulse".
OutputUnitString	Output unit string (+36)	WSTRING(31)	The unit for output data is set as a string. When omitted, the setting is handled as "pulse".
StartPoint	Start point*1 (+68)	LREAL	The start point is set. Set "0.0". *: If a value other than "0.0" is set, Operation Profile Data Incorrect (error code: 1B8FH) occurs.
StartStroke	Initial stroke amount*1 (+72)	LREAL	The stroke amount for the start point is set. (-Stroke amount (Stroke) ≤ Initial stroke amount (StartStroke) ≤ Stroke amount (Stroke)) • -100.0000000 to 100.0000000
StartVelocity	Initial velocity*2 (+76)	LREAL	The velocity for the start point is set. • -2500000000.0 < Initial velocity (StartVelocity) ≤ 2500000000.0
StartAcceleration	Initial acceleration*2 (+80)	LREAL	The acceleration for the start point is set. • -2147483647.0 < Initial acceleration (StartAcceleration) ≤ 2147483647.0
CycleLength	Length per cycle*1 (+84)	LREAL	The input amount that is required for one cycle is set. (0.00000000000001 ≤ Length per cycle (CycleLength) ≤ 10000000000.0) • 0.00000000000001 to 10000000000.0
CycleMin	Minimum value per cycle (+88)	LREAL	The minimum value per cycle is set. Set "0.0". *: If a value other than "0.0" is set, Operation Profile Data Incorrect (error code: 1B8FH) occurs.
CycleMax	Maximum value per cycle (+92)	LREAL	The maximum value per cycle is set. (Minimum value per cycle (CycleMin) < Maximum value per cycle (CycleMax) ≤ 10000000000.0) • -10000000000.0 to 10000000000.0
CycleTime	Time per cycle*2 (+96)	LREAL	The time per cycle is set. (0.001 < Time per cycle (CycleTime) ≤ 100000.0) • 0.001 to 100000.000
Stroke	Stroke amount (+100)	LREAL	The stroke amount is set. (0.00000000000001 ≤ Stroke amount (Stroke) ≤ 10000000000.0) • 0.00000000000001 to 10000000000.0
NumberOfSections	Number of sections*1 (+104)	DWORD(UDINT)	The number of sections for the cam data is set. ■When set to "1: Specifying for each section" • 1 to 360 ■When set to "2: Spline interpolation" • 3 to 360 *: When set to "0: Linear interpolation", this variable is not used. It is ignored if set.
Options	Options (+106)	DWORD(HEX)	Options are set in the specified bit position.  ■Periodic (Bit 0) • 0: Non periodic • 1: Periodic ■Master axis absolute coordinate (Bit 1) • 0: Relative coordinate ■Slave axis absolute position (Bit 2) • 0: Relative coordinate • 1: Absolute coordinate *: Set bits 3 to 31 to "0".

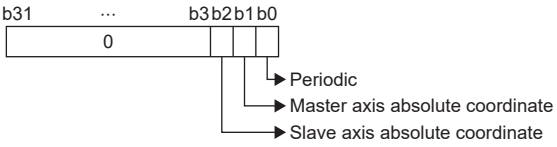
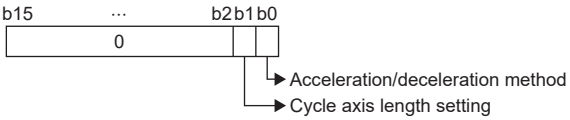
*1 When Specifying interpolation method (Interpolate) is set to "0: Linear interpolation", this variable is not used. It is ignored if set.
*2 Only used when the cam curve is "12: 5th curve (Adjustment) (FifthCurve_SpeedDesignation)". It is ignored for other curves.

PROFILE_CAM_ELEMENT

Variable name	Name (device offset)	Data type	Description
CurveType	Cam curve type (+0)	INT (MC_CAM_CURVE_TYPE) ☞ Page 1476 MC_CAM_CURVE_TYPE	The cam curve is set. • 0: Constant Velocity (ConstantSpeed) • 1: Constant acceleration (ConstantAcceleration) • 2: Distorted trapezoid (DistortedTrapezoid) • 3: Distorted sine (DistortedSine) • 4: Distorted constant speed (DistortedConstantSpeed) • 5: Cycloid (Cycloid) • 6: 5th curve (FifthCurve) • 7: Trapecloid (Trapecloid) • 8: Reverse trapecloid (ReverseTrapecloid) • 9: Double hypotenuse (DoubleHypotenuse) • 10: Reverse double hypotenuse (ReverseDoubleHypotenuse) • 11: Single hypotenuse (SingleHypotenuse) • 12: 5th curve (Adjustment) (FifthCurve_SpeedDesignation)
EndPoint	End point*1 (+4)	LREAL	The position (the current value per cycle) to the length per cycle is set. • 0.0 < End point (EndPoint) ≤ Length per cycle (CycleLength)
Stroke	Stroke (+8)	LREAL	The stroke position is set. • Absolute value of Stroke (Stroke) ≤ Stroke amount (Stroke)
RangeP1	Curve application range (P1) (+12)	LREAL	Set the curve applicable range (start point: P1, end point: P2) for the cam curve. Set this variable within the range of "P1 < P2". Note that when "P1 = P2 = 0", "P1 = 0" and "P2 = 1" are applied. • 0.0 to 1.0
RangeP2	Curve application range (P2) (+16)	LREAL	
RangeL1	Acceleration/deceleration range compensation (Range L1) (+20)	LREAL	The acceleration/deceleration range (L1, L2) for the cam curve is set. The range that can be set differs depending on the cam curve. When "L1 = L2 = 0.0000", the default value for each cam curve is applied. The setting value is ignored for curves that do not use L1 or L2. • 0.0001 < Range (L1, L2) < 1.0000
RangeL2	Acceleration/deceleration range compensation (Range L2) (+24)	LREAL	
EndVelocity	End point velocity*2 (+28)	LREAL	The end point velocity of the cam curve is set. • -2500000000.0 < End point velocity (EndVelocity) < 2500000000.0
EndAcceleration	End point acceleration*2 (+32)	LREAL	The end point acceleration of the cam curve is set. • -2147483647.0 < End point acceleration (EndAcceleration) < 2147483647.0

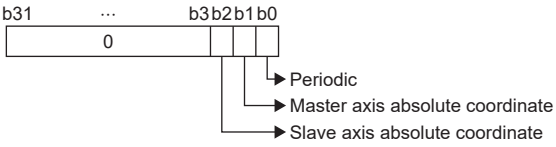
*1 When the end point exceeds the length per cycle before reaching the final section, the section number at the time is determined as the final section number and the end point is overwritten with the length per cycle.
*2 The setting items used to read/write differ depending on Specifying interpolation method of the operation profile data.

PROFILE_ROTARY_CUTTER

Variable name	Name (device offset)	Data type	Description
Resolution	Resolution (+0)	DWORD(UDINT)	The resolution of the generated cam is set. • 8 to 32768
InputUnitString	Input unit string (+2)	WSTRING(31)	The input unit string is set.
Options	Options (+34)	DWORD(HEX)	Options are set in the specified bit position.  ■Periodic (Bit 0) • 0: Non periodic • 1: Periodic ■Master axis absolute coordinate (Bit 1) • 0: Relative coordinate ■Slave axis absolute position (Bit 2) • 0: Relative coordinate • 1: Absolute coordinate *: Set bits 3 to 31 to "0".
AutoGenerationOptions	Auto-generation option (+36)	WORD(HEX)	The auto-generation options are set in the specified bit position.  ■Acceleration/deceleration method (Bit 0) • 0: Trapezoidal acceleration/deceleration • 1: S-curve acceleration/deceleration ■Synchronous axis length setting (Bit 1) • 0: Diameter • 1: Cycle length *: Set bits 2 to 15 to "0".
NumberOfCutter	Number of cutters (+37)	WORD(UINT)	The number of cutters is set. • Setting range: 1 to 256
SyncSectionAccRatio	Synchronous section acceleration ratio (+40)	LREAL	Set this variable to precisely adjust the synchronous speed in the synchronous section. The value becomes "Synchronous section speed = Synchronous speed × (100% + Acceleration ratio)". • -50.0 to 50.0
SheetLength	Sheet length (+44)	LREAL	The sheet length is set. (0.0 < Sheet length (SheetLength) ≤ 10000000000.0) • 0.00000000000001 to 10000000000.0
SheetSyncWidth	Sheet synchronization width (+48)	LREAL	The sheet synchronization width (seal width) is set. When the synchronous speed section for retracting operation is required in front of and behind the sheet synchronization width, add the retracting width. (0.0 < Sheet synchronization width (SheetSyncWidth) < Sheet length (SheetLength)) (0.0 < Sheet synchronization width (SheetSyncWidth) < Cycle length/Number of cutters (NumberOfCutter)) • 0.00000000000001 to 10000000000.0
SyncAxisLength	Synchronous axis length (+52)	LREAL	The rotary cutter axis length is set. ■When Auto-generation options is set to "0: Diameter" Calculated as "Cycle length = setting value × π". ■When Auto-generation options is set to "1: Cycle length" Calculated as "Cycle length = setting value". (0.0 < Cycle length < 10000000000.0) • 0.00000000000001 to 10000000000.0

Variable name	Name (device offset)	Data type	Description
SyncPositionAdjustment	Synchronous position adjustment (+56)	LREAL	The position adjustment value of the synchronous section is set. Negative value: The synchronous section is adjusted to the sheet start side. 0: The center of the sheet is in the synchronous section. Positive value: The synchronous section is adjusted to the sheet end side. (Absolute value of Synchronous position adjustment (SyncPositionAdjustment) < Sheet length (SheetLength) / 2) • -10000000000.0 to 10000000000.0
AccDecWidth	Acceleration/deceleration width (+60)	LREAL	The sheet width (one side) of the acceleration/deceleration area is set. When a negative value is set, the acceleration/deceleration width is calculated to be the maximum. (2 × Acceleration/deceleration width (AccDecWidth) ≤ Sheet length (SheetLength) - Sheet synchronization width (SheetSyncWidth)) • -10000000000.0 to 10000000000.0
AsyncSpd	Asynchronous speed result (+64)	LREAL	When the auto-generation is successfully completed, the asynchronous speed is stored as the ratio of the synchronous speed.

PROFILE_CAMSWITCH_DATA

Variable name	Name (device offset)	Data type	Description
InputUnitString	Input unit string (+0)	WSTRING (31)	The unit for input data is set as a string.
NumberOfSwitches	Number of switches (+32)	DWORD (UDINT)	The total number of switches is specified. • 1 to 10000
Options	Options (+34)	DWORD(HEX)	Options are set in the specified bit position. Set bits 3 to 31 to "0".  ■Periodic (Bit 0)*1 • 0: Non periodic • 1: Periodic ■Master axis absolute coordinate (bit 1)*1 • 0: Relative coordinate • 1: Absolute coordinate ■Slave axis absolute position (Bit 2)*1 • 0: Relative coordinate • 1: Absolute coordinate
ReferenceSpeed	Reference speed (+36)	LREAL	This setting is not used. Set 0.0.
ReferenceSpeedUnitString	Reference speed unit string (+40)	WSTRING (31)	This setting is not used. Leave it empty.
RingCount_LowerValue	Ring counter lower limit value (+72)	LREAL	The ring counter lower limit value is set.
RingCount_UpperValue	Ring counter upper limit value (+76)	LREAL	The ring counter upper limit value is set.

*1 Specify "1".

PROFILE_CAMSWITCH_ELEMENT

Variable name	Name (device offset)	Data type	Description
FirstOnPosition	ON start position (+0)	LREAL	The lower limit position or the start position to turn on the switch is set. • -10000000000.0 to 10000000000.0
LastOnPosition	ON end position (+4)	LREAL	The upper limit position to turn on the switch is set. • -10000000000.0 to 10000000000.0
Direction	Direction selection (+8)	INT	The direction of the input to turn on the switch is set. • 0: Both direction • 1: Positive direction • 2: Negative Direction
CamSwitchMode	Switch mode selection (+9)	INT	The specification method for the ON condition is set. • 0: Switch disabled • 1: Position reference
Duration	ON time (+12)	LREAL	This setting is not used. The setting value is ignored.

PROFILE_POSITIONING_DATA

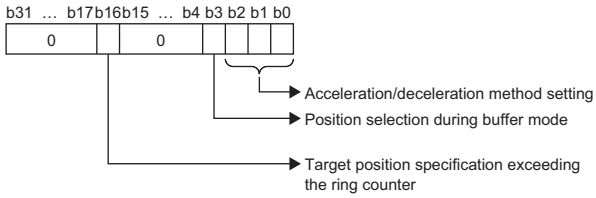
Variable name	Name (device offset)	Data type	Description
NumberOfDataSettings	Number of positioning data settings (+0)	WORD (UINT)	The number of positioning data is set. Set the value so that Number of positioning data settings (NumberOfDataSettings) is equal to or less than Total number of positioning data (TotalNumberOfData). • 0 to 5000 If a value out of range or exceeding the total number of positioning data is set, Operation profile data incorrect (error code: 1B8FH) occurs when the value is applied.
TotalNumberOfData	Total number of positioning data (+1)	WORD (UINT)	The total number of positioning data is set. The maximum value of the positioning data numbers to be set in application or writing is set. • 1 to 5000 If a value out of range is set, Operation profile data incorrect (error code: 1B8FH) occurs when the value is applied.
Options	Options (+2)	DWORD (HEX)	The options are set. Set "00000000H".

PROFILE_POSITIONING_DATA_ELEMENT

Variable name	Name (device offset)	Data type	Description
DataNo	Positioning data number (+0)	WORD (UINT)	The index number of the positioning data is set. The positioning data is executed in the order of Positioning data numbers (DataNo). The positioning data is set so that Positioning data numbers (DataNo) are sorted in ascending order. • 1 to Total number of positioning data (5000)
OperationPattern	Operation pattern (+1)	INT (MC_OPERATION_PATTERN) ☞ Page 1479 MC_OPERATION_PATTERN	The operation pattern of the positioning data is set. Whether to execute positioning data numbers successively or terminate the process is set. • 0: Positioning Complete (PositioningComplete) • 1: Continuous Positioning (ContinuousPositioning) • 2: Continuous Path (BlendingLow) (ContinuousBlendingLow) • 3: Continuous Path (BlendingPrevious) (ContinuousBlendingPrevious) • 4: Continuous Path (BlendingNext) (ContinuousBlendingNext) • 5: Continuous Path (BlendingHigh) (ContinuousBlendingHigh)
ControlMethod	Control method (+2)	INT (MC_CONTROL_METHOD) ☞ Page 1479 MC_CONTROL_METHOD	The control method for positioning data numbers is set. • 0080H: NOP (NOP) • 0082H: JUMP (JUMP) • 0083H: LOOP (LOOP) • 0084H: LEND (LEND) • 0101H: Absolute Value Linear Interpolation (LinearAbsolute) • 0102H: Relative Value Linear Interpolation (LinearRelative) • 0103H: Absolute Value Circular Interpolation (CircularAbsolute) • 0104H: Relative Value Circular Interpolation (CircularRelative) • 0105H: Absolute Value Helical Interpolation (HelicalAbsolute) • 0106H: Relative Value Helical Interpolation (HelicalRelative)
InterpolationAxis1	Interpolation axis 1 (+3)	WORD (UINT)	An interpolation axis that performs interpolation control from among the structuring axes in the axes group is set. Set the linear interpolation axes of linear interpolation and the circular interpolation axes of circular interpolation by the setting of control method (ControlMethod). For configuration axes to be specified as interpolation axes, different configuration axes and different number of interpolation axes can be set for each positioning data. However, note that if the specified configuration axis or the number of interpolation axes differs between the continuous path data and the next positioning data, the speed of each axis may suddenly change when the positioning data is switched. • 1 to 16
InterpolationAxis2	Interpolation axis 2 (+4)	WORD (UINT)	
InterpolationAxis3	Interpolation axis 3 (+5)	WORD (UINT)	
InterpolationAxis4	Interpolation axis 4 (+6)	WORD (UINT)	
CircMode	Circular interpolation mode (+7)	INT (MC_CIRC_MODE) ☞ Page 1475 MC_CIRC_MODE	The circular interpolation control specification method is set. • 0: Border point specification (mcBorder) • 1: Center point specification (mcCenter) • 2: Radius specification (mcRadius)

Variable name	Name (device offset)	Data type	Description
CircPathChoice	Path selection (+8)	INT (MC_CIRC_PATHCHOICE) ☞ Page 1475 MC_CIRC_PATHCHOICE	The rotation direction for circular interpolation control is set. • 0: CW (mcCW) • 1: CCW (mcCCW) • 2: Shortcut (mcShortWay) • 3: Detour (mcLongWay) • 4: CW Detour (mcCWLongWay) • 5: CCW Detour (mcCCWLongWay)
Reserve1 [0..2]	Reserve 1 (+9)	WORD (UINT)	Not used (For offset adjustment)
Position1	Target position/movement amount/end point 1 (+12)	LREAL	The following settings are applied according to the Control method (ControlMethod) setting. <Linear interpolation> For the target position/movement amount/end points 1 to 4 (Position1 to Position4), the target positions and movement amounts of the structuring axes set for the interpolation axes 1 to 4 (InterpolationAxis1 to InterpolationAxis4) are set. • -10000000000.0 to 10000000000.0 "0" or the target position/movement amount/end points 1 to 4 (Position1 to Position4) corresponding to the omitted interpolation axes 1 to 4 (InterpolationAxis1 to InterpolationAxis4) can be omitted. ■When "0101H: Absolute Value Linear Interpolation (LinearAbsolute)" is set The target position of the absolute position is set. ■When "0102H: Relative Value Linear Interpolation (LinearRelative)" is set The movement amount from the current position at execution of positioning data is set. <Circular interpolation> For the target position/movement amount/end points 1 and 2 (Position1 and Position2), the target positions and movement amounts of the structuring axes set for the interpolation axes 1 and 2 (InterpolationAxis1 and InterpolationAxis2) are set. • -10000000000.0 to 10000000000.0 The target position/movement amount/end points 3 and 4 (Position3 and Position4) can be omitted. (The setting is ignored.) ■When "0103H: Absolute Value Circular Interpolation (CircularAbsolute)" is set The end point of the absolute position is set. ■When "0104H: Relative Value Circular Interpolation (CircularRelative)" is set The movement amount from the current position at execution of positioning data to the end point is set.
Position2	Target position/movement amount/end point 2 (+16)	LREAL	
Position3	Target position/movement amount/end point 3 (+20)	LREAL	
Position4	Target position/movement amount/end point 4 (+24)	LREAL	
Direction1	Direction selection 1 (+28)	INT (MC_DIRECTION) ☞ Page 1474 MC_DIRECTION	The direction used to move from the current position to the target position during linear interpolation control is set. The direction is set when "0101H: Absolute Value Linear Interpolation (LinearAbsolute)" is set for the Control method (ControlMethod). For Direction selection 1 to 4 (Direction1 to Direction4), the directions of the structuring axes set for Interpolation axes 1 to 4 (InterpolationAxis1 to InterpolationAxis4) are set. • 1: Positive direction (mcPositiveDirection) • 2: Negative direction (mcNegativeDirection) • 3: Shortest path (mcShortestWay) The behavior will be as follows according to this setting. • An interpolation axis where the software stroke limit is valid ignores the setting. • "0" or the array elements of Direction selection 1 to 4 (Direction1 to Direction4) corresponding to omitted interpolation axes 1 to 4 (InterpolationAxis1 to InterpolationAxis4) can be omitted.
Direction2	Direction selection 2 (+29)	INT (MC_DIRECTION) ☞ Page 1474 MC_DIRECTION	
Direction3	Direction selection 3 (+30)	INT (MC_DIRECTION) ☞ Page 1474 MC_DIRECTION	
Direction4	Direction selection 4 (+31)	INT (MC_DIRECTION) ☞ Page 1474 MC_DIRECTION	

Variable name	Name (device offset)	Data type	Description
CircAuxPoint1	Sub point 1 (+32)	LREAL	The position of the sub point (border point, center point, or radius) used to perform circular interpolation control is set. The setting range is as follows according to the Circular interpolation mode (CircMode) setting. ■When "0: Border point specification (mcBorder)" or "1: Center point specification (mcCenter)" is set The following is applied according to Control method (ControlMethod). • "0103H: Absolute Value Circular Interpolation (CircularAbsolute)": -10000000000.0 ≤ Setting value < 10000000000.0 When the ring counter is enabled, the ring counter range is used. • "0104H: Relative Value Circular Interpolation (CircularRelative)": -10000000000.0 to 10000000000.0 ■When "2: Radius specification (mcRadius)" is set • 0.000001 to 2147483647.0
CircAuxPoint2	Sub point 2 (+36)	LREAL	
CircAuxPoint3	Sub point 3 (+40)	LREAL	
CircAuxPoint4	Sub point 4 (+44)	LREAL	
VelocityMode	Speed mode (+48)	INT (MC_INTERPOLATE_SPEED_MODE) ☞ Page 1476 MC_INTERPOLATE_SPEED_MODE	The speed mode for linear interpolation control is set. • 0: Vector speed (VectorSpeed) • 1: Long axis speed (LongAxisSpeed) • 2: Reference axis speed (ReferenceAxisSpeed)
Reserve2 [0..2]	Reserve 2 (+49)	WORD (UINT)	Not used (For offset adjustment)
Velocity	Velocity (+52)	LREAL	The positioning speed of multiple axes interpolation. • 0.0000, 0.0001 to 2500000000.0
Acceleration	Acceleration (+56)	LREAL	The acceleration of multiple axes interpolation. The setting range is as follows according to the acceleration/deceleration method setting (bit 0 to 2) of Positioning data options (DataOptions). ■When "0: Acceleration/deceleration specification method (mcAccDec)" is set • 0.0000, 0.0001 to 2147483647.0 [U/s^2] ■When "1: Acceleration/deceleration time-fixed method (mcFixedTime)" is set • 0.000000, 0.000001 to 8400.0 [s]
Deceleration	Deceleration (+60)	LREAL	The deceleration of multiple axes interpolation. ■When "0: Acceleration/deceleration specification method (mcAccDec)" is set • 0.0000, 0.0001 to 2147483647.0 [U/s^2] ■When "1: Acceleration/deceleration time-fixed method (mcFixedTime)" is set • Deceleration (Deceleration) is not used.
Jerk	Jerk (+64)	LREAL	The jerk of multiple axes positioning control is set. ■When "0: Acceleration/deceleration specification method (mcAccDec)" is set • 0.0000, 0.0001 to 2147483647.0 [U/s^3] ■When "1: Acceleration/deceleration time-fixed method (mcFixedTime)" is set • Jerk (Jerk) is not used.
CircErrorTolerance	Circular interpolation error tolerance (+68)	LREAL	The circular interpolation error tolerance is set for circular interpolation with the center specified. • 0.000001 to 100000.0

Variable name	Name (device offset)	Data type	Description
DataOptions	Positioning data options (+72)	DWORD (HEX)	Options are set in the specified bit position.  ■Acceleration/deceleration method setting (bit 0 to 2) • 0: Acceleration/deceleration specification method (mcAccDec) • 1: Acceleration/deceleration time-fixed method (mcFixedTime) ■Position selection during buffer mode (Bit 3) This option is set when "0102H: Relative Value Linear Interpolation (LinearRelative)" or "0104H: Relative Value Circular Interpolation (CircularRelative)" is set for Control method (ControlMethod). • 0: Set position • 1: Actual position Set "0" when "0101H: Absolute Value Linear Interpolation (LinearAbsolute)" or "0103H: Absolute Value Circular Interpolation (CircularAbsolute)" is set. ■Target position specification exceeding the ring counter (bit 16) This option is set when "0101H: Absolute Value Linear Interpolation (LinearAbsolute)" is set for Control method (ControlMethod). • 0: Disabled • 1: Enabled Set "0" when "0102H: Relative Value Linear Interpolation (LinearRelative)", "0103H: Absolute Value Circular Interpolation (CircularAbsolute)", or "0104H: Relative Value Circular Interpolation (CircularRelative)" is set.
Reserve3 [0..1]	Reserve 3 (+74)	WORD (UINT)	Not used (For offset adjustment)
DwellTime	Dwell time (+76)	LREAL	The dwell time is set. • 0.0[s]: Dwell function disabled • 0.000001 to 8400.0[s]: Dwell function enabled When a positive number less than 0.000001 is set, the value is imported as "0.0[s]".
Mcode	M code (+80)	WORD (UINT)	The M code is set for each positioning data. When the M code is not output, set "0" (initial value). • 1 to 65535
McodeOutput_Override	M code output timing override (+81)	INT (MC_MCODE_OUTPUT_OVERRIDE) ☞ Page 1479 MC_MCODE_OUTPUT_OVERRIDE	The M code output timing is set for each positioning data. • 0: Use FB Options (UseFbOptions) • 1: WITH Mode (WithMode) • 2: AFTER Mode (AfterMode)
JumpDestinationDataNo	Jump destination positioning data number (+82)	WORD (UINT)	The positioning data number is set for the jump destination specified with JUMP. Positioning data number (DataNo) is set for positioning data whose Control method (ControlMethod) is other than "0082H: JUMP (JUMP)". • 1 to Total number of positioning data (5000)
ConditionSignalNo	Condition signal number (+83)	WORD (UINT)	The condition signal number is set. When the condition signal is not used, set "0" (initial value). • 1 to 10
LoopCount	Number of loops from LOOP to LEND (+84)	WORD (UINT)	The number of loop control repetitions is set. • 1 to 65535
SkipSignalNo	Skip signal number (+85)	WORD (UINT)	The skip signal number is set. • 1 to 10 When the skip signal is not used, set "0".
Pitch	Pitch (+86)	WORD (UINT)	The pitch is set. Set "0".
Reserve4	Reserve 4 (+87)	WORD (UINT)	Not used (For offset adjustment)

PROFILE_POSITIONING_DATA_EXTENDED

Variable name	Name (device offset)	Data type	Description
ConditionSignal	Condition signal (+0)	WSTRING (63)	The condition signal is set as a TARGET_REF-type string. For details, refer to the following. ☞ Page 1444 TARGET_REF (Input signal) The setting range of strings is as follows.*1 ■Data classification • [VAR] • [DEV] ■Type • PROFILE_SIGNAL_SELECT_ELEMENT (☞ Page 1459 PROFILE_SIGNAL_SELECT_ELEMENT) When "" (no setting) is set, reading/writing is not performed.
SkipSignal	Skip signal (+64)	WSTRING (63)	The skip signal is set as a TARGET_REF-type string. For details, refer to the following. ☞ Page 1444 TARGET_REF (Input signal) The setting range of strings is as follows.*1 ■Data classification • [VAR] • [DEV] ■Type • PROFILE_SIGNAL_SELECT_ELEMENT (☞ Page 1459 PROFILE_SIGNAL_SELECT_ELEMENT) When "" (no setting) is set, reading/writing is not performed.
PositioningData	Positioning data (+128)	WSTRING (63)	The positioning data is set as a TARGET_REF-type string. For details, refer to the following. ☞ Page 1444 TARGET_REF (Input signal) The setting range of strings is as follows. ■Data classification • [VAR] • [DEV] ■Type • POFILE_POSITIONING_DATA_ELEMENT (☞ Page 1454 PROFILE_POSITIONING_DATA_ELEMENT) When "" (no setting) is set, Read/Write data type incorrect (error code: 1BAAH) occurs.

*1 If the specified type does not satisfy the requirements of individual structure members, Read/Write data type incorrect (error code: 1BAAH) occurs. Also, if the specified data type does not satisfy the requirements of individual structure members, Read/Write data data type incorrect (error code: 1BA8H) occurs.

PROFILE_SIGNAL_SELECT_ELEMENT

Variable name	Name (device offset)	Data type	Description*1
StartIO	I/O number (+0)	WORD (HEX)	Not used The setting is ignored.
Reserve1	Reserve 1 (+1)	WORD (UINT)	Not used (For offset adjustment)
Target	Target (+2)	WSTRING (63)	The signal to be used is set as a string. For details, refer to the following. ☞ Page 1444 TARGET_REF (Input signal) The setting range of strings is as follows. • "" (string with a length of 0)*2 ■Data classification • [OBJ]*3 • [VAR] • [DEV] ■Type • BOOL
Detection	Signal detection method (+66)	INT (MC_SIGNAL_LOGIC) ☞ Page 1478 MC_SIGNAL_LOGIC	The signal logic is specified. • 0: Detection at TRUE (HighLevel) • 1: Detection at FALSE (LowLevel)
Reserve2	Reserve 2 (+67)	WORD (UINT)	Not used (For offset adjustment)
CompensationTime	Compensation time (+68)	LREAL	"0.0" is set.
FilterTime	Filter time (+72)	LREAL	The filter time for removing chattering of the I/O signal is set in units of seconds. • 0.0 to 5.0[s]

*1 Errors due to setting values do not occur during writing. An error occurs if the setting is out of range when an operation profile data control FB uses a written open area.

*2 If Target (Target) is a zero-length string (""), the applicable skip signal is handled without any setting.

*3 If [OBJ] is set, the target modification cannot be omitted. If the setting is complete with the target modification omitted, Operation profile data control profile ID setting incorrect warning (event code: 00D48H) occurs.

SLAVE_OBJECT_REAL

Variable name	Name	Data type	Description
ControlWord	ControlWord	WSTRING(63)	Switching of the status is requested to the drive unit.
EncoderIncrements	EncoderIncrements	WSTRING(63)	The encoder resolution of the drive unit is acquired.
FollowingErrActualValue	FollowingErrActualValue	WSTRING(63)	The pulse droop of the drive unit
GearRatioMotorRevolutions	GearRatioMotorRevolutions	WSTRING(63)	The number of servo motor axis rotations (numerator) is set to the drive unit.
HomeOffset	HomeOffset	WSTRING(63)	The difference between zero position and homing position of the machine coordinate system is set to the drive unit.
MaxMotorSpeed	MaxMotorSpeed	WSTRING(63)	The maximum speed of the servo motor is acquired from the drive unit.
MaxTorque	MaxTorque	WSTRING(63)	The maximum torque of the servo motor is acquired from the drive unit.
ModesOfOp	ModesOfOp	WSTRING(63)	Switching of the control mode is requested to the drive unit.
ModesOfOpDisp	ModesOfOpDisp	WSTRING(63)	The drive unit control mode is acquired.
NegativeTorqueLimitValue	NegativeTorqueLimitValue	WSTRING(63)	The negative direction torque limit value is set to the drive unit.
Polarity	Polarity	WSTRING(63)	The rotation direction selection of the drive unit is set.
PosActualValue	PosActualValue	WSTRING(63)	The current position of the drive unit
PosEncoderResolution	PosEncoderResolution	WSTRING(63)	The total number of Sub Index (=2) is returned.
PosEncoderResolutionMotorRevolutions	PosEncoderResolutionMotorRevolutions	WSTRING(63)	The revolution of the drive unit
PositiveTorqueLimitValue	PositiveTorqueLimitValue	WSTRING(63)	The positive direction torque limit value is set to the drive unit.
ShaftRevolutions	ShaftRevolutions	WSTRING(63)	The number of drive axis rotations (denominator) of the drive unit is set.
SiUnitPos	SiUnitPos	WSTRING(63)	The SI unit position of the drive unit is acquired.
SiUnitVel	SiUnitVel	WSTRING(63)	The SI unit velocity of the drive unit is acquired.
StatusWord	StatusWord	WSTRING(63)	The drive unit status is acquired.
SupportedDriveModes	SupportedDriveModes	WSTRING(63)	The control mode supported by the drive unit is acquired.
TargetPos	TargetPos	WSTRING(63)	The command position output to the drive unit
TargetTorque	TargetTorque	WSTRING(63)	The command torque output to the drive unit
TargetVelocity	TargetVelocity	WSTRING(63)	The command velocity output to the drive unit
TargetActualValue	TorqueActualValue	WSTRING(63)	The current torque of the drive unit
VelActualValue	VelActualValue	WSTRING(63)	The current speed of the drive unit
vControlDil1	ControlDi1	WSTRING(63)	The input device is set to the drive unit.
vControlDil2	ControlDi2	WSTRING(63)	The input device is set to the drive unit.
vControlDil3	ControlDi3	WSTRING(63)	The input device is set to the drive unit.
vControlDil4	ControlDi4	WSTRING(63)	The input device is set to the drive unit.
vControlDil5	ControlDi5	WSTRING(63)	The input device is set to the drive unit.
vControlDil6	ControlDi6	WSTRING(63)	The input device is set to the drive unit.
vControlDil7	ControlDi7	WSTRING(63)	The input device is set to the drive unit.
vCspr_ForwardSideStopFunction	ForwardSideStopFunction	WSTRING (63)	The forward-side position output to the drive unit
vCspr_PressureActualValue	PressureActualValue	WSTRING (63)	The current pressure of the drive unit
vCspr_TargetPressure	TargetPressure	WSTRING (63)	The command pressure output to the drive unit
vCurrentAlarm	CurrentAlarm	WSTRING(63)	The occurring alarm is acquired from the drive unit.
vEncoderStatus1	EncoderStatus1	WSTRING(63)	The encoder status is acquired from the drive unit.
vHomeAbsCounter	HomeAbsCounter	WSTRING(63)	The encoder multiple revolution counter saved as the home position is acquired from the drive unit.
vHomeCycleCounter	HomeCycleCounter	WSTRING(63)	The encoder position within one revolution at power-on is acquired from the drive unit.
vInitialAbsCounter	InitialAbsCounter	WSTRING(63)	The encoder multiple revolution counter at power-on is acquired from the drive unit.
vInitialCycleCounter	InitialCycleCounter	WSTRING(63)	The encoder position within one revolution at power-on is acquired from the drive unit.
vInitialPos	InitialPos	WSTRING(63)	The current position at power-on is acquired from the drive unit.
vMaxAbsCounter	MaxAbsCounter	WSTRING(63)	The maximum value of the encoder multiple revolution counter is acquired from the drive unit.

Variable name	Name	Data type	Description
vStatusDo1	StatusDo1	WSTRING(63)	The output device status of the drive unit is acquired.
vStatusDo2	StatusDo2	WSTRING(63)	The output device status of the drive unit is acquired.
vStatusDo3	StatusDo3	WSTRING(63)	The output device status of the drive unit is acquired.
vStatusDo4	StatusDo4	WSTRING(63)	The output device status of the drive unit is acquired.
vStatusDo5	StatusDo5	WSTRING(63)	The output device status of the drive unit is acquired.
vStatusDo6	StatusDo6	WSTRING(63)	The output device status of the drive unit is acquired.
vStatusDo7	StatusDo7	WSTRING(63)	The output device status of the drive unit is acquired.
vSupportedControlDi1	SupportedControlDi1	WSTRING(63)	The input device supported by the drive unit is acquired.
vSupportedControlDi2	SupportedControlDi2	WSTRING(63)	The input device supported by the drive unit is acquired.
vSupportedControlDi3	SupportedControlDi3	WSTRING(63)	The input device supported by the drive unit is acquired.
vSupportedControlDi4	SupportedControlDi4	WSTRING(63)	The input device supported by the drive unit is acquired.
vSupportedControlDi5	SupportedControlDi5	WSTRING(63)	The input device supported by the drive unit is acquired.
vSupportedControlDi6	SupportedControlDi6	WSTRING(63)	The input device supported by the drive unit is acquired.
vSupportedControlDi7	SupportedControlDi7	WSTRING(63)	The input device supported by the drive unit is acquired.
vSupportedStatusDo1	SupportedStatusDo1	WSTRING(63)	The output device supported by the drive unit is acquired.
vSupportedStatusDo2	SupportedStatusDo2	WSTRING(63)	The output device supported by the drive unit is acquired.
vSupportedStatusDo3	SupportedStatusDo3	WSTRING(63)	The output device supported by the drive unit is acquired.
vSupportedStatusDo4	SupportedStatusDo4	WSTRING(63)	The output device supported by the drive unit is acquired.
vSupportedStatusDo5	SupportedStatusDo5	WSTRING(63)	The output device supported by the drive unit is acquired.
vSupportedStatusDo6	SupportedStatusDo6	WSTRING(63)	The output device supported by the drive unit is acquired.
vSupportedStatusDo7	SupportedStatusDo7	WSTRING(63)	The output device supported by the drive unit is acquired.
vSyncAbsCounter	SyncAbsCounter	WSTRING(63)	The multiple revolution counter of the scale measurement encoder is acquired from the drive unit.
vSyncCycleCounter	SyncCycleCounter	WSTRING(63)	The position within one revolution of the scale measurement encoder is acquired from the drive unit.
vVelLimitValue	VelLimitValue	WSTRING(63)	The velocity limit value is set to the drive unit.
vWatchDogCounterDI	WatchDogCounterDI	WSTRING(63)	The watchdog counter value is notified to the drive unit.
vWatchDogCounterUI	WatchDogCounterUI	WSTRING(63)	The watchdog counter value is acquired from the drive unit.
vEncoderStatus2	EncoderStatus2	WSTRING(63)	The scale measurement encoder status is acquired from the drive unit.
vScaleCycleCounter	ScaleCycleCounter	WSTRING(63)	The position within one revolution of the scale measurement encoder is acquired from the drive unit.
vScaleAbsCounter	ScaleAbsCounter	WSTRING(63)	The multiple revolution counter of the scale measurement encoder is acquired from the drive unit.
vScaleMeasurementEncoderResolution	ScaleMeasurementEncoderResolution	WSTRING(63)	The scale measurement encoder resolution is acquired from the drive unit.
vScaleMeasurementEncoderReceptionStatus	ScaleMeasurementEncoderReceptionStatus	WSTRING(63)	The encoder position within one revolution is acquired from the drive unit.

SLAVE_OBJECT_ENCODER

Variable name	Name	Data type	Description
PosActualValue	PosActualValue	WSTRING(63)	The data storing the synchronous encoder input is set. When the I/O device is connected, imports the value from the set location.

SLAVE_OBJECT_VIRTUAL_ENCODER

Variable name	Name	Data type	Description
PosActualValue	PosActualValue	WSTRING(63)	The data used as the encoder input value is set in the string format. [VAR], [DEV], and [CONST] can be specified for the type. ■When [VAR] is set • Data of the (INT), (DINT), (WORD), and (DWORD) data types is set. • When [VAR] is used, local labels cannot be set. ■When [DEV] or [CONST] is set • Data of the (INT), (DINT), (WORD), and (DWORD) data types is set. *: When this setting is omitted, Encoder input value (AxisName.Cd.Encoder_InputValue) is used.

Other structure variable (Advanced synchronous control)

ADV_GEAR_PARAM (Gear parameter)

Variable name	Name	Import	Data type	Attribute	Description
MasterCompositeGear	Master axis composite gear	Operation cycle (FB is starting)	WORD (HEX)	LIST_WRITE_BACK	The method of combining the current positions of the master axis and sub input axis is set. • 0000H: No master axis input/no sub input axis input • 0001H: Positive master axis input and no sub-input axis input • 0002H: Negative master axis input and no sub-input axis input • 0010H: No master axis input and positive sub-input axis input • 0011H: Positive master axis input and positive sub-input axis input • 0012H: Negative master axis input and positive sub-input axis input • 0020H: No master axis input and negative sub-input axis input • 0021H: Positive master axis input and negative sub-input axis input • 0022H: Negative master axis input and negative sub-input axis input
MasterNumerator	Master axis gear numerator	At start	DINT	LIST_WRITE_BACK	The master axis gear numerator is set. • -2147483647 to 2147483647
MasterDenominator	Master axis gear denominator	At start	DWORD (UDINT)	LIST_WRITE_BACK	The master axis gear denominator is set. • 1 to 2147483647
AuxCompositeGear	Auxiliary axis composite gear	Operation cycle (FB is starting)	WORD (HEX)	LIST_WRITE_BACK	The method of combining the current position after the master axis and sub input axis are combined and the current position of the auxiliary axis is set. • 0000H: No master axis input and no auxiliary axis input • 0001H: Positive master axis input and no auxiliary axis input • 0002H: Negative master axis input and no auxiliary axis input • 0010H: No master axis input and positive auxiliary axis input • 0011H: Positive master axis input and positive auxiliary axis input • 0012H: Negative master axis input and positive auxiliary axis input • 0020H: No master axis input and negative auxiliary axis input • 0021H: Positive master axis input and negative auxiliary axis input • 0022H: Negative master axis input and negative auxiliary axis input
AuxNumerator	Auxiliary axis gear numerator	At start	DINT	LIST_WRITE_BACK	The auxiliary axis gear numerator is set. • -2147483647 to 2147483647
AuxDenominator	Auxiliary axis gear denominator	At start	DWORD (UDINT)	LIST_WRITE_BACK	The auxiliary axis gear denominator is set. • 1 to 2147483647

ADV_CLUTCH_PARAM (Clutch parameter)

Variable name	Name	Import	Data type	Attribute	Description
MasterOnControl	Master axis ON clutch control setting	Operation cycle (FB is starting)	INT (MC_CLUTCH_METHOD) ☞ Page 1480 MC_CLUTCH_METHOD	LIST_WRITE_BACK	The ON control method for the master axis clutch is set. • 0: Invalid (NoClutch) (Direct operation) • 1: Clutch command (ClutchCommand) • 2: Clutch command rising edge (ClutchLeadingEdge) • 3: Clutch command falling edge (ClutchTrailingEdge) • 4: Address mode (ClutchAddress) • 15: I/O data specification (ClutchSignal) *: The value can be changed during synchronization control. However, it cannot be changed from the setting other than "0: Invalid (NoClutch)" to "0: Invalid (NoClutch)".
MasterOffControl	Master axis OFF clutch control setting	Operation cycle (FB is starting)	INT (MC_CLUTCH_METHOD) ☞ Page 1480 MC_CLUTCH_METHOD	LIST_WRITE_BACK	The OFF control method for the master axis clutch is set. • 0: Invalid (NoClutch) (OFF control invalid) • 1: Clutch command (ClutchCommand) (One-shot operation) • 2: Clutch command rising edge (ClutchLeadingEdge) • 3: Clutch command falling edge (ClutchTrailingEdge) • 4: Address mode (ClutchAddress) • 15: I/O data specification (ClutchSignal)
MasterReference	Master axis clutch reference address setting	At start	INT (MC_CLUTCH_REFERENCE) ☞ Page 1480 MC_CLUTCH_REFERENCE	LIST_WRITE_BACK	The address to be referenced during master axis clutch control is set. • 0: Current position before gear (GearFrontPosition) • 1: Current position per cycle after gear (GearPositionPerCycle) *: Note that the reference address changes the processing order of gears and clutches.
MasterOnAddress	Master axis clutch ON address	Operation cycle (FB is starting)	LREAL	LIST_WRITE_BACK	The address for turning on the clutch is set when Master axis ON clutch control setting (AdvOutputName.Pr.Clutch.MasterOnControl) is set to "4: Address mode (ClutchAddress)". When Master axis clutch reference address setting (AdvOutputName.Pr.Clutch.MasterReference) is set to "1: Current position per cycle after gear (GearPositionPerCycle)", the set value is converted for control within the range from 0.0 to (Length per cycle - 0.00001). • -10000000000.0 to 10000000000.0
MasterMovementAmountBeforeOn	Movement amount before master axis clutch ON	When the clutch ON condition is met	LREAL	LIST_WRITE_BACK	A signed value for the movement amount of the reference address, from the point when the ON condition is met in the master axis clutch ON control to when the clutch is actually turned on, is set. • -2147483648.0 to 2147483647.0
MasterOffAddress	Master axis clutch OFF address	Operation cycle (FB is starting)	LREAL	LIST_WRITE_BACK	The address for turning off the clutch is set when Master axis OFF clutch control setting (AdvOutputName.Pr.Clutch.MasterOffControl) is set to "4: Address mode (ClutchAddress)". When Master axis clutch reference address setting (AdvOutputName.Pr.Clutch.MasterReference) is set to "1: Current position per cycle after gear (GearPositionPerCycle)", the set value is converted for control within the range from 0.0 to (Length per cycle - 0.00001). • -10000000000.0 to 10000000000.0
MasterMovementAmountBeforeOff	Master axis clutch OFF preliminary movement amount	When the clutch OFF condition is met*1	LREAL	LIST_WRITE_BACK	A signed value for the movement amount of the reference address, from the point when the OFF condition is met in the master axis clutch OFF control to when the clutch is actually turned off, is set. • -2147483648.0 to 2147483647.0
MasterSmoothingMethod	Master axis clutch smoothing method	At start	INT (MC_CLUTCH_SMOOTHING_METHOD) ☞ Page 1480 MC_CLUTCH_SMOOTHING_METHOD	LIST_WRITE_BACK	The smoothing method for the time when the master axis clutch is set to on/off is set. • 0: Direct (ClutchSmoothingDisabled) • 1: Time constant method (exponent) (TimeConstantExponent) • 2: Time constant method (linear) (TimeConstantLinear) • 3: Slippage amount method (exponent) (SlippageExponent) • 4: Slippage amount method (linear) (SlippageLinear) • 5: Slippage amount method (linear: input amount follow-up) (SlippageLinearFollow)

Variable name	Name	Import	Data type	Attribute	Description
MasterOnSlippageAmount	Master axis clutch ON slippage amount	At clutch ON start	LREAL	LIST_WRITE_BACK	The slippage amount at which the clutch turned on is set when Master axis clutch smoothing method (AdvOutputName.Pr.Clutch.MasterSmoothingMethod) is set to "3: Slippage amount method (exponent) (SlippageExponent)", "4: Slippage amount method (linear) (SlippageLinear)", or "5: Slippage amount method (linear: input amount follow-up) (SlippageLinearFollow)". • 0.0 to 2147483647.0 *: When the setting value is negative, the control is applied on the assumption that Master axis clutch ON slippage amount (AdvOutputName.Pr.Clutch.MasterOnSlippageAmount) is set to "0.0 (Direct)".
MasterOffSlippageAmount	Master axis clutch OFF slippage amount	At clutch OFF start	LREAL	LIST_WRITE_BACK	The slippage amount at which the clutch turned off is set when Master axis clutch smoothing method (AdvOutputName.Pr.Clutch.MasterSmoothingMethod) is set to "3: Slippage amount method (exponent) (SlippageExponent)", "4: Slippage amount method (linear) (SlippageLinear)", or "5: Slippage amount method (linear: input amount follow-up) (SlippageLinearFollow)". • 0.0 to 2147483647.0 *: When the setting value is negative, the control is applied on the assumption that Master axis clutch OFF slippage amount (AdvOutputName.Pr.Clutch.MasterOffSlippageAmount) is set to "0.0 (Direct)".
AuxOnControl	Auxiliary axis ON clutch control setting	Operation cycle (FB is starting)	INT (MC_CLUTCH_METHOD) ☞ Page 1480 MC_CLUTCH_METHOD	LIST_WRITE_BACK	The ON control method for the auxiliary axis clutch is set. • 0: Invalid (NoClutch) (Direct operation) • 1: Clutch command (ClutchCommand) • 2: Clutch command rising edge (ClutchLeadingEdge) • 3: Clutch command falling edge (ClutchTrailingEdge) • 4: Address mode (ClutchAddress) • 15: I/O data specification (ClutchSignal) *: The value can be changed during synchronization control. However, it cannot be changed from the setting other than "0: Invalid (NoClutch) (Direct operation)" to "0: Invalid (NoClutch) (Direct operation)".
AuxOffControl	Auxiliary axis OFF clutch control setting	Operation cycle (FB is starting)	INT (MC_CLUTCH_METHOD) ☞ Page 1480 MC_CLUTCH_METHOD	LIST_WRITE_BACK	The OFF control method for the auxiliary axis clutch is set. • 0: Invalid (NoClutch) (OFF control invalid) • 1: Clutch command (ClutchCommand) (One-shot operation) • 2: Clutch command rising edge (ClutchLeadingEdge) • 3: Clutch command falling edge (ClutchTrailingEdge) • 4: Address mode (ClutchAddress) • 15: I/O data specification (ClutchSignal)
AuxReference	Auxiliary axis clutch reference address setting	At start	INT (MC_CLUTCH_REFERENCE) ☞ Page 1480 MC_CLUTCH_REFERENCE	LIST_WRITE_BACK	The address to be referenced during auxiliary axis clutch control is set. • 0: Current position before gear (GearFrontPosition) • 1: Current position per cycle after gear (GearPositionPerCycle) *: Note that the reference address changes the processing order of gears and clutches.
AuxOnAddress	Auxiliary axis clutch ON address	Operation cycle (FB is starting)	LREAL	LIST_WRITE_BACK	The address for turning on the clutch is set when Auxiliary axis ON clutch control setting (AdvOutputName.Pr.Clutch.AuxOnControl) is set to "4: Address mode (ClutchAddress)". When Auxiliary axis clutch reference address setting (AdvOutputName.Pr.Clutch.AuxReference) is set to "1: Current position per cycle after gear (GearPositionPerCycle)", the set value is converted for control within the range from 0.0 to (Length per cycle - 0.00001). • -10000000000.0 to 10000000000.0
AuxMovementAmountBeforeOn	Movement amount before auxiliary axis clutch ON	When the clutch ON condition is met	LREAL	LIST_WRITE_BACK	A signed value for the movement amount of the reference address, from the point when the ON condition is met in the auxiliary axis clutch ON control to when the clutch is actually turned on, is set. • -2147483648.0 to 2147483647.0

Variable name	Name	Import	Data type	Attribute	Description
AuxOffAddress	Auxiliary axis clutch OFF address	Operation cycle (FB is starting)	LREAL	LIST_WRITE_BACK	The address for turning off the clutch is set when Auxiliary axis OFF clutch control setting (AdvOutputName.Pr.Clutch.AuxOffControl) is set to "4: Address mode (ClutchAddress)". When Auxiliary axis clutch reference address setting (AdvOutputName.Pr.Clutch.AuxReference) is set to "1: Current position per cycle after gear (GearPositionPerCycle)", the set value is converted for control within the range from 0.0 to (Length per cycle - 0.00001). • -10000000000.0 to 10000000000.0
AuxMovementAmountBeforeOff	Movement amount before auxiliary axis clutch OFF	When the clutch OFF condition is met*1	LREAL	LIST_WRITE_BACK	A signed value for the movement amount of the reference address, from the point when the OFF condition is met in the auxiliary axis clutch OFF control to when the clutch is actually turned off, is set. • -2147483648.0 to 2147483647.0
AuxSmoothingMethod	Auxiliary axis clutch smoothing method	At start	INT (MC_CLUTCH_SMOOTHING_METHOD) ☞ Page 1480 MC_CLUTCH_SMOOTHING_METHOD	LIST_WRITE_BACK	The smoothing method for when the auxiliary axis clutch is set to on/off. • 0: Direct (ClutchSmoothingDisabled) • 1: Time constant method (exponent) (TimeConstantExponent) • 2: Time constant method (linear) (TimeConstantLinear) • 3: Slippage amount method (exponent) (SlippageExponent) • 4: Slippage amount method (linear) (SlippageLinear) • 5: Slippage amount method (linear: input amount follow-up) (SlippageLinearFollow)
AuxOnSlippageAmount	Auxiliary axis clutch ON slippage amount	At clutch ON start	LREAL	LIST_WRITE_BACK	The slippage amount at which the clutch turned on is set when Auxiliary axis clutch smoothing method (AdvOutputName.Pr.Clutch.AuxSmoothingMethod) is set to "3: Slippage amount method (exponent) (SlippageExponent)", "4: Slippage amount method (linear) (SlippageLinear)", or "5: Slippage amount method (linear: input amount follow-up) (SlippageLinearFollow)". • 0.0 to 2147483647.0 *: When the setting value is negative, the control is applied on the assumption that Auxiliary axis clutch ON slippage amount (AdvOutputName.Pr.Clutch.AuxOnSlippageAmount) is set to "0.0 (Direct)".
AuxOffSlippageAmount	Auxiliary axis clutch OFF slippage amount	At clutch OFF start	LREAL	LIST_WRITE_BACK	The slippage amount at which the clutch turned off is set when Auxiliary axis clutch smoothing method (AdvOutputName.Pr.Clutch.AuxSmoothingMethod) is set to "3: Slippage amount method (exponent) (SlippageExponent)", "4: Slippage amount method (linear) (SlippageLinear)", or "5: Slippage amount method (linear: input amount follow-up) (SlippageLinearFollow)". • 0.0 to 2147483647.0 *: When the setting value is negative, the control is applied on the assumption that Auxiliary axis clutch OFF slippage amount (AdvOutputName.Pr.Clutch.AuxOffSlippageAmount) is set to "0.0 (Direct)".

*1 When using one-shot operation, this is "When the clutch ON condition is met".

ADV_CLUTCH_MONI (Clutch monitor)

Variable name	Name	Import	Data type	Attribute	Description
MasterOnOffStatus	Master axis clutch ON/OFF status	Operation cycle (FB is starting)	BOOL	LIST_READ_ONLY	The on/off status of the master axis clutch is stored. • FALSE: Clutch OFF status • TRUE: Clutch ON status
MasterSmoothingStatus	Master axis clutch smoothing status	Operation cycle (FB is starting)	BOOL	LIST_READ_ONLY	The smoothing status of the master axis clutch is stored. • FALSE: No clutch smoothing • TRUE: Clutch smoothing in progress
MasterCumulativeSlippage	Master axis clutch cumulative slippage amount	Operation cycle (FB is starting)	LREAL	LIST_READ_ONLY	The cumulative slippage amount during clutch smoothing with a slippage method is stored with a signed value.
AuxOnOffStatus	Auxiliary axis clutch ON/OFF status	Operation cycle (FB is starting)	BOOL	LIST_READ_ONLY	The on/off status of the auxiliary axis clutch is stored. • FALSE: Clutch OFF status • TRUE: Clutch ON status
AuxSmoothingStatus	Auxiliary axis clutch smoothing status	Operation cycle (FB is starting)	BOOL	LIST_READ_ONLY	The smoothing status of the auxiliary axis clutch is stored. • FALSE: Not on clutch smoothing • TRUE: On clutch smoothing
AuxCumulativeSlippage	Auxiliary axis clutch cumulative slippage amount	Operation cycle (FB is starting)	LREAL	LIST_READ_ONLY	The cumulative slippage amount during clutch smoothing with a slippage method is stored with a signed value.

ADV_CLUTCH_CMD (Clutch control data)

Variable name	Name	Import	Data type	Attribute	Description
MasterCommand	Master axis clutch command	Operation cycle (FB is starting)	BOOL	LIST_READ_ONLY	Whether to turn on or off the master axis clutch command is set. This variable is used when Master axis ON clutch control setting (AdvOutputName.Pr.Clutch.MasterOnControl) is set to "1: Clutch command (ClutchCommand)", "2: Clutch command rising edge (ClutchLeadingEdge)", or "3: Clutch command falling edge (ClutchTrailingEdge)". • FALSE: The clutch command is turned off. • TRUE: The clutch command is turned on.
MasterInvalidCommand	Master axis clutch control invalid command	Operation cycle (FB is starting)	BOOL	LIST_READ_ONLY	The valid/invalid of master axis clutch control is switched. The on/off status of the clutch remains the status before clutch control becomes invalid. Clutch control does not become invalid during movement before clutch ON or before clutch OFF. Clutch control becomes invalid after movement before clutch ON or before clutch OFF. • FALSE: Clutch control is valid. • TRUE: Clutch control is invalid.
MasterForcedOff	Master axis clutch forced off command	Operation cycle (FB is starting)	BOOL	LIST_READ_ONLY	The master axis clutch is forcibly turned off. • FALSE: The clutch is controlled normally. • TRUE: The clutch is forcibly turned off. *: The output from the clutch immediately becomes "0" during the execution of clutch smoothing. *: When the slippage amount method smoothing is performed, the cumulative slippage amount is cleared to 0. *: When the setting value is changed to "FALSE" after the clutch is forcibly turned off, the normal clutch control restarts from the status in which the clutch is turned off.
AuxCommand	Auxiliary axis clutch command	Operation cycle (FB is starting)	BOOL	LIST_READ_ONLY	Whether to turn on or off the auxiliary axis clutch command is set. This variable is used when Auxiliary axis ON clutch control setting (AdvOutputName.Pr.Clutch.AuxOnControl) is set to "1: Clutch command (ClutchCommand)", "2: Clutch command rising edge (ClutchLeadingEdge)", or "3: Clutch command falling edge (ClutchTrailingEdge)". • FALSE: The clutch command is turned off. • TRUE: The clutch command is turned on.

Variable name	Name	Import	Data type	Attribute	Description
AuxInvalidCommand	Auxiliary axis clutch control invalid command	Operation cycle (FB is starting)	BOOL	LIST_READ_ONLY	The valid/invalid of auxiliary axis clutch control is switched. The on/off status of the clutch remains the status before clutch control becomes invalid. Clutch control does not become invalid during movement before clutch ON or before clutch OFF. Clutch control becomes invalid after movement before clutch ON or before clutch OFF. • FALSE: Clutch control is valid. • TRUE: Clutch control is invalid.
AuxForcedOff	Auxiliary axis clutch forced off command	Operation cycle (FB is starting)	BOOL	LIST_READ_ONLY	The auxiliary axis clutch is forcibly turned off. • FALSE: The clutch is controlled normally. • TRUE: The clutch is forcibly turned off. *: The output from the clutch immediately becomes "0" during the execution of clutch smoothing. *: When the slippage amount method smoothing is performed, the cumulative slippage amount is cleared to 0. *: When the setting value is changed to "FALSE" after the clutch is forcibly turned off, the normal clutch control restarts from the status in which the clutch is turned off.
MasterClutchSmoothingTimeConstant	Master axis clutch smoothing time constant change value	At start	INT	LIST_READ_ONLY	Master axis clutch smoothing time constant (AdvOutPutName.Cd.Clutch.MasterClutchSmoothingTimeConstant) is changed. • 0 to 5000 [ms] *: Smoothing is not performed if a negative value is set. *: When "0" is set, smoothing is performed with the setting value of Master axis clutch smoothing time constant (AdvOutPutName.PrConst.MasterClutchSmoothingTimeConstant).
AuxClutchSmoothingTimeConstant	Auxiliary axis clutch smoothing time constant change value	At start	INT	LIST_READ_ONLY	Auxiliary axis clutch smoothing time constant (AdvOutPutName.Cd.Clutch.AuxClutchSmoothingTimeConstant) is changed. • 0 to 5000 [ms] *: Smoothing is not performed if a negative value is set. *: When "0" is set, smoothing is performed with the setting value of Auxiliary axis clutch smoothing time constant (AdvOutPutName.PrConst.AuxClutchSmoothingTimeConstant).

ADV_SPEEDCHANGEGEAR_PARAM (Speed change gear parameter)

Variable name	Name	Import	Data type	Attribute	Description
MasterGearIn	Master axis speed change gear valid setting	At start	BOOL	LIST_WRITE_BACK	Whether to use the master axis speed change gear is set. • FALSE: The speed change gear is not used. • TRUE: The speed change gear is used.
MasterRatioNumerator	Master axis speed change ratio numerator	Operation cycle (FB is starting)	DINT	LIST_WRITE_BACK	The master axis speed change ratio numerator is set. • -2147483648 to 2147483647
MasterRatioDenominator	Master axis speed change ratio denominator	Operation cycle (FB is starting)	DWORD (UDINT)	LIST_WRITE_BACK	The master axis speed change ratio denominator is set. • 1 to 2147483647
AuxGearIn	Auxiliary axis speed change gear valid setting	At start	BOOL	LIST_WRITE_BACK	Whether to use the auxiliary axis speed change gear is set. • FALSE: The speed change gear is not used. • TRUE: The speed change gear is used.
AuxRatioNumerator	Auxiliary axis speed change ratio numerator	Operation cycle (FB is starting)	DINT	LIST_WRITE_BACK	The auxiliary axis speed change ratio numerator is set. • -2147483648 to 2147483647
AuxRatioDenominator	Auxiliary axis speed change ratio denominator	Operation cycle (FB is starting)	DWORD (UDINT)	LIST_WRITE_BACK	The auxiliary axis speed change ratio denominator is set. • 1 to 2147483647
OutGearIn	Output axis speed change gear valid setting	At start	BOOL	LIST_WRITE_BACK	Whether to use the output axis speed change gear is set. • FALSE: The speed change gear is not used. • TRUE: The speed change gear is used.
OutRatioNumerator	Output axis speed change ratio numerator	Operation cycle (FB is starting)	DINT	LIST_WRITE_BACK	The output axis speed change ratio numerator is set. • -2147483648 to 2147483647
OutRatioDenominator	Output axis speed change ratio denominator	Operation cycle (FB is starting)	DWORD (UDINT)	LIST_WRITE_BACK	The output axis speed change ratio denominator is set. • 1 to 2147483647

ADV_SPEEDCHANGEGEAR_CMD (Speed change gear control data)

Variable name	Name	Import	Data type	Attribute	Description
MasterSmoothingTimeConstant	Master axis speed change gear smoothing time constant change value	At start	INT	LIST_WRITE_BACK	The master axis speed change gear smoothing time constant is changed. • 0 to 5000 [ms] *: Smoothing is not performed if a negative value is set. *: When "0" is set, smoothing is performed with the setting value of Master axis speed change gear smoothing time constant (AdvOutPutName.PrConst.MasterSpeedChangeGearSmoothingTimeConstant).
AuxSmoothingTimeConstant	Auxiliary axis speed change gear smoothing time constant change value	At start	INT	LIST_WRITE_BACK	The auxiliary axis speed change gear smoothing time constant is changed. • 0 to 5000 [ms] *: Smoothing is not performed if a negative value is set. *: When "0" is set, smoothing is performed with the setting value of Auxiliary axis speed change gear smoothing time constant (AdvOutPutName.PrConst.AuxSpeedChangeGearSmoothingTimeConstant).
OutSmoothingTimeConstant	Output axis speed change gear smoothing time constant change value	At start	INT	LIST_WRITE_BACK	The output axis speed change gear smoothing time constant is changed. • 0 to 5000 [ms] *: Smoothing is not performed if a negative value is set. *: When "0" is set, smoothing is performed with the setting value of Output axis speed change gear smoothing time constant (AdvOutPutName.PrConst.OutSpeedChangeGearSmoothingTimeConstant).

ADV_CAM_PARAM (Cam parameter)

Variable name	Name	Import	Data type	Attribute	Description
LengthPerCycle	Length per cycle	At start/When the 0th point of cam data is passed	LREAL	LIST_WRITE_BACK	The length per cycle of the cam shaft for generating the current position per cycle is set. The unit is that of the input axis set in Master axis (Master) of MCv_AdvancedSync (Advanced synchronous control). • 1.0 to 2147483647.0
StrokeAmount	Cam stroke amount	At start/When the 0th point of cam data is passed	LREAL	LIST_WRITE_BACK	The cam stroke amount corresponding to the stroke ratio at 100% is set by output axis position units when Interpolation method specification (Interpolate) of the operation profile data is set to "1: Specifying for each section" or "2: Spline interpolation", or when the data format type is set to "rotary cutter". • -2147483648.0 to 2147483647.0
CamNo	Cam number	At start/When the 0th point of cam data is passed	WORD (UINT)	LIST_WRITE_BACK	The cam number to be used for cam control is set. • 0 to 60000
StartingPoint	Cam start point	At start/When the 0th point of cam data is passed	DWORD (UDINT)	LIST_WRITE_BACK	The start position of cam data is set by the resolution. This variable is valid only when Interpolation method specification (Interpolate) of the operation profile data is set to "1: Specifying for each section" or "2: Spline interpolation", or when the data format type is set to "rotary cutter". The initial value of Cam start point (AdvOutputName.Pr.Cam.StrokeAmount) is "0". (The cam axis is controlled from the 0th point of cam data (stroke ratio 0%).) If a value other than "0" is set, the cam control starts from the stroke ratio other than 0%. • 0 to 65535
LengthPerCycleChange	Length per cycle change setting	At start	INT (MC_LENGTH_PER_CYCLE_CHANGE) ☞ Page 1480 MC_LENGTH_PER_CYCLE_CHANGE	LIST_WRITE_BACK	This variable is set when Length per cycle (AdvOutputName.Pr.Cam.LengthPerCycle) is to be changed during synchronous control. • 0: Invalid (LengthChangeInvalid) • 1: Valid (LengthChangeValid)
PhaseCompensationAdvancedTime	Phase compensation advancement time	At start/ Operation cycle (FB is starting)	DINT	LIST_WRITE_BACK	This variable is set to advance or delay the phase of the current position per cycle during cam control. • -100000000 to 100000000 [μs]
PhaseCompensationTimeConstant	Phase compensation time constant	At start	WORD (UINT)	LIST_WRITE_BACK	The time constant at which the phase compensation amount during phase compensation is reflected with a primary delay is set. 63% of the phase compensation amount is reflected with the set time constant. • 0 to 65535 [ms]

ADV_CAM_MONI (Cam monitor)

Variable name	Name	Import	Data type	Attribute	Description
PhaseCompensationAmount	Phase compensation amount	Operation cycle (FB is starting)	LREAL	LIST_READ_ONLY	The cam axis phase compensation amount is stored in the unit that of the input axis set in Master axis (Master) of MCv_AdvancedSync (Advanced synchronous control). The phase compensation amount after smoothing processing with Phase compensation time constant (AdvOutputName.Pr.Cam.PhaseCompensationTimeConstant) is stored.
MasterCompositeGearSetPosition	Current position after master axis composite gear	Operation cycle (FB is starting)	LREAL	LIST_READ_ONLY	The cumulative value that indicates the current position after the master axis and sub-input axis are combined using the master axis composite gear in stored. The value in the unit that of the input axis set as Master axis (Master) of MCv_AdvancedSync (Advanced synchronous control) is stored.
MasterGearPositionPerCycle	Current position per cycle after master axis gear	Operation cycle (FB is starting)	LREAL	LIST_READ_ONLY	The input movement amount after the master axis gear is stored in the range from 0.0 to (Length per cycle - 0.00001). The unit is that of the input axis set in Master axis (Master) of MCv_AdvancedSync (Advanced synchronous control).
AuxGearPositionPerCycle	Current position per cycle after auxiliary axis gear	Operation cycle (FB is starting)	LREAL	LIST_READ_ONLY	The input movement amount after the auxiliary axis gear is stored in the range from 0.0 to (Length per cycle - 0.00001). The unit is that of the input axis set in Master axis (Master) of MCv_AdvancedSync (Advanced synchronous control).
PositionPerCycle	Current position per cycle	Operation cycle (FB is starting)	LREAL	LIST_READ_ONLY	The current position per cycle is stored in the range from 0.0 to (Length per cycle - 0.00001). The current position after phase compensation processing can be monitored. The unit is that of the input axis set in Master axis (Master) of MCv_AdvancedSync (Advanced synchronous control).
ReferenceSetPosition	Cam reference position	Operation cycle (FB is starting)	LREAL	LIST_READ_ONLY	The set position to be the cam reference position of the cam operation is stored. The set position unit is that of the axis set to Axis information (AdvOutputName.Axis) on the output axis.
SetPosition	Cam set position	Operation cycle (FB is starting)	LREAL	LIST_READ_ONLY	The cam axis set position is stored. The set position unit is that of the axis set to Axis information (AdvOutputName.Axis) on the output axis.
CamNo	Execution profile number	Operation cycle (FB is starting)	WORD (UINT)	LIST_READ_ONLY	The cam number currently being executed is stored. When Cam number (AdvOutputName.Pr.Cam.CamNo) is changed during synchronous control, the value is updated when the cam number actually being controlled is switched. The same value is also stored in Execution profile ID number (AxisName.Md.ProfileID).
StrokeAmount	Execution cam stroke amount	Operation cycle (FB is starting)	LREAL	LIST_READ_ONLY	The cam stroke amount currently being executed is stored. When Cam stroke amount (AdvOutputName.Pr.Cam.StrokeAmount) is changed during synchronous control, the value is updated when the cam stroke amount actually being controlled is switched.
LengthPerCycle	Length per execution cycle	Operation cycle (FB is starting)	LREAL	LIST_READ_ONLY	The length per cycle of the cam currently being executed is stored. When Length per cycle (AdvOutputName.Pr.Cam.LengthPerCycle) is changed during synchronous control, the value is updated when the length per cycle actually being controlled is switched.
SyncControlChangeStatus	Synchronous control change status	Operation cycle (FB is starting)	INT	LIST_READ_ONLY	The status where the synchronous control change is applied upon Synchronous control change request (AdvOutputName.Cd.Cam.SyncControlChangeRequest) is stored. • -1: Synchronous control change failed • 0: No synchronous control change request • 1: Synchronous control change in progress • 2: Synchronous control change completed
StartingPoint	Execution cam start point	Operation cycle (FB is starting)	DWORD (UDINT)	LIST_READ_ONLY	The cam start point currently being executed is stored. When Cam start point (AdvOutputName.Pr.Cam.StartingPoint) is changed during synchronous control, the value is updated when the cam is switched.

ADV_CAM_CMD (Cam control data)

Variable name	Name	Import	Data type	Attribute	Description
SyncControlChangeRequest	Synchronous control change request	Operation cycle (FB is starting)	BOOL	LIST_READ_ONLY	A synchronous control change request is made. When TRUE (Synchronous control change request) is set, Synchronous control change instruction (AdvOutpuName.Cd.Cam.SyncControlChangeCommand) is executed. • FALSE: No synchronous control change request • TRUE: Synchronous control change request
SyncControlChangeCommand	Synchronous control change instruction	Operation cycle (FB is starting)	INT (MC_SYNC_CHANGE_COMMAND) ☞ Page 1480 MC_SYNC_CHANGE_COMMAND	LIST_READ_ONLY	The synchronous control change instruction is set. • 0: Cam reference position movement (ReferenceSetPositionMovement) • 1: Current position change per cycle (ChangeCurrentPositionPerCycle) • 2: Current position change per cycle after master axis gear (ChangeMasterGearPositionPerCycle) • 3: Current position change per cycle after auxiliary axis gear (ChangeAuxGearPositionPerCycle) • 4: Current position movement per cycle (PositionPerCycleMovement)
SyncControlChange	Synchronous control change value	Operation cycle (FB is starting)	LREAL	LIST_READ_ONLY	The synchronous control change processing change value is set. • -2147483648.0 to 2147483647.0
SyncControlReflectionTime	Synchronous control change reflection time	Operation cycle (FB is starting)	WORD (UINT)	LIST_READ_ONLY	The synchronous control change processing reflection time is set. • 0 to 65535 [ms]
OutSmoothingTimeConstant	Output axis smoothing time constant change value	At start	INT	LIST_WRITE_BACK	The output axis smoothing time constant is changed. • 0 to 5000 [ms] *: Smoothing is not performed if a negative value is set. *: When "0" is set, smoothing is performed with the setting value of Output axis smoothing time constant (AdvOutputName.PrConst.SmoothingTimeConstant).

ADV_RESTORE_PARAM (Synchronous control initial position parameter)

Variable name	Name	Import	Data type	Attribute	Description
MasterGearPositionPerCycleMethod	Setting method for current position per cycle after master axis gear	At start	INT (MC_GEAR_RESTORE_METHOD) ☞ Page 1481 MC_GEAR_RESTORE_METHOD	LIST_WRITE_BACK	The setting method for Current position per cycle after master axis gear (AdvOutputName.Md.Gear.MasterGearPositionPerCycle) is set at start of MCv_AdvancedSync (Advanced synchronous control). • 0: Last value (PreviousPosition) • 1: Default setting value for current position per cycle after master axis gear (InitialGearPositionPerCycle) • 2: Calculated from input axis (CalculateFromInputAxis)
MasterGearInitialPositionPerCycle	Current position default value per cycle after master axis gear	At start	LREAL	LIST_WRITE_BACK	The initial setting value of the current position per cycle after master axis gear is set when Setting method for current position per cycle after master axis gear (AdvOutputName.Md.Gear.MasterGearPositionPerCycleMethod) is set to "1: Default setting value for current position per cycle after master axis gear (InitialGearPositionPerCycle)". The setting range is "0.0 to (Length per cycle - 0.00001)". • 0.0 to 2147483647.0
AuxGearPositionPerCycleMethod	Setting method for current position per cycle after auxiliary axis gear	At start	INT (MC_GEAR_RESTORE_METHOD) ☞ Page 1481 MC_GEAR_RESTORE_METHOD	LIST_WRITE_BACK	The setting method for Current position per cycle after auxiliary axis gear (AdvOutputName.Md.Gear.AuxGearPositionPerCycle) is set at the start of MCv_AdvancedSync (Advanced synchronous control). • 0: Last value (PreviousPosition) • 1: Default setting value for current position per cycle after auxiliary axis gear (InitialGearPositionPerCycle) • 2: Calculated from input axis (CalculateFromInputAxis)
AuxGearInitialPositionPerCycle	Current position default value per cycle after auxiliary axis gear	At start	LREAL	LIST_WRITE_BACK	The initial setting value of the current position per cycle after auxiliary axis gear is set when Setting method for current position per cycle after auxiliary axis gear (AdvOutputName.Md.Gear.AuxGearPositionPerCycle) is set to "1: Default setting value for current position per cycle after auxiliary axis gear (InitialGearPositionPerCycle)". The setting range is "0.0 to (Length per cycle - 0.00001)". • 0.0 to 2147483647.0
PositionRestorationObject	Cam axis position restoration target	At start	INT (MC_CAM_RESTORE_METHOD) ☞ Page 1481 MC_CAM_RESTORE_METHOD	LIST_WRITE_BACK	The target to be restored at start of synchronous control is set. • 0: Current position restoration per cycle (RestorePositionPerCycle) • 1: Cam reference position restoration (RestoreReferenceSetPosition) • 2: Cam set position restoration (RestoreCamSetPosition)
ReferenceSetPositionMethod	Cam reference position setting method	At start	INT (MC_CAM_REFERENCE_METHOD) ☞ Page 1481 MC_CAM_REFERENCE_METHOD	LIST_WRITE_BACK	The setting method for the cam reference position to be used for restoration is set when Cam axis position restoration target (AdvOutputName.Pr.Restore.PositionRestorationObject) is set to "0: Current position restoration per cycle (RestorePositionPerCycle)" or "2: Cam set position restoration (RestoreCamSetPosition)". • 0: Last value (PreviousPosition) • 1: Cam reference position default value (InitialReferenceSetPosition) • 2: Cam set position (CamSetPosition)
InitialReferenceSetPosition	Cam reference position default value	At start	LREAL	LIST_WRITE_BACK	The initial setting value of the cam reference position is set in the unit that of the axis set to Axis information (AdvOutputName.Axis) on the output axis when Cam reference position setting method (AdvOutputName.Pr.Restore.PositionRestorationMethod) is set to "1: Cam reference position default value (InitialReferenceSetPosition)". • -10000000000.0 to 10000000000.0

Variable name	Name	Import	Data type	Attribute	Description
PositionPerCycleMethod	Setting method for current position per cycle	At start	INT (MC_CAM_CYCLE_METHOD) ☞ Page 1481 MC_CAM_CYCLE_METHOD	LIST_WRITE_BACK	The setting method for the current position per cycle to be used for restoration is set when Cam axis position restoration target (AdvOutputName.Pr.Restore.PositionRestorationObject) is set to "1: Cam reference position restoration (RestorePositionPerCycle)" or "2: Cam set position restoration (RestoreCamSetPosition)". • 0: Last value (PreviousPosition) • 1: Current position default value per cycle (InitialPositionPerCycle) • 2: Current position per cycle after master axis gear (MasterGearPositionPerCycle) • 3: Current position per cycle after auxiliary axis gear (AuxGearPositionPerCycle)
InitialPositionPerCycle	Current position default value per cycle	At start	LREAL	LIST_WRITE_BACK	A value is set according to the Cam axis position restoration target (AdvOutputName.Pr.Restore.PositionRestorationObject) setting. The unit is that of the input axis set to Master axis (Master) of MCv_AdvancedSync (Advanced synchronous control). The setting range is "0.0 to (Length per cycle - 0.00001)". • 0.0 to 2147483647.0
RestorationAllowablePosition	Cam set position restoration allowable movement amount setting value	At start	DWORD (UDINT)	LIST_WRITE_BACK	The allowable value of the difference between the restored cam set position and the set position is set in the unit of the set position value of the output axis when Cam axis position restoration target (AdvOutputName.Pr.Restore.PositionRestorationObject) is set to "2: Cam set position restoration". If the setting value is too large, abrupt movements may occur when synchronization starts. • 0.0 to 2147483647.0 [pulse]

40.2 List of Enumerators

This section lists enumerators used for various parameters, monitor data, and motion control FBs.

ENUM enumerators

The enumeration type constant used for various parameters, monitor data, and motion control FBs actually uses INT type values.

"Enumeration type name__enumerator" INT type global labels are available on the engineering tool.

Ex.

When using MC_BUFFER_MODE type enumerator "mcBuffered" on the engineering tool

· When using an INT type global label: Set "MC_BUFFER_MODE__mcBuffered".

MC_BUFFER_MODE

Enumerator	Setting value	Description
mcAborting	0	Aborting
mcBuffered	1	Buffered
mcBlendingLow	2	BlendingLow
mcBlendingPrevious	3	BlendingPrevious
mcBlendingNext	4	BlendingNext
mcBlendingHigh	5	BlendingHigh

MC_EXECUTION_MODE

Enumerator	Setting value	Description
mcImmediately	0	Execute immediately
mcQueued	1	Execute at completion
mcSpeculatively	3	Execute speculatively

MC_COMBINE_MODE

Enumerator	Setting value	Description
mcAddAxes	0	Add positions of 2 input axes
mcSubAxes	1	Subtract positions of 2 input axes

MC_DIRECTION

Enumerator	Setting value	Description
mcPositiveDirection	1	Positive direction
mcNegativeDirection	2	Negative direction
mcShortestWay	3	Shortest path
mcCurrentDirection	4	Current direction

MC_SOURCE

Enumerator	Setting value	Description
mcSetValue	1	Set value
mcActualValue	2	Actual value
mcLatestSetValue	101	Latest set value
mcLatestActualValue	102	Latest actual value

MC_CIRC_MODE

Enumerator	Setting value	Description
mcBorder	0	Border point specification
mcCenter	1	Center point specification
mcRadius	2	Radius specification

MC_CIRC_PATHCHOICE

Enumerator	Setting value	Description
mcCW	0	CW
mcCCW	1	CCW
mcShortWay	2	Shortcut
mcLongWay	3	Detour
mcCWLongWay	4	CW detour
mcCCWLongWay	5	CCW detour

MC_START_MODE

Enumerator	Setting value	Description
mcImmediate	0	Immediately
mcAbsolute	1	Absolute
mcRelative	2	Relative
mcRampIn	3	Ramp-in

MC_AXIS_STATUS

Enumerator	Setting value	Description
Invalid	-1	Axis variable not initialized/axis parameter error
Disabled	0	Axis disabled
ErrorStop	1	Stopping on error
Stopping	2	Decelerating to stop
Homing	3	During home position return
Standstill	4	Standby
DiscreteMotion	5	During positioning operation
ContinuousMotion	6	During continuous operation
SynchronizedMotion	7	During synchronous operation

MC_AXES_GROUP_STATUS

Enumerator	Setting value	Description
Invalid	-1	Axes group variable not initialized/axes group parameter error
GroupDisabled	0	Axes group disabled
GroupErrorStop	1	Stopping on error
GroupStopping	2	Decelerating to stop
GroupStandby	4	Standby
GroupMoving	5	Operating

MC_DRIVE_MODE

Enumerator	Setting value	Description
NoModeChange	0	Do not change
hm	6	hm (Homing mode)
csp	8	csp (Cyclic synchronous position mode)
csv	9	csv (Cyclic synchronous velocity mode)
cst	10	cst (Cyclic synchronous torque mode)
ct	-104	ct (continuous operation to torque control mode)
cspr	-111	cspr (Cyclic synchronous pressure mode)

MC_VELOCITY_LIMIT_MODE

Enumerator	Setting value	Description
Ignore	0	Ignore
Truncate	2	Truncate
ImmediateStop	3	Immediate stop
ClampWithoutRamp	4	Clamp (Without ramp at deceleration)

MC_INTERPOLATE_SPEED_MODE

Enumerator	Setting value	Description
VectorSpeed	0	Vector speed
LongAxisSpeed	1	Long axis speed
ReferenceAxisSpeed	2	Reference axis speed

MC_RECORD_MODE

Enumerator	Setting value	Description
OneShot	0	One shot mode
RecordCount	1	Frequency specified mode
RingBuffer	2	Ring buffer mode

MC_CAM_CURVE_TYPE

Enumerator	Setting value	Description
ConstantSpeed	0	Constant speed
ConstantAcceleration	1	Constant acceleration
DistortedTrapezoid	2	Distorted trapezoid
DistortedSine	3	Distorted sine
DistortedConstantSpeed	4	Distorted constant speed
Cycloid	5	Cycloid
FifthCurve	6	5th curve
Trapecloid	7	Trapecloid
ReverseTrapecloid	8	Reverse trapecloid
DoubleHypotenuse	9	Double hypotenuse
ReverseDoubleHypotenuse	10	Reverse double hypotenuse
SingleHypotenuse	11	Single hypotenuse
FifthCurve_SpeedDesignation	12	5th curve (Adjustment)

MC_AXIS_TYPE

Enumerator	Setting value	Description
DriveAxis	0	Real drive axis
EncoderAxis	2	Real encoder axis
VirtualDriveAxis	3	Virtual drive axis
VirtualEncoderAxis	4	Virtual encoder axis
VirtualLinkAxis	5	Virtual link axis

MC_ENCODER_AXIS_TYPE

Enumerator	Setting value	Description
Drive	1	Via drive unit

MC_ABS_SYSTEM

Enumerator	Setting value	Description
ABSDisabled	0	Disable absolute position system
Enabled	1	Enable absolute position system
Auto	-1	Automatic setting (Acquire from connected device)

MC_DRIVE_STATE

Enumerator	Setting value	Description
NotReadyToSwitchOn	0	Not Ready To Switch On
Fault	1	Fault
FaultReactionActive	2	Fault Reaction Active
SwitchOnDisabled	3	Switch On Disabled
ReadyToSwitchOn	4	Ready To Switch On
SwitchedOn	5	Switched On
OperationEnable	6	Operation Enable
QuickStopActive	7	Quick Stop Active
Invalid	-1	Invalid

MC_DECEL_STOP_MODE

Enumerator	Setting value	Description
Ignore	0	Ignore
ImmediateStop	1	Immediate stop
KeepCurrentAcc	2	Keep current acceleration/deceleration
AlternativeAcc	3	Alternative acceleration/deceleration
ServoOffAfterImmediateStop	4	Servo OFF after immediate stop
ServoOffAfterDecelStop	5	Servo OFF after deceleration to stop

MC_STOP_CURVE_MODE

Enumerator	Setting value	Description
OverrideCurve	1	Recreate deceleration curve

MC_POS_SOURCE

Enumerator	Setting value	Description
Invalid	-1	Invalid
SetPosition	1	Set position
FeedMachinePosition	3	Feed machine position

MC_ACC_ZERO_MODE

Enumerator	Setting value	Description
ACCError	-1	Error (not started)
MaximumAcceleration	1	Maximum acceleration/deceleration

MC_SIGNAL_LOGIC

Enumerator	Setting value	Description
HighLevel	0	Detection at TRUE
LowLevel	1	Detection at FALSE
RisingEdge	2	Detection at FALSE → TRUE (rising edge)
FallingEdge	3	Detection at TRUE → FALSE (falling edge)
BothEdges	4	Detection at rising edge/falling edge

MC_ERROR_CLASS

Enumerator	Setting value	Description
_None	0	None
Warning	1	Warning
MinorError	2	Minor error
ModerateError	3	Moderate error
MajorError	4	Major error

MC_EXECUTION_STATE

Enumerator	Setting value	Description
Ready	0	Waiting execution request
Executing	1	Executing
Done	2	Execution completed
Error_	-1	Error occurrence

MC_POS_RESTORATION_STATUS

Enumerator	Setting value	Description
NotExecute	0	Not executed
WaitingRequest	1	Waiting restoration request
RestoredInIncSystem	2	Restoration completed by incremental system
RestoredInAbsSystemUnHomed	3	Restored in absolute position system (Home position return uncompleted)
RestoredInAbsSystem	4	Restored in absolute position system

MC_AXES_GROUP_TYPE

Enumerator	Setting value	Description
Standard	0	Standard

MC_OPERATION_PATTERN

Enumerator	Setting value	Description
PositioningComplete	0	Positioning completed
ContinuousPositioning	1	Continuous positioning
ContinuousBlendingLow	2	Continuous path (BlendingLow)
ContinuousBlendingPrevious	3	Continuous path (BlendingPrevious)
ContinuousBlendingNext	4	Continuous path (BlendingNext)
ContinuousBlendingHigh	5	Continuous path (BlendingHigh)

MC_CONTROL_METHOD

Enumerator	Setting value	Description
NOP	128	NOP
JUMP	130	JUMP
LOOP	131	LOOP
LEND	132	LEND
LinearAbsolute	257	Absolute value linear interpolation
LinearRelative	258	Relative value linear interpolation
CircularAbsolute	259	Absolute value circular interpolation
CircularRelative	260	Relative value circular interpolation
HelicalAbsolute	261	Absolute value helical interpolation
HelicalRelative	262	Relative value helical interpolation

MC_MCODE_OUTPUT_OVERRIDE

Enumerator	Setting value	Description
UseFbOptions	0	Use FB options
WithMode	1	WITH mode
AfterMode	2	AFTER mode

MC_STEP_MODE

Enumerator	Setting value	Description
Ignored	0	Ignore
StepPerDec	1	Deceleration unit step
StepPerPositioningData	2	Positioning data unit step

MC_POS_RESTORATION_BASE

Enumerator	Setting value	Description
BackupPosition	0	Backup position
HomePosition	1	Home position
Auto	-1	Automatic determination

MC_POS_RESTORATION_REQUEST

Enumerator	Setting value	Description
NoRequest	0	No operation
RestorationRequest	1	Current position restoration execution
RestorationRequest_EraseAbsPos	2	Current position restoration execution in absolute position erased status

MC_PRESSURE_PROCESS

Enumerator	Setting value	Description
Feed	0	Feed
Dwell	1	Dwell
Release	2	Pressure release

MC_INPUT_DIRECTION

Enumerator	Setting value	Description
NoDirectionRestriction	0	No moving direction restriction
mcPositiveDirection	1	Allowed only when the current position is in increasing direction
mcNegativeDirection	2	Allowed only when the current position is in decreasing direction

MC_CLUTCH_METHOD

Enumerator	Setting value	Description
NoClutch	0	Invalid
ClutchCommand	1	Clutch command
ClutchLeadingEdge	2	Clutch command rising edge
ClutchTrailingEdge	3	Clutch command falling edge
ClutchAddress	4	Address mode
ClutchSignal	15	I/O data specification

MC_CLUTCH_REFERENCE

Enumerator	Setting value	Description
GearFrontPosition	0	Current position before gear
GearPositionPerCycle	1	Current position per cycle after gear

MC_CLUTCH_SMOOTHING_METHOD

Enumerator	Setting value	Description
ClutchSmoothingDisabled	0	Direct
TimeConstantExponent	1	Time constant method (exponent)
TimeConstantLinear	2	Time constant method (linear)
SlippageExponent	3	Slippage amount method (exponent)
SlippageLinear	4	Slippage amount method (linear)
SlippageLinearFollow	5	Slippage amount method (linear: input amount follow-up)

MC_LENGTH_PER_CYCLE_CHANGE

Enumerator	Setting value	Description
LengthChangeInvalid	0	Invalid
LengthChangeValid	1	Valid

MC_SYNC_CHANGE_COMMAND

Enumerator	Setting value	Description
ReferenceSetPositionMovement	0	Cam reference position movement
ChangeCurrentPositionPerCycle	1	Current position change per cycle
ChangeMasterGearPositionPerCycle	2	Current position change per cycle after master axis gear
ChangeAuxGearPositionPerCycle	3	Current position change per cycle after auxiliary axis gear
PositionPerCycleMovement	4	Current position movement per cycle

MC_GEAR_RESTORE_METHOD

Enumerator	Setting value	Description
PreviousPosition	0	Last value
InitialGearPositionPerCycle	1	Current position default value per cycle after gear
CalculateFromInputAxis	2	Calculated from input axis

MC_CAM_RESTORE_METHOD

Enumerator	Setting value	Description
RestorePositionPerCycle	0	Current position restoration per cycle
RestoreReferenceSetPosition	1	Cam reference position restoration
RestoreCamSetPosition	2	Cam set position restoration

MC_CAM_REFERENCE_METHOD

Enumerator	Setting value	Description
PreviousPosition	0	Last value
InitialReferenceSetPosition	1	Cam reference position default value
CamSetPosition	2	Set position

MC_CAM_CYCLE_METHOD

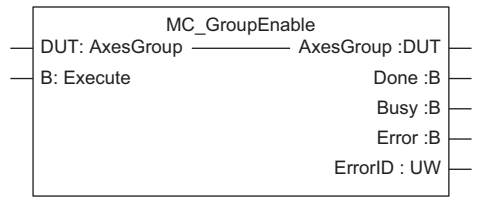
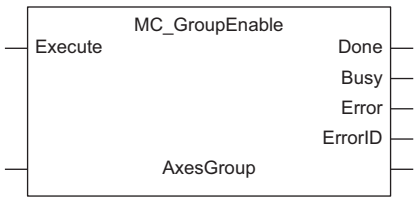
Enumerator	Setting value	Description
PreviousPosition	0	Last value
InitialPositionPerCycle	1	Current position default value per cycle
MasterGearPositionPerCycle	2	Current position per cycle after master axis gear
AuxGearPositionPerCycle	3	Current position per cycle after auxiliary axis gear

41 ADMINISTRATIVE FUNCTION BLOCKS

41.1 Axes Group Enabled

MC_GroupEnable

This FB changes the specified Axes group status (AxesGroupName.Md.GroupStatus) from "0: Axes group disabled (GroupDisabled)" to "4: Standby (GroupStandby)" and enables the axes group.

Ladder	FBD/LD	ST
		MC_GroupEnable(AxesGroup:= ?AXES_GROUP_REF? , Execute:= ?BOOL? , Done=> ?BOOL? , Busy=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
AxesGroup	Axes group information	AXES_GROUP_REF	At start	—	Omission not possible	An axes group is set. For the variable (AxesGroupName.AxesGroupRef.) used, refer to the following. ☞ Page 1423 AxesGroupName.AxesGroupRef. (Axes group information)

■Input variable

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_GroupEnable (Axes Group Enabled) is executed.

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the axes group status transits to "4: Standby (GroupStandby)".
Busy	Executing	BOOL	FALSE	This variable becomes TRUE while MC_GroupEnable (Axes Group Enabled) is being executed. This variable becomes FALSE when Axes Group Status (AxesGroupName.Md.GroupStatus) transits to "4: Standby (GroupStandby)".
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

- This FB changes Axes group status (AxesGroupName.Md.GroupStatus) of the AXES_GROUP_REF structure from "0: Axes group disabled (GroupDisabled)" to "4: Standby (GroupStandby)".
- MC_GroupEnable (Axes Group Enabled) is executed when Execution command (Execute) becomes TRUE, and Executing (Busy) becomes TRUE when the processing is started normally.
- When the processing is completed successfully, Execution completion (Done) becomes TRUE and Executing (Busy) becomes FALSE.
- When an error occurs in MC_GroupEnable (Axes Group Enabled), Error (Error) becomes TRUE and the error code is stored in Error code (ErrorID). For details on error codes, refer to the following.

📖 User's manual for the controller used

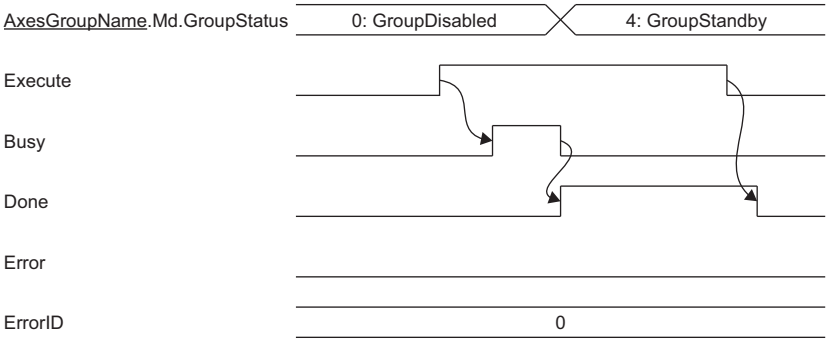
- For the axes group state transition, refer to the following.

📖 User's manual for the controller used

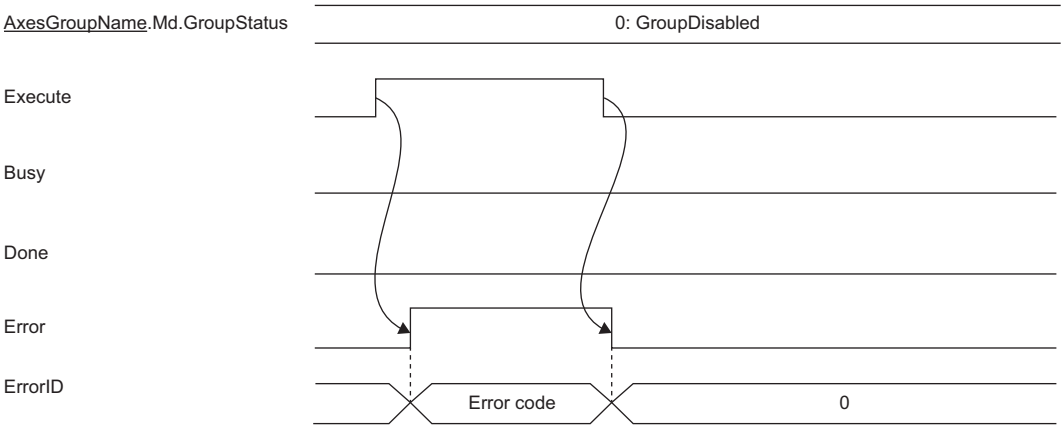
- MC_GroupEnable (Axes Group Enabled) can be executed only when Axis Status (AxisName.Md.AxisStatus) of all structuring axes is "4: Standby (Standstill)" or "0: Axis disabled (Disabled)".
- Using axes group (AxisName.Md.UseInGroup) becomes TRUE when Axes group status (AxesGroupName.Md.GroupStatus) changes to "4: Standby (GroupStandby)".
- When MC_GroupEnable (Axes Group Enabled) is executed while another axes group including a structuring axis for which Using axes group (AxisName.Md.UseInGroup) is TRUE is being set, Axes Group Configuration Axis Is in Use (error code: 1BB8H) occurs.
- If the error occurs in any of the structuring axes, Axes group status (AxesGroupName.Md.GroupStatus) changes to "1: Stopping on error (GroupErrorStop)".
- When MC_GroupEnable (Axes Group Enabled) is executed for an axes group that has already been enabled, Execution completion (Done) becomes TRUE and the execution ends.

■Timing chart

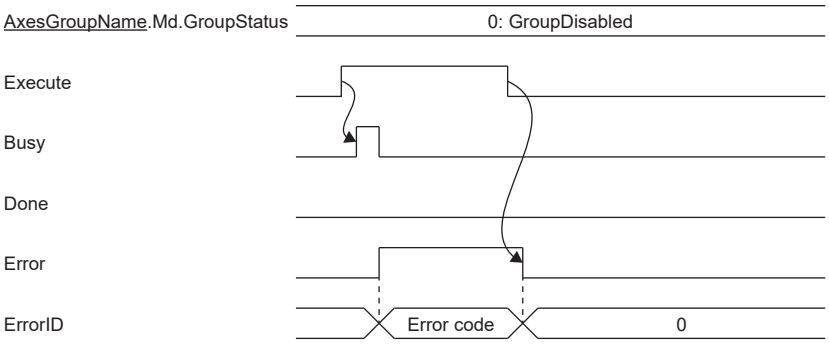
• When the processing is completed successfully



• When I/O variables are abnormal



• When the processing is completed with an error



Precautions

- Axes group No. (GroupNo) of the AXES_GROUP_REF structure must be set in advance before MC_GroupEnable (Axes Group Enabled) is executed.
- MC_GroupEnable (Axes Group Enabled) can only be executed when Axis status (AxisName.Md.AxisStatus) of all structuring axes set in the axes group is "4: Standby (Standstill)" or "0: Axis disabled (Disabled)".
- If a parameter error occurs at axes group variable initialization, Out of Parameter Range (Axes Group) (error code: 1A61H) occurs. In this case, Axes group status (AxesGroupName.Md.GroupStatus) changes to "-1: Axes group variable not initialized/Axes group parameter error (Invalid)". Refreshing of the monitor data or others is not executed for axes groups whose status is "-1: Axes group variable not initialized/Axes group parameter error (Invalid)".

Program example

The following is a program example. The axes group enable command (bGroupEnable) is set to TRUE, and Axes group 1 (AxesGroup001) is enabled.

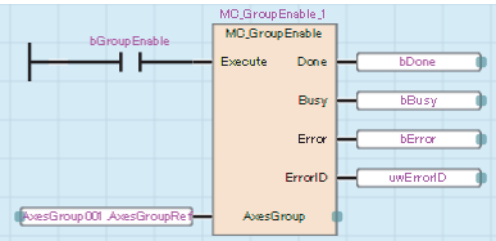
■Axes group

Axes group No.	Label name	Data type	Comment
1	AxesGroup001	AXES_GROUP_REF	Axes group 1

■Labels used

Label name	Data type	Comment
MC_GroupEnable_1	MC_GroupEnable	Axes Group Enabled FB
bGroupEnable	Bit	Axes group enable command
bDone	Bit	Execution completion
bBusy	Bit	Executing
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program



■ST program

```
MC_GroupEnable_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bGroupEnable ,
   Done=> bDone ,
   Busy=> bBusy ,
   Error=> bError ,
   ErrorID=> uwErrorID
);
```

Point

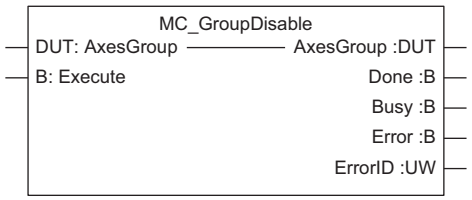
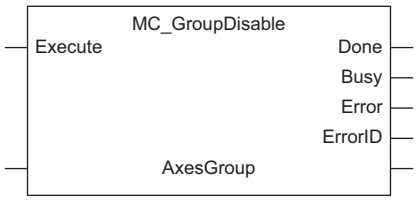
MC_GroupEnable (Axes Group Enabled) is used in combination with the motion control FB that performs interpolation control. For an example program with interpolation control, refer to the following.

- ☞ Page 1690 Absolute Value Linear Interpolation Control
- ☞ Page 1705 Relative Value Linear Interpolation Control
- ☞ Page 1719 Absolute Value Circular Interpolation Control
- ☞ Page 1736 Relative Value Circular Interpolation Control

41.2 Axes Group Disabled

MC_GroupDisable

This FB changes the specified Axes group status (AxesGroupName.Md.GroupStatus) to "0: Axes group disabled (GroupDisabled)" and disables the axes group.

Ladder	FBD/LD	ST
		MC_GroupDisable(AxesGroup:= ?AXES_GROUP_REF? , Execute:= ?BOOL? , Done=> ?BOOL? , Busy=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
AxesGroup	Axes group information	AXES_GROUP_REF	At start	—	Omission not possible	An axes group is set. For the variable (AxesGroupName.AxesGroupRef.) used, refer to the following. ☞ Page 1423 AxesGroupName.AxesGroupRef. (Axes group information)

■Input variable

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_GroupDisable (Axes Group Disabled) is executed.

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the axes group status transits to "0: Axes group disabled (GroupDisabled)".
Busy	Executing	BOOL	FALSE	This variable becomes TRUE while MC_GroupDisable (Axes Group Disabled) is being executed. This variable becomes FALSE when Axes Group Status (AxesGroupName.Md.GroupStatus) transits to "0: Axes group disabled (GroupDisabled)".
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

- This FB changes Axes group status (AxesGroupName.Md.GroupStatus) of the AXES_GROUP_REF structure to "0: Axes group disabled (GroupDisabled)".
- MC_GroupDisable (Axes Group Disabled) is executed when Execution command (Execute) becomes TRUE, and Executing (Busy) becomes TRUE when the processing is started normally.
- When the processing is completed successfully, Execution completion (Done) becomes TRUE and Executing (Busy) becomes FALSE.
- When an error occurs in MC_GroupDisable (Axes Group Disabled), Error (Error) becomes TRUE and the error code is stored in Error code (ErrorID). For details on error codes, refer to the following.

📖 User's manual for the controller used

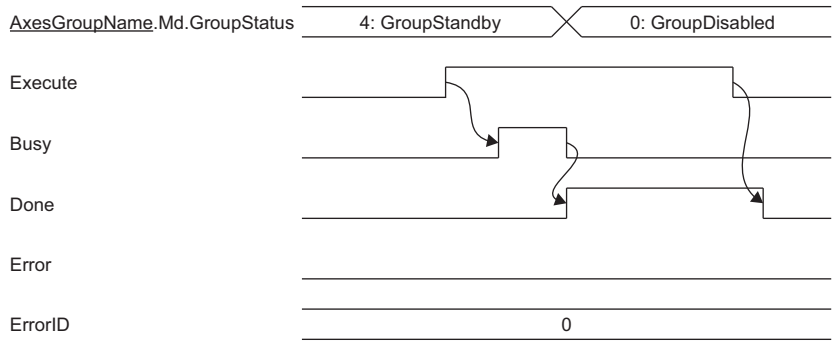
- For the axes group state transition, refer to the following.

📖 User's manual for the controller used

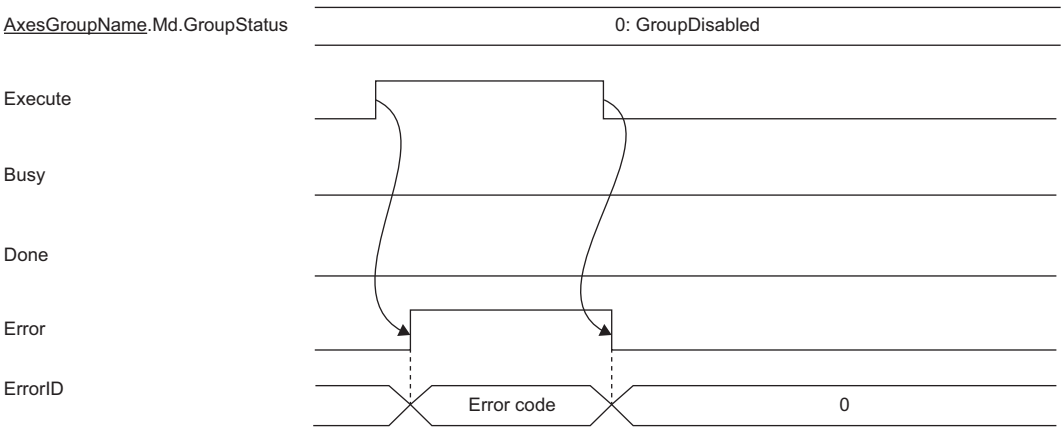
- MC_GroupDisable (Axes Group Disabled) can only be executed when Axes Group Status (AxesGroupName.Md.GroupStatus) is "4: Standby (GroupStandby)" or "1: Stopping on error (GroupErrorStop)". When executing MC_GroupDisable (Axes Group Disabled) while the axes group status is "1: Stopping on error (GroupErrorStop)" and the axes group is in operation, the axes group status transits to "0: Axes group disabled (GroupDisabled)" after the stop completes. When MC_GroupDisable (Axes Group Disabled) is executed while Axes group status (AxesGroupName.Md.GroupStatus) is other than "4: Standby (GroupStandby)" or "1: Stopping on error (GroupErrorStop)", Axes Group Status Incorrect (Axes Group Disable) (error code: 1AEFH) occurs and the FB being executed stops.
- When Axes group status (AxesGroupName.Md.GroupStatus) changes to "0: Axes group disabled (GroupDisabled)", Using axes group (AxisName.Md.UseInGroup) of the structuring axis becomes FALSE.
- When MC_GroupDisable (Axes Group Disabled) is executed for an axes group that has already been disabled, Execution completion (Done) becomes TRUE and the execution ends.

■Timing chart

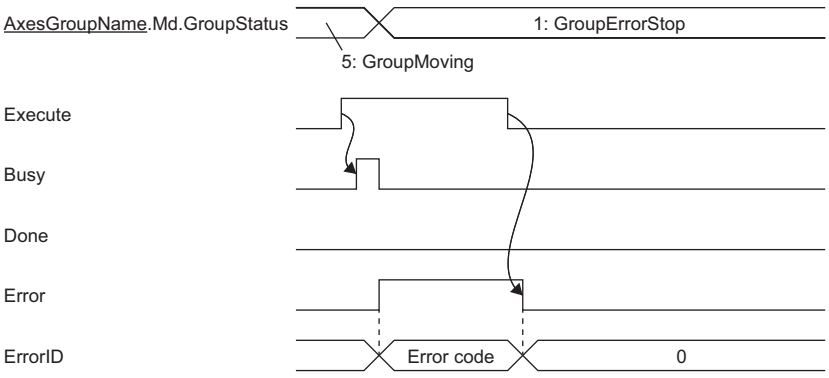
• When the processing is completed successfully



• When I/O variables are abnormal



• When the processing is completed with an error



Precautions

- Axes group No. (GroupNo) of the AXES_GROUP_REF structure must be set in advance before MC_GroupDisable (Axes Group Disabled) is executed.
- If a parameter error occurs at axes group variable initialization, Out of Parameter Range (Axes Group) (error code: 1A61H) occurs. In this case, Axes group status (AxesGroupName.Md.GroupStatus) changes to "-1: Axes group variable not initialized/Axes group parameter error (Invalid)". Refreshing of the monitor data or others is not executed for axes groups whose status is "-1: Axes group variable not initialized/Axes group parameter error (Invalid)".

Program example

The following is a program example. The axes group disable command (bGroupDisable) is set to TRUE, and Axes group 1 (AxesGroup001) is disabled.

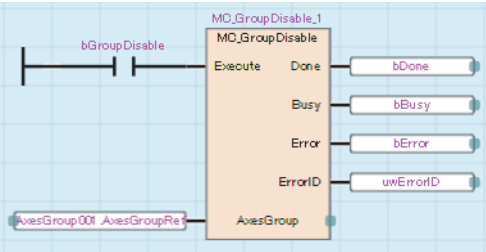
■Axes group

Axes group No.	Label name	Data type	Comment
1	AxesGroup001	AXES_GROUP_REF	Axes group 1

■Labels used

Label name	Data type	Comment
MC_GroupDisable_1	MC_GroupDisable	Axes Group Disabled FB
bGroupDisable	Bit	Axes group disable command
bDone	Bit	Execution completion
bBusy	Bit	Executing
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program



■ST program

```
MC_GroupDisable_1(
  AxesGroup:= AxesGroup001.AxesGroupRef ,
  Execute:= bGroupDisable ,
  Done=> bDone ,
  Busy=> bBusy ,
  Error=> bError ,
  ErrorID=> uwErrorID
);
```

Point

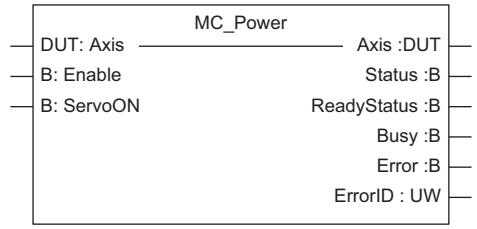
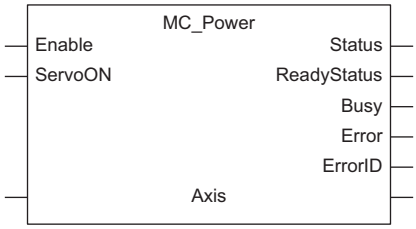
MC_GroupDisable (Axes Group Disabled) is used in combination with the motion control FB that performs interpolation control. For an example program with interpolation control, refer to the following.

- ☞ Page 1690 Absolute Value Linear Interpolation Control
- ☞ Page 1705 Relative Value Linear Interpolation Control
- ☞ Page 1719 Absolute Value Circular Interpolation Control
- ☞ Page 1736 Relative Value Circular Interpolation Control

41.3 Operable

MC_Power

This FB switches a specified axis to the operation possible status.

Ladder	FBD/LD	ST
		MC_Power(Axis:= ?AXIS_REF? , Enable:= ?BOOL? , ServoON:= ?BOOL? , Status=> ?BOOL? , ReadyStatus=> ?BOOL? , Busy=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	Always	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information) *: The axis information is established when the FB is called for the first time "PLC Ready (MotionSystem.Cd.SequenceReady)" is turned on.

■Input variables

Input variable	Name	Data type	Input	Setting range	Default value	Description
Enable	Enable	BOOL	Always	TRUE, FALSE	FALSE	When this variable is set to TRUE, control of the specified axis is enabled (operation possible status). When this variable is set to FALSE, control of the specified axis is disabled.
ServoON	Servo ON request	BOOL	Always	TRUE, FALSE	FALSE	When this variable is set to TRUE, the servo ON is requested to the specified axis.

■Output variables

Output variable	Name	Data type	Default value	Description
Status	Operable	BOOL	FALSE	This variable becomes TRUE when the axis becomes operation possible status.
ReadyStatus	Ready ON status	BOOL	FALSE	This variable becomes TRUE when Driver READY turns on.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_Power (Operation Available) is executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

- This FB initializes the information of the specified axis and switches the axis to the operation possible status.
- When the inputs for Enable (Enable) and Servo ON request (ServoON) are TRUE, the selected axis is switched to the operation possible status.
- Executing (Busy) becomes TRUE when the processing is started.
- Operable (Status) becomes TRUE when the drive unit has been switched to the operation possible status.
- When an error occurs in MC_Power (Operation Available), Error (Error) becomes TRUE and the error code is stored in Error code (ErrorID). For details on error codes, refer to the following.

User's manual for the controller used

- Axis status (AxisName.Md.AxisStatus) of the AXIS_MONI structure transits from "0: Axis disabled (Disabled)" to "4: Standby (Standstill)". For the axis state transition, refer to the following.

User's manual for the controller used

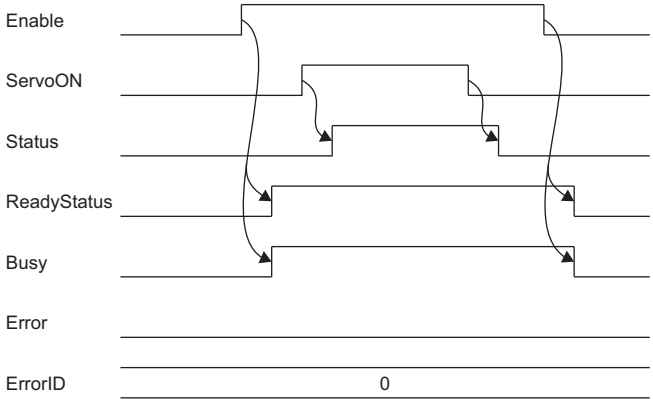
- Inputting Enable (Enable) and Servo ON request (ServoON) switches the servo ON/OFF status and the drive unit status of the axis as shown below.

Input variable		Output variable		Servo ON/OFF status	Driver status (AxisName.Md.Driver_State)
Enable (Enable)	Servo ON request (ServoON)	Ready ON status (ReadyStatus)	Operable (Status)		
TRUE	TRUE	TRUE	TRUE	Servo ON	6: Operation Enable (OperationEnable)
	FALSE	TRUE	FALSE	Servo OFF	5: Switched On (SwitchedOn)
FALSE	TRUE	FALSE	FALSE	Servo OFF	3: Switched On Disabled (SwitchOnDisabled)
	FALSE	FALSE	FALSE	Servo OFF	3: Switched On Disabled (SwitchOnDisabled)

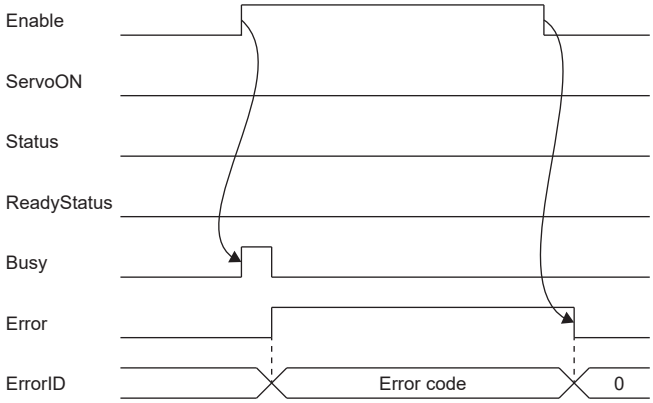
- If the real axis is rotated by external force during the servo OFF status, follow up processing is performed.
- The servo ON/OFF control can be operated regardless of the control mode. The control mode used during servo OFF depends on the specification of the drive unit.
- MC_Power (Operation Available) is sent to the drive unit while a drive unit error is occurring, so Enable (Enable) and Servo ON request (ServoON) do not need to be turned from FALSE to TRUE again.

■Timing chart

• When the processing is completed successfully



• When the processing is completed with an error



Precautions

- Axis No. (AxisNo) of the AXIS_REF structure must be set in advance before MC_Power (Operation Available) is executed.
- MC_Power (Operation Available) establishes the axis information the first time that the FB is called after "PLC Ready (MotionSystem.Cd.SequenceReady)" is switched to on. Changes made to the axis information while Enable (Enable) is FALSE are not reflected.
- Do not set two or more MC_Power (Operation Available) for one axis. The operation is not guaranteed if two or more of this FB are set.
- The servo ON/OFF status when stopping the program is maintained while Built-in motion ready (X20) is off.

Program example

The following is a program example. The servo ON request command (bServoON) is set to TRUE, and Axis 1 (Axis0001) is switched to the operation possible status.

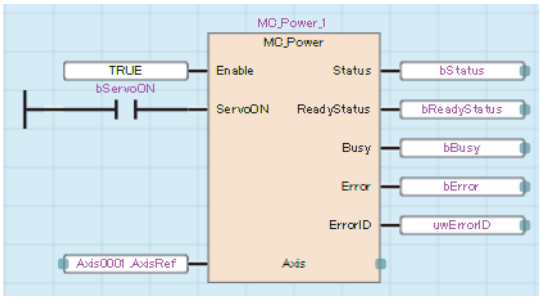
■Axis

Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

Label name	Data type	Comment
MC_Power_1	MC_Power	Operation Available FB
bServoON	Bit	Servo ON request command
bStatus	Bit	Operable
bReadyStatus	Bit	Ready ON status
bBusy	Bit	Executing
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program



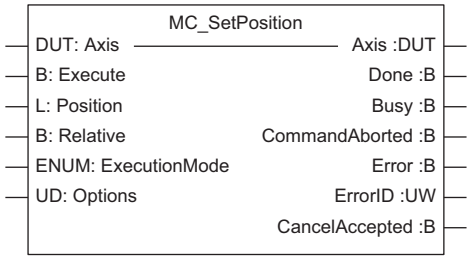
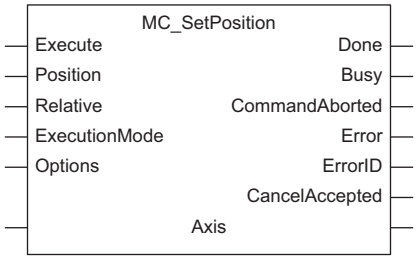
■ST program

```
MC_Power_1(
   Axis:= Axis0001.AxisRef ,
   Enable:= TRUE ,
   ServoON:= bServoON ,
   Status=> bStatus ,
   ReadyStatus=> bReadyStatus ,
   Busy=> bBusy ,
   Error=> bError ,
   ErrorID=> uwErrorID
);
```

41.4 Current Position Change

MC_SetPosition

This FB changes the current position (commanded position, actual position) of the specified axis.

Ladder	FBD/LD	ST
		MC_SetPosition(Axis:= ?AXIS_REF? , Execute:= ?BOOL? , Position:= ?LREAL? , Relative:= ?BOOL? , ExecutionMode:= ?INT? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD? , CancelAccepted=> ?BOOL?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_SetPosition (Current Position Change) is executed.
Position	Target position	LREAL	At start	-10000000000.0 to 10000000000.0	0.0	The change target position of the specified axis is set. The setting value differs depending on the setting of Relative position selection (Relative). For details, refer to the following. ☞ Page 1496 Target position (Position)
Relative	Relative position selection	BOOL	At start	TRUE, FALSE	FALSE	Whether to change the current position with the relative distance or the absolute position is set. • FALSE: Current position change with the absolute position • TRUE: Current position change with the relative distance
ExecutionMode	Execution mode	INT (MC_EXECUTION_MODE)	At start	1, 3	3	The execution method of MC_SetPosition (Current Position Change) is set. • 1: Execute at completion (mcQueued)*1 • 3: Execute speculatively (mcSpeculatively)*2 For details, refer to the following. ☞ Page 1497 Execution mode (ExecutionMode)
Options	Options	DWORD(HEX)	At start	00000000H to 00010000H	00000000H	The function options for MC_SetPosition (Current Position Change) are set in the specified bit position. For details, refer to the following. ☞ Page 1498 Options (Options)

*1 The FB is executed after the previous FB is completed.

*2 The FB can be executed only when there is no running FB on the specified axis.

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the current position change is completed.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_SetPosition (Current Position Change) is executed.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MC_SetPosition (Current Position Change) is aborted.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used
CancelAccepted	Cancel acceptance	BOOL	FALSE	This variable becomes TRUE when MC_SetPosition (Current Position Change) accepted the cancel.

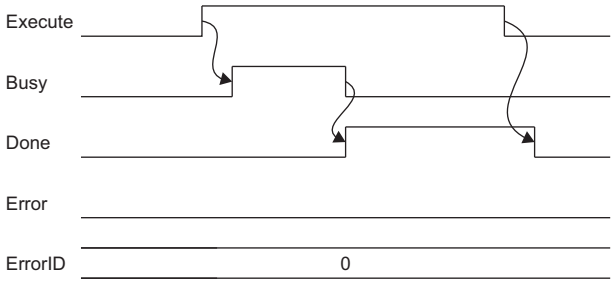
Processing details

- This FB changes the current position of the specified axis.
- When Relative position selection (Relative) is TRUE, changes the current position to a position obtained by adding the target position (relative distance) to the current position.
- When Relative position selection (Relative) is FALSE, changes current position to the target position (absolute position).
- MC_SetPosition (Current Position Change) is executed when Execution command (Execute) becomes TRUE, and Executing (Busy) becomes TRUE when the processing is started normally.
- When the processing is completed and the current position is changed, Execution completion (Done) becomes TRUE.
- When Axis status (AxisName.Md.AxisStatus) becomes "0: Axis disabled (Disabled)" or "1: Stopping on error (ErrorStop)" before the processing is completed after executing MC_SetPosition (Current Position Change) with Execution mode (ExecutionMode) set to "1: Execute at completion (mcQueued)", MC_SetPosition Instruction Error (error code: 1B9CH) occurs.
- When an error occurs in MC_SetPosition (Current Position Change), Error (Error) becomes TRUE and the error code is stored in Error code (ErrorID). For details on error codes, refer to the following.

📖 User's manual for the controller used

■Timing chart

- When the processing is completed successfully



- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Target position (Position)

The values that can be set in Target position (Position) differ depending on whether Relative position selection (Relative) is TRUE or FALSE.

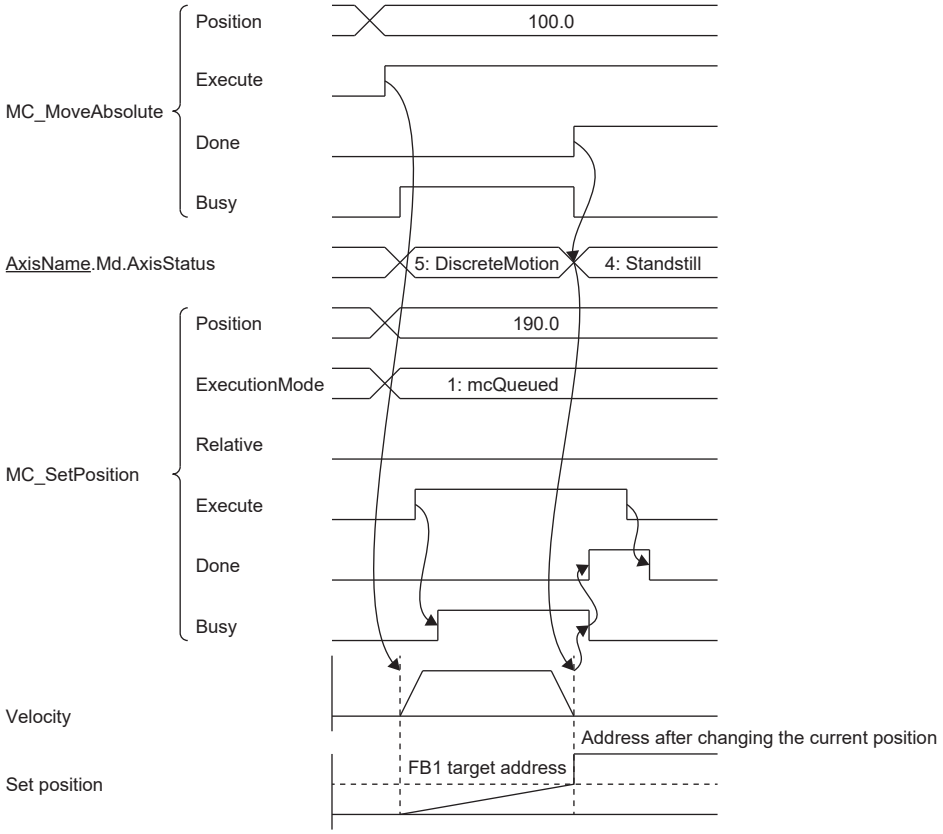
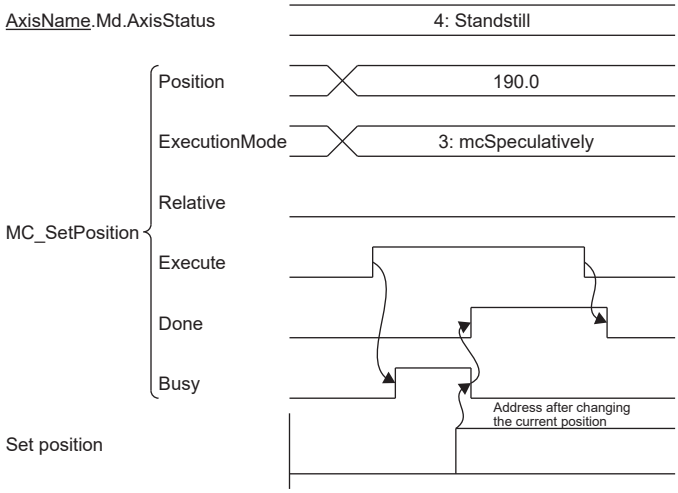
Set Target position (Position) within the following setting range.

However, even when the position is within the specified range, an error occurs and the operation is not executed if the set position is outside the software stroke limit range after the current position change.

Relative position selection (Relative)	Target position (Position) setting range
FALSE (Current position change with absolute position)	■When the ring counter is disabled -10000000000.0 ≤ Setting value < 10000000000.0 ■When the ring counter is enabled Ring counter lower limit value ≤ Setting value < Ring counter upper limit value
TRUE (Current position change with relative distance)	■When the ring counter is disabled -10000000000.0 ≤ Setting value ≤ 10000000000.0 ■When the ring counter is enabled - (Ring counter upper limit value - ring counter lower limit value) / 2 ≤ Setting value ≤ (Ring counter upper limit value - ring counter lower limit value) / 2

■Execution mode (ExecutionMode)

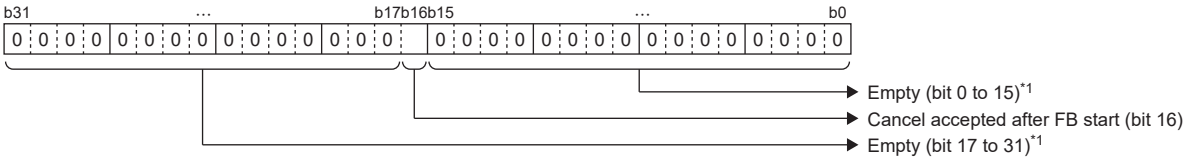
When MC_SetPosition (Current Position Change) is executed during axis operation, the timing for the change depends on the setting of Execution mode (ExecutionMode). The timing for Execution mode (ExecutionMode) is shown below.

Setting value	Description
1: Execute at completion (mcQueued)	It is executed after the FB being executed completes. At rising edge detection of Execution command (Execute), Executing (Busy) becomes TRUE and remains TRUE until Axis status (AxisName.Md.AxisStatus) changes to "4: Standby (Standstill)". Current position change operation is started when Axis status (AxisName.Md.AxisStatus) changes to "4: Standby (Standstill)". 
3: Execute speculatively (mcSpeculatively)	When Axis status (AxisName.Md.AxisStatus) of the relevant axis is "4: Standby (Standstill)" at rising edge detection of Execution command (Execute), the current position change can be executed. When Axis status (AxisName.Md.AxisStatus) of the relevant axis is not "4: Standby (Standstill)" or when an FB is being executed, MC_SetPosition Instruction Error (error code: 1B9CH) occurs and the current position change is not executed. 

■Options (Options)

The function options for MC_SetPosition (Current Position Change) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to "0". If a value other than "0" is set, an MC_SetPosition instruction error (error code: 1B9CH) occurs.

Bit	Name	Description
16	Cancel accepted after FB start	Whether or not to allow a cancel after MC_SetPosition (Current Position Change) is executed is set. • 0: Disabled • 1: Enabled

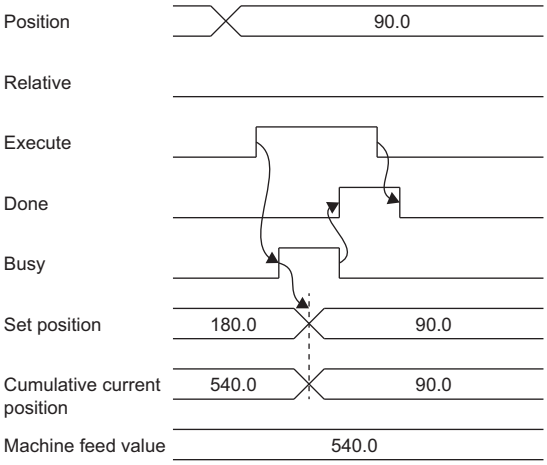
■Operation overview

- When MC_SetPosition (Current Position Change) is executed while the axis is stopped, the current position change operation starts at rising edge detection of Execution command (Execute). In the current position change, the set position and the cumulative current position are changed to the position set in Target position (Position). Target position (Position) becomes "Current position change with relative distance" when Relative position selection (Relative) is TRUE and "Current position change with absolute position" when Relative position selection (Relative) is FALSE.

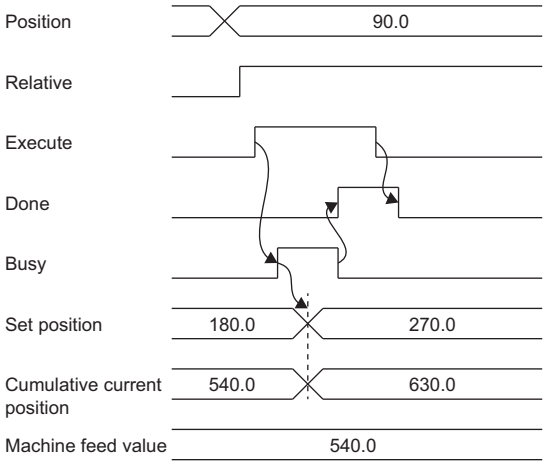
Ex.

When executing the current position change to "90.0" at a stop with a ring counter lower limit value of "0.0", a ring counter upper limit value of "360.0", and a set position of "180.0" (a cumulative current position of "540.0").

<When Relative position selection (Relative) is FALSE (absolute position)>



<When Relative position selection (Relative) is TRUE (relative distance)>



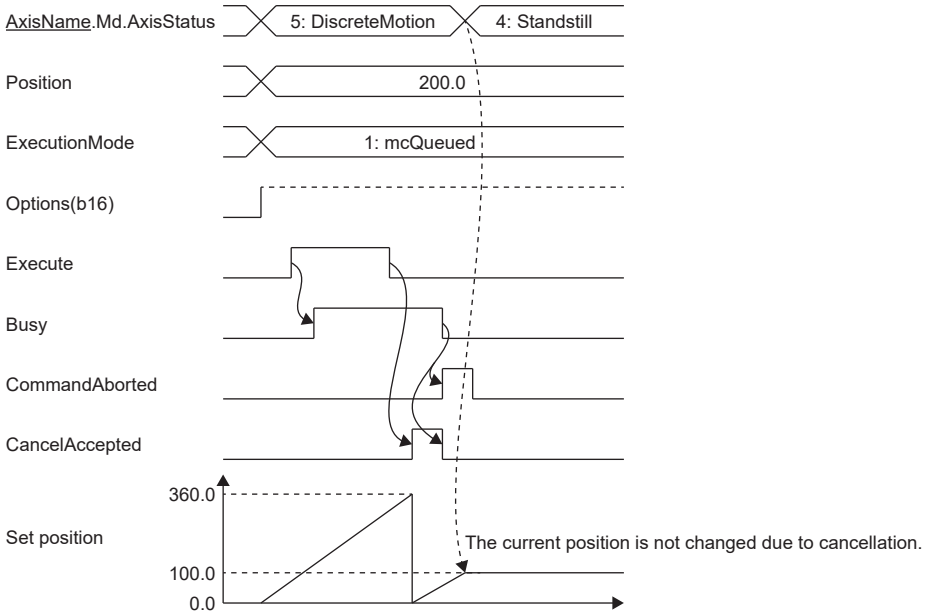
■Cancel

A current position change that is in the standby status can be canceled after MC_SetPosition (Current Position Change) execution.

- To enable cancel, execute MC_SetPosition (Current Position Change) with the status of "Cancel accepted after FB start (Options (Options): Bit 16)" set to "1: Allow".
- Cancel is started at falling edge detection of Execution command (Execute).
- Cancel is accepted only when Executing (Busy) of the output pin is TRUE.
- When MC_SetPosition (Current Position Change) accepts the cancel, Cancel acceptance (CancelAccepted) of the output pin becomes TRUE.
- When cancel is completed, Abortion of execution (CommandAborted) of the output pin becomes TRUE.
- When cancelled, the set position is not changed.

Ex.

After starting with Execution mode (ExecutionMode) set to "1: Execute at completion (mcQueued)", the change is cancelled while Axis Status (AxisName.Md.AxisStatus) is waiting to switch to "4: Standby (Standstill)"



Precautions

- Although the target position of the MC_SetPosition (Current Position Change) being executed will not change if the current position is changed during axis operation, the operation to the target value will be changed because of the current position being changed.

Program example

The following is a program example. The current position change command (bSetPosition) is set to TRUE, and the current position of Axis 1 (Axis0001) is changed to "100.0".

■Axis

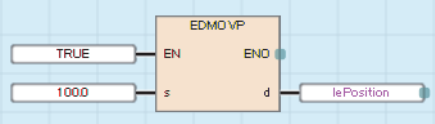
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

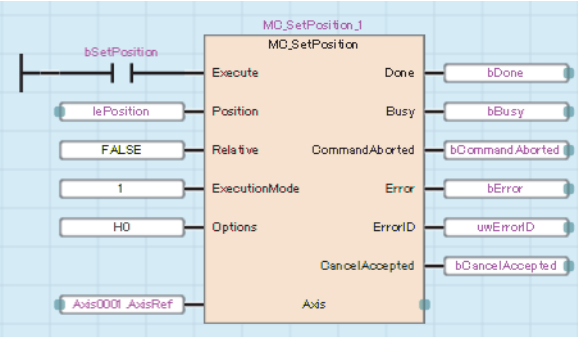
Label name	Data type	Comment
MC_SetPosition_1	MC_SetPosition	Current Position Change FB
bSetPosition	Bit	Current position change command
lePosition	Double-precision real number	Target position
bDone	Bit	Ready
bBusy	Bit	Executing
bCommandAborted	Bit	Abortion of execution
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code
bCancelAccepted	Bit	Cancel acceptance

■FBD/LD program

• Data setting for Current Position Change



• Current Position Change



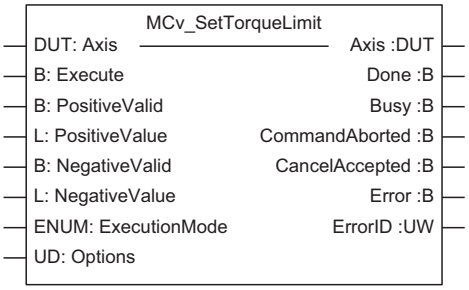
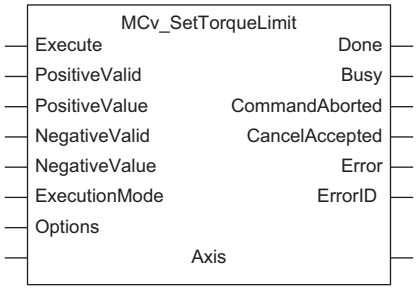
■ST program

```
//-----Data setting for Current Position Change-----
lePosition:= 100.0;

//-----Current Position Change-----
MC_SetPosition_1(
   Axis:= Axis0001.AxisRef ,
   Execute:= bSetPosition ,
   Position:= lePosition ,
   Relative:= FALSE ,
   ExecutionMode:= MC_EXECUTION_MODE__mcQueued ,
   Options:= H00000000 ,
   Done=> bDone ,
   Busy=> bBusy ,
   CommandAborted=> bCommandAborted ,
   Error=> bError ,
   ErrorID=> uwErrorID ,
   CancelAccepted=> bCancelAccepted
);
```

41.5 Torque Limit Value

MCv_SetTorqueLimit

This FB changes a torque limit value.

Ladder	FBD/LD	ST
		MCv_SetTorqueLimit(Axis:= ?AXIS_REF? , Execute:= ?BOOL? , PositiveValid:= ?BOOL? , PositiveValue:= ?LREAL? , NegativeValid:= ?BOOL? , NegativeValue:= ?LREAL? , ExecutionMode:= ?INT? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , CommandAborted=> ?BOOL? , CancelAccepted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_SetTorqueLimit (Torque Limit Value) is executed.
PositiveValid	Positive direction torque limit valid	BOOL	At start/Retrigger possible	TRUE, FALSE	FALSE	Whether to enable or disable the positive direction torque limit setting value change is set. • FALSE: Positive direction torque limit disabled • TRUE: Positive direction torque limit enabled
PositiveValue	Positive direction torque limit value	LREAL	At start/Retrigger possible	0.0 to 1000.0 [%]	0.0	The positive direction torque limit value is set. The value is imported when Positive direction torque limit valid (PositiveValid) is TRUE.
NegativeValid	Negative direction torque limit valid	BOOL	At start/Retrigger possible	TRUE, FALSE	FALSE	Whether to enable or disable the negative direction torque limit setting value change is set. • FALSE: Negative direction torque limit disabled • TRUE: Negative direction torque limit enabled
NegativeValue	Negative direction torque limit value	LREAL	At start/Retrigger possible	0.0 to 1000.0 [%]	0.0	The negative direction torque limit value is set. The value is imported when Negative direction torque limit valid (NegativeValid) is TRUE.

Input variable	Name	Data type	Input	Setting range	Default value	Description
ExecutionMode	Execution mode	INT (MC_EXECUTION_MODE)	At start	0, 1, 3	0	The execution timing of MCv_SetTorqueLimit (Torque Limit Value) is set. • 0: Execute immediately (mcImmediately) • 1: Execute at completion (mcQueued) • 3: Execute speculatively (mcSpeculatively) For details, refer to the following. ☞ Page 1505 Execution mode (ExecutionMode)
Options	Options	DWORD(HEX)	At start	00000000H to 00010000H	00000000H	The function options for MCv_SetTorqueLimit (Torque Limit Value) are set in the specified bit position. For details, refer to the following. ☞ Page 1505 Options (Options)

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the torque limit value change is completed successfully.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_SetTorqueLimit (Torque Limit Value) is executed.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MCv_SetTorqueLimit (Torque Limit Value) is aborted.
CancelAccepted	Cancel acceptance	BOOL	FALSE	This variable becomes TRUE when MCv_SetTorqueLimit (Torque Limit Value) accepts the cancel.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

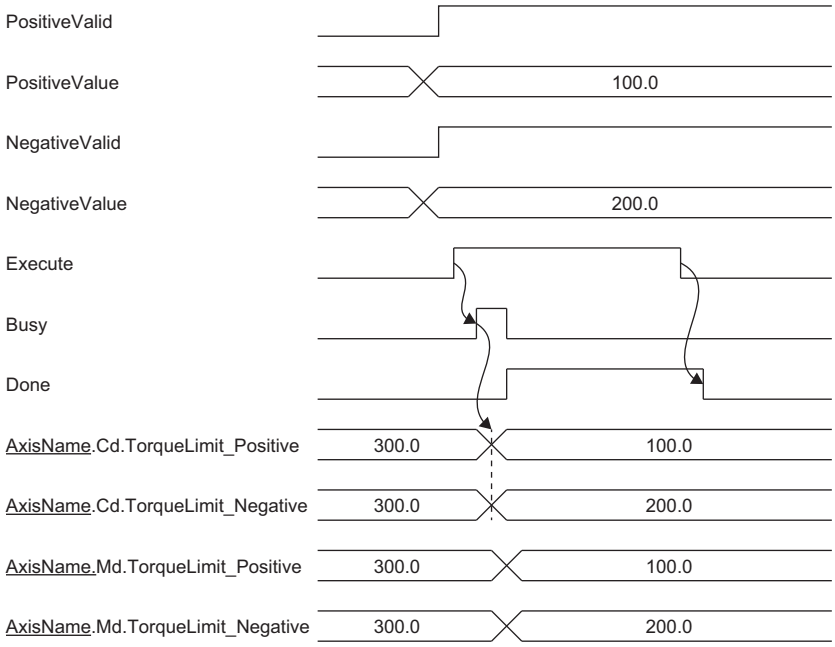
- This FB changes a positive direction torque limit value or negative direction torque limit value.
- When Positive direction torque limit valid (PositiveValid) is TRUE, a positive direction torque limit value is changed to the value of Positive direction torque limit value (PositiveValue).
- When Negative direction torque limit valid (NegativeValid) is TRUE, a negative direction torque limit value is changed to the value of Negative direction torque limit value (NegativeValue).
- MCv_SetTorqueLimit (Torque Limit Value) is executed when Execution command (Execute) is TRUE, and Executing (Busy) becomes TRUE when the processing starts normally.
- The timing for the torque limit change is set in Execution mode (ExecutionMode).
- When the processing is completed and the torque limit value is changed, Execution completion (Done) becomes TRUE.
- When an error occurs in MCv_SetTorqueLimit (Torque Limit Value), Error (Error) becomes TRUE and the error code is stored in Error code (ErrorID). For details on error codes, refer to the following.

📖 User's manual for the controller used

- When Positive direction torque limit valid (PositiveValid) and Negative direction torque limit valid (NegativeValid) are both FALSE, the torque limit change is not performed and Execution completion (Done) becomes TRUE.

■Timing chart

- When the processing is completed successfully



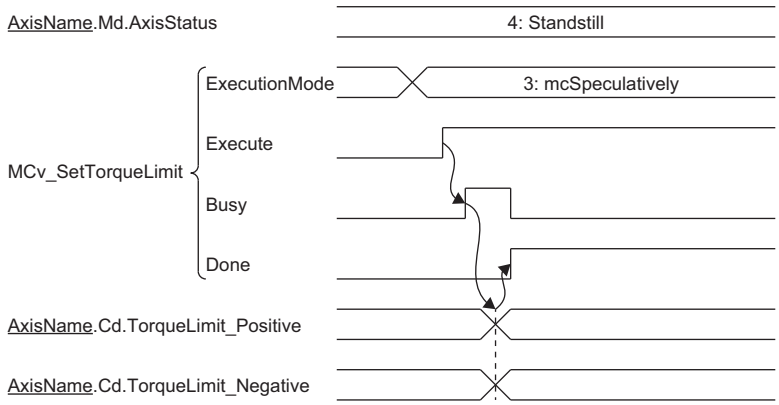
- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Execution mode (ExecutionMode)

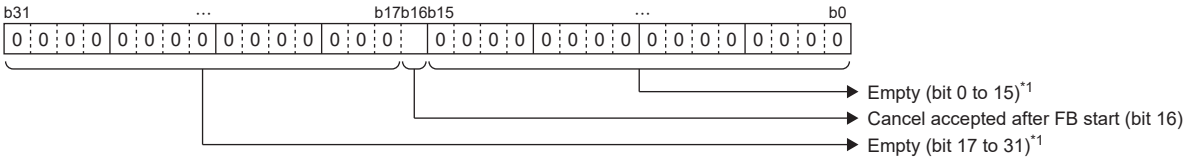
When executing MCv_SetTorqueLimit (Torque Limit Value) during axis operation, the timing of the change depends on the setting of Execution mode (ExecutionMode). The timing for Execution mode (ExecutionMode) is shown below.

Setting value	Description
0: Execute immediately (mcImmediately)	It is executed immediately regardless of Axis status (AxisName.Md.AxisStatus). Positive direction torque limit value (AxisName.Cd.TorqueLimit_Positive) and Negative direction torque limit value (AxisName.Cd.TorqueLimit_Negative) are changed at rising edge detection of Execution command (Execute).
3: Execute speculatively (mcSpeculatively)	When Axis status (AxisName.Md.AxisStatus) of the relevant axis is "4: Standby (Standstill)" at rising edge detection of Execution command (Execute), the torque limit value change is executed. When Axis status (AxisName.Md.AxisStatus) of the relevant axis is not "4: Standby (Standstill)" or when an FB is being executed, MCv_SetTorqueLimit Instruction Error (error code: 1B9EH) occurs and the torque limit value change is not executed. 

■Options (Options)

The function options for MCv_SetTorqueLimit (Torque Limit Value) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.

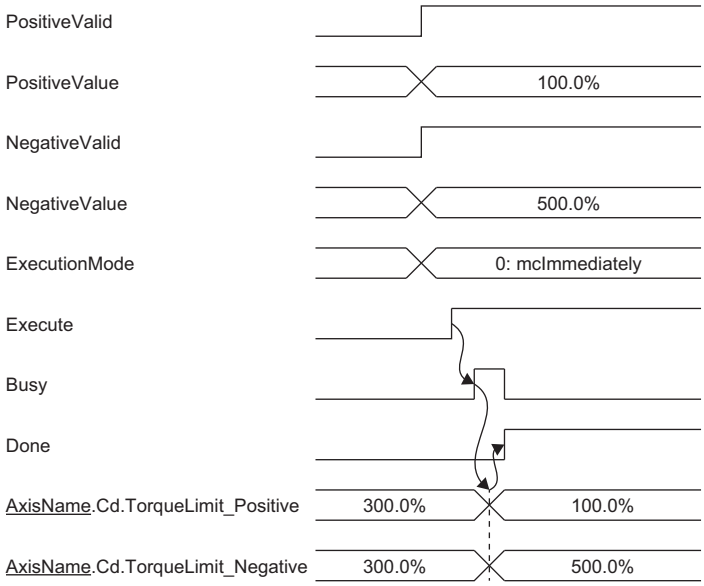


*1 Set empty areas to "0". If a value other than "0" is set, an MCv_SetTorqueLimit instruction error (error code: 1B9EH) occurs.

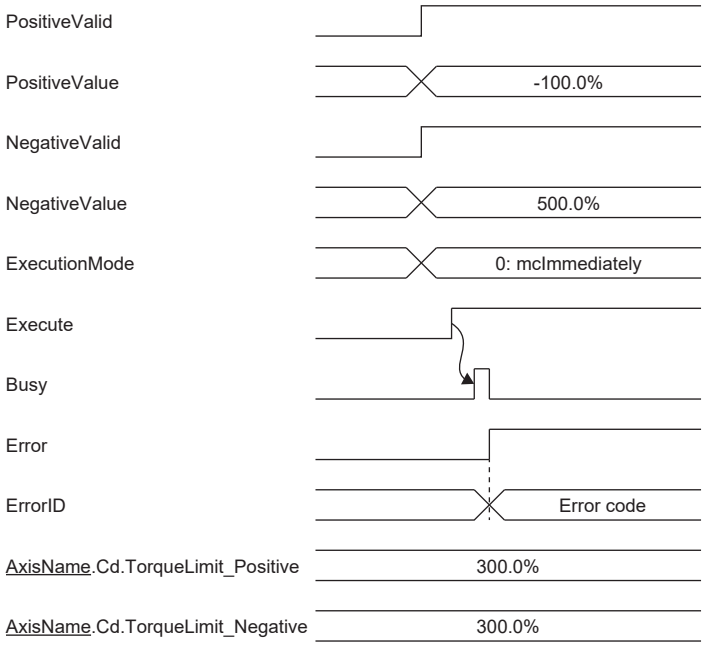
Bit	Name	Description
16	Cancel accepted after FB start	Whether or not to allow a cancel after MCv_SetTorqueLimit (Torque Limit Value) is executed is set. • 0: Disabled • 1: Enabled

■Operation overview

- In MCv_SetTorqueLimit (Torque Limit Value), Positive direction torque limit value (AxisName.Cd.TorqueLimit_Positive) and Negative direction torque limit value (AxisName.Cd.TorqueLimit_Negative) are changed.



- If Execution command (Execute) is set to TRUE while a value outside the range is set in Positive direction torque limit value (PositiveValue) or Negative direction torque limit value (NegativeValue), an error occurs in MCv_SetTorqueLimit (Torque Limit Value) and Positive direction torque limit value (AxisName.Cd.TorqueLimit_Positive) and Negative direction torque limit value (AxisName.Cd.TorqueLimit_Negative) will not be changed.



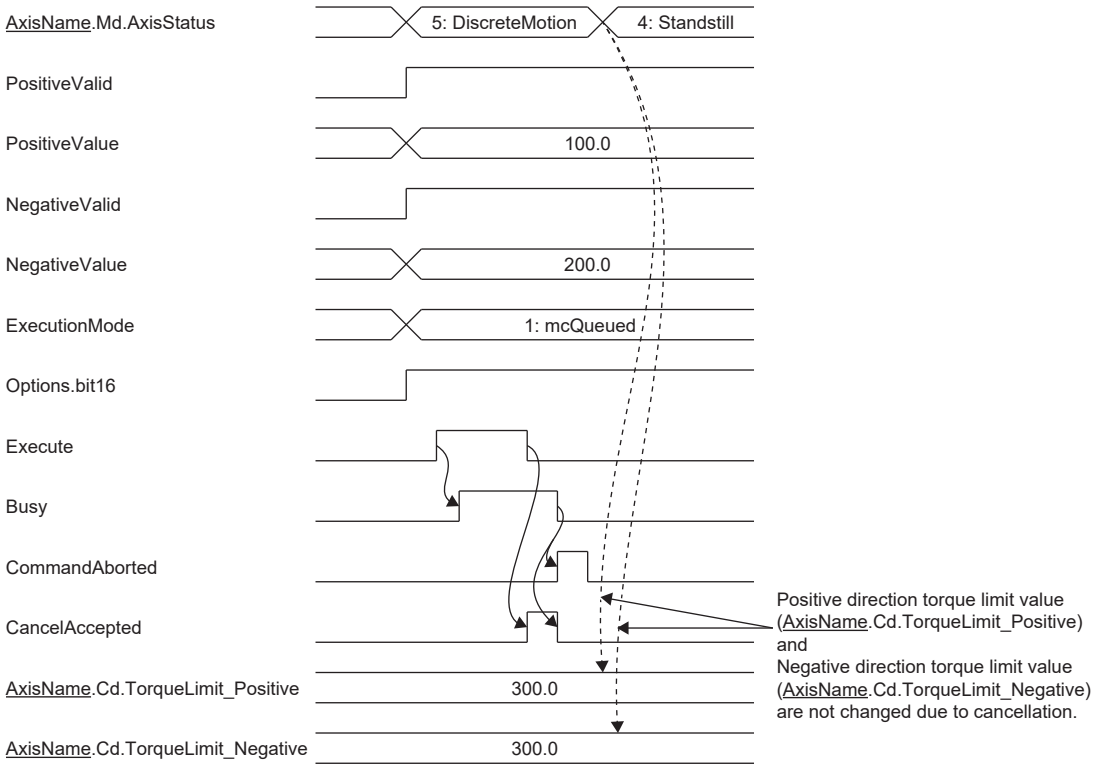
■Cancel

A torque limit value change that is in the standby status can be canceled after MCv_SetTorqueLimit (Torque Limit Value) execution.

- To enable cancel, execute MCv_SetTorqueLimit (Torque Limit Value) with the status of "Cancel accepted after FB start (Options (Options): Bit 16)" set to "1: Allow".
- Cancel is started at falling edge detection of Execution command (Execute).
- Cancel is accepted only when Executing (Busy) of the output pin is TRUE.
- When MCv_SetTorqueLimit (Torque Limit Value) accepts the cancel, Cancel acceptance (CancelAccepted) of the output pin becomes TRUE.
- When cancel is completed, Abortion of execution (CommandAborted) of the output pin becomes TRUE.
- When cancelled, Positive direction torque limit value (AxisName.Cd.TorqueLimit_Positive) and Negative direction torque limit value (AxisName.Cd.TorqueLimit_Negative) will not be changed.

Ex.

After starting with Execution mode (ExecutionMode) set to "1: Execute at completion (mcQueued)", the change is cancelled while Axis Status (AxisName.Md.AxisStatus) is waiting to switch to "4: Standby (Standstill)"



Precautions

- In MCv_SetTorqueLimit (Torque Limit Value), changing of Positive direction torque limit value (AxisName.Cd.TorqueLimit_Positive) and Negative direction torque limit value (AxisName.Cd.TorqueLimit_Negative) is performed.

Program example

The following is a program example. The torque limit value change command (bSetTorqueLimit) is set to TRUE, and the positive direction torque limit value for Axis 1 (Axis0001) is changed to "100.0" and the negative direction torque limit value for Axis 1 (Axis0001) is changed to "200.0".

■Axis

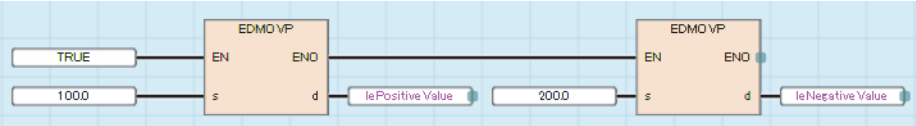
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

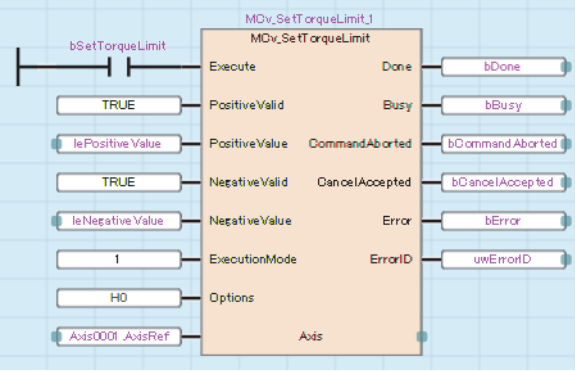
Label name	Data type	Comment
MCv_SetTorqueLimit_1	MCv_SetTorqueLimit	Torque Limit Value FB
bSetTorqueLimit	Bit	Torque limit value change command
lePositiveValue	Double-precision real number	Positive direction torque limit value
leNegativeValue	Double-precision real number	Negative direction torque limit value
bDone	Bit	Ready
bBusy	Bit	Executing
bCommandAborted	Bit	Abortion of execution
bCancelAccepted	Bit	Cancel acceptance
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

- Data setting for torque limit value change



- Torque limit value change



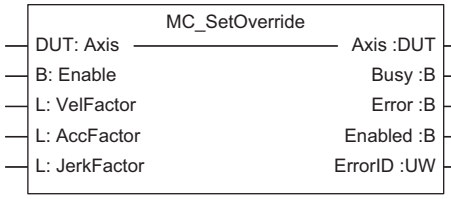
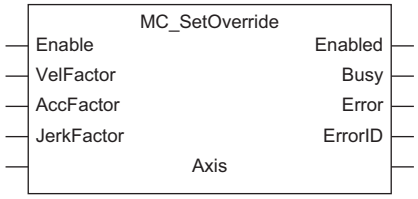
■ST program

```
//-----Data setting for torque limit value change-----
lePositiveValue:= 100.0;
leNegativeValue:= 200.0;

//-----Torque limit value change-----
MCv_SetTorqueLimit_1(
    Axis:= Axis0001.AxisRef ,
    Execute:= bSetTorqueLimit ,
    PositiveValid:= TRUE ,
    PositiveValue:= lePositiveValue ,
    NegativeValid:= TRUE ,
    NegativeValue:= leNegativeValue ,
    ExecutionMode:= MC_EXECUTION_MODE__mcQueued ,
    Options:= H00000000 ,
    Done=> bDone ,
    Busy=> bBusy ,
    CommandAborted=> bCommandAborted ,
    CancelAccepted=> bCancelAccepted ,
    Error=> bError ,
    ErrorID=> uwErrorID
);
```

41.6 Override Value Setting

MC_SetOverride

This FB changes the target velocity, the target acceleration, and the target deceleration of the specified axis.

Ladder	FBD/LD	ST
		MC_SetOverride(Axis:= ?AXIS_REF? , Enable:= ?BOOL? , VelFactor:= ?LREAL? , AccFactor:= ?LREAL? , JerkFactor:= ?LREAL? , Enabled=> ?BOOL? , Busy=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Setting range	Default value	Description
Enable	Enable	BOOL	Always	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_SetOverride (Override Value Setting) is executed.
VelFactor	Velocity override factor	LREAL	Always	0.00 to 10.00	0.00	The velocity override factor is set. When Enable (Enable) is TRUE, the value is always imported.
AccFactor	Acceleration override factor	LREAL	Always	0.00, 0.01 to 10.00	0.00	The acceleration override factor is set. When Enable (Enable) is TRUE, the value is always imported. When "0.00" is set, the acceleration override factor is not changed and the control is performed with the previous value.
JerkFactor	Jerk override factor	LREAL	Always	0.0	0.0	"0.0" must be set. *: If a value other than "0.0" is set, Out of Jerk Override Coefficient (JerkFactor) Range (error code: 1BBEH) occurs.

■Output variables

Output variable	Name	Data type	Default value	Description
Enabled	Enabled	BOOL	FALSE	This variable becomes TRUE when the override values are set correctly.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_SetOverride (Override Value Setting) is executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. User's manual for the controller used

Processing details

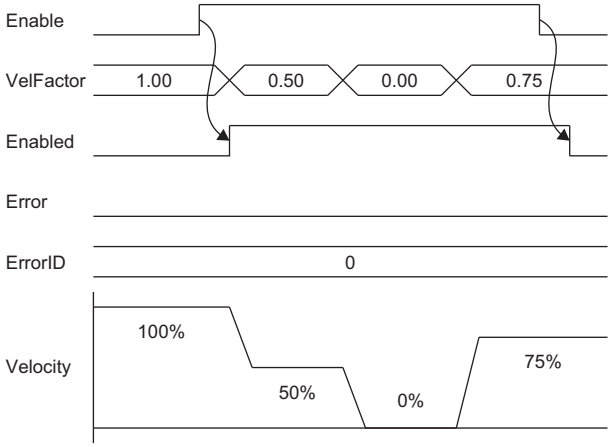
- This FB changes the target velocity, the target acceleration, and the target deceleration of the specified axis.
- The current operating target velocity, target acceleration, and target deceleration values are changed by multiplying each value by the override factor.
- MC_SetOverride (Override Value Setting) is executed when Enable (Enable) becomes TRUE. Enabled (Enabled) becomes TRUE while the override factor is valid.
- If the override factor value is changed while Enable (Enable) is TRUE, the new override factor is reflected.
- When an error occurs in MC_SetOverride (Override Value Setting), Error (Error) becomes TRUE and the error code is stored in Error code (ErrorID). For details on error codes, refer to the following.

📖 User's manual for the controller used

- When the value for Velocity override factor (VelFactor) is set to "0.00", the axis stops without Axis status (AxisName.Md.AxisStatus) changing to "4: Standby (Standstill)".
- When the value for Acceleration override factor (AccFactor) is set to "0.00", the acceleration override factor is not changed and the previous acceleration override factor is maintained.

■Timing chart

- When the processing is completed successfully



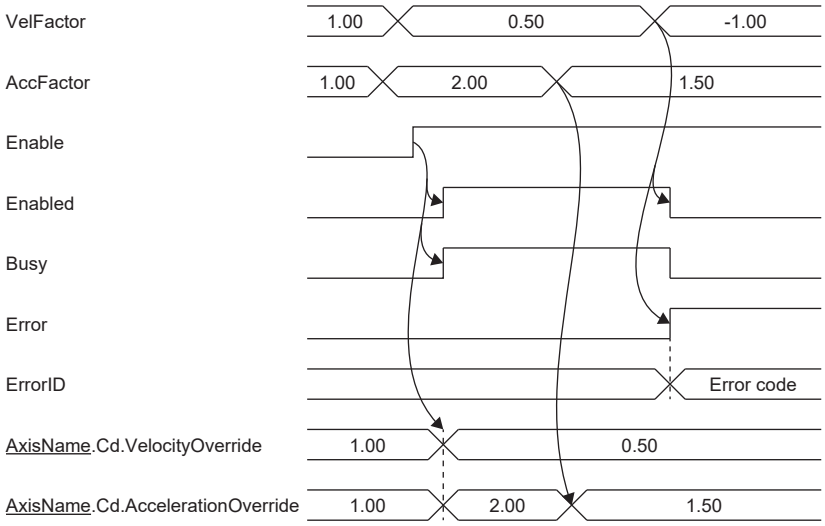
- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1390 Basic operation of Enable (Enable) type motion control FBs

■Operation overview

- In MC_SetOverride (Override Value Setting), Velocity override factor (AxisName.Cd.VelocityOverride) and Acceleration override factor (AxisName.Cd.AccelerationOverride) are changed.



- If a value outside the range is set as the override factor, an error occurs in MC_SetOverride (Override Value Setting) and subsequent values will not be imported. To import values again, restart Enable (Enable).

Precautions

- Do not arrange two or more MC_SetOverride (Override Value Setting) in the same axis. The operation is not guaranteed if two or more of this FB are set.
- In single axis control, only the override factor set with the axis control data has an effect.
- Do not operate Velocity override factor (AxisName.Cd.VelocityOverride) or Acceleration override factor (AxisName.Cd.AccelerationOverride) directly while executing MC_SetOverride (Override Value Setting).
- When the speed after Speed override is out of range, refer to "Velocity Range" in the following.

📖 User's manual for the controller used

- When the acceleration, deceleration, acceleration time, and deceleration time after acceleration override is out of range, refer to "Acceleration/deceleration processing function" in the following.

📖 User's manual for the controller used

Program example

The following is a program example. The override value change command (bSetOverride) is set to TRUE, and the target velocity, target acceleration, and target deceleration of Axis 1 (Axis0001) are changed to values multiplied by a velocity override factor of "1.0" and an acceleration/deceleration override factor of "2.0".

■Axis

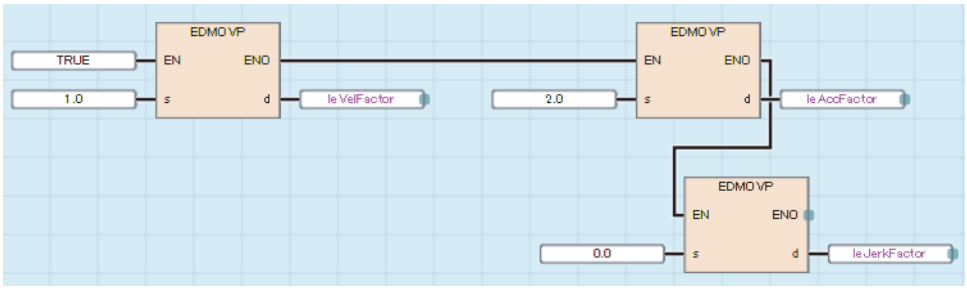
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

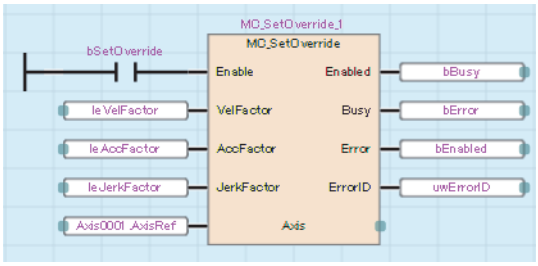
Label name	Data type	Comment
MC_SetOverride_1	MC_SetOverride	Override Value Setting FB
bSetOverride	Bit	Override value change command
leVelFactor	Double-precision real number	Velocity override factor
leAccFactor	Double-precision real number	Acceleration/deceleration override factor
leJerkFactor	Double-precision real number	Jerk override factor
bEnabled	Bit	Enabled
bBusy	Bit	Executing
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data setting for override value change



• Override value change



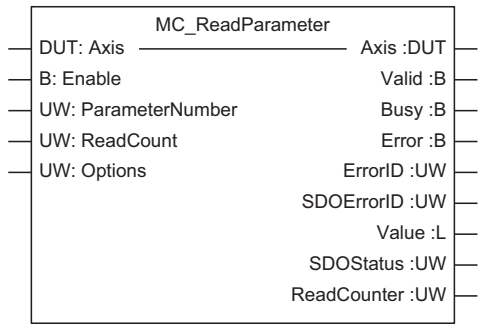
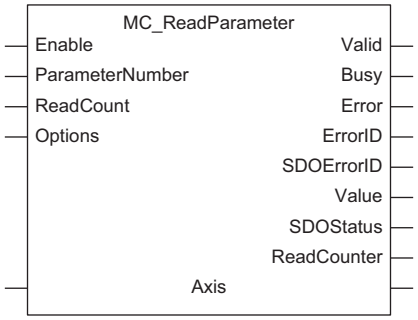
■ST program

```
//-----Data setting for override value change-----
leVelFactor:= 1.00;
leAccFactor:= 2.00;
leJerkFactor:= 0.0;

//-----Override value change-----
MC_SetOverride_1(
   Axis:= Axis0001.AxisRef ,
   Enable:= bSetOverride ,
   VelFactor:= leVelFactor ,
   AccFactor:= leAccFactor ,
   JerkFactor:= leJerkFactor ,
   Enabled=> bEnabled ,
   Busy=> bBusy ,
   Error=> bError ,
   ErrorID=> uwErrorID
);
```

41.7 Parameter Read

MC_ReadParameter

This FB reads objects of a device.

Ladder	FBD/LD	ST
		MC_ReadParameter(Axis:= ?AXIS_REF? , Enable:= ?BOOL? , ParameterNumber:= ?DWORD? , ReadCount:= ?WORD? , Options:= ?DWORD? , Valid=> ?BOOL? , Busy=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD? , SDOErrorID=> ?DWORD? , Value=> ?LREAL? , SDOStatus=> ?WORD? , ReadCounter=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Setting range	Default value	Description
Enable	Enable	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_ReadParameter (Parameter Read) is executed.
ParameterNumber	Parameter No.	DWORD(HEX)	Always	00010000H to FFFFFFFFH	00000000H	The object of a device is set. For details, refer to the following. ☞ Page 1517 Parameter No. (ParameterNumber)
ReadCount	Number of reading times	WORD(UNIT)	Always	0 to 65535	0	The number of parameter reading times is set. When "0" is set, the parameter is continuously read until Enable (Enable) is set to FALSE. Reimport is performed when the parameter No. is changed.
Options	Options	DWORD(HEX)	Always	00000000H to 00010000H	00000000H	The function options for MC_ReadParameter (Parameter Read) are set in the specified bit position. Reimport is performed when the parameter No. is changed. For details, refer to the following. ☞ Page 1517 Options (Options)

■Output variables

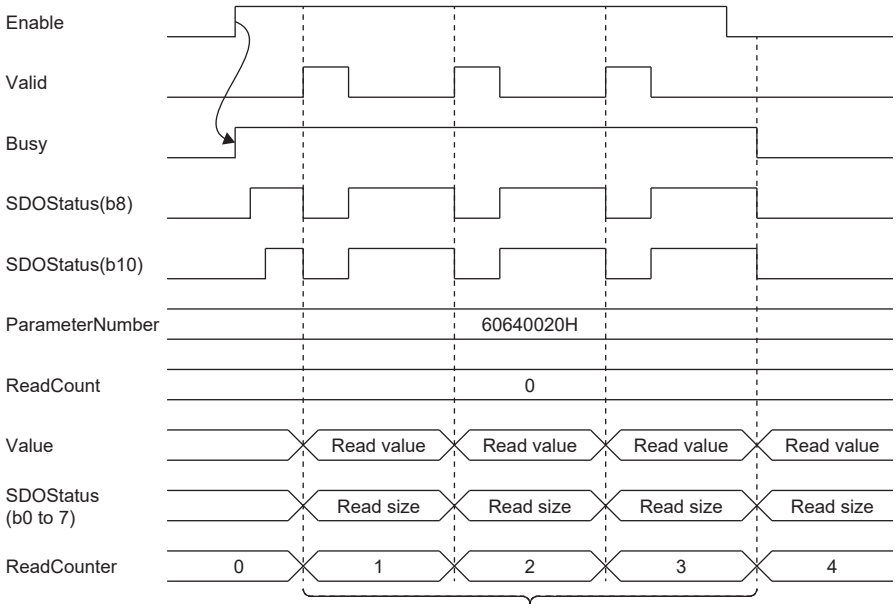
Output variable	Name	Data type	Default value	Description
Valid	Output value valid	BOOL	FALSE	This variable becomes TRUE when the output value is valid.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_ReadParameter (Parameter Read) is executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used
SDOErrorID	Transient error code	DWORD(HEX)	00000000H	When an error occurs in SDO communication, this variable returns the response code (SDO Abort Code).
Value	Read value	LREAL	0.0	This variable outputs the value read from the specified parameter. The value is stored as LREAL type even when the target parameter data is integer.
SDOStatus	SDO transfer status	WORD(UINT)	0	This variable stores the processing status of the transient request. For details, refer to the following. ☞ Page 1517 SDO transfer status (SDOStatus)
ReadCounter	Number of reading times	WORD(UINT)	0	This variable stores the number of parameter reading times. When "0" is set to Number of reading times (ReadCount), the variable works as a ring counter of "0 to 65535".

Processing details

- This FB reads object data by setting a value later than 00010000H in Parameter No. (ParameterNumber). In this case, the object data is sent/received using the transient transmission function.
- When a transient request error (response code (SDO Abort Code)) is received from the device, Error (Error) becomes TRUE, SDO Communication Error (error code: 1A48H) is stored in Error code (ErrorID), and the response code (SDO Abort Code) from the device is stored in Transient error code (SDOErrorID).

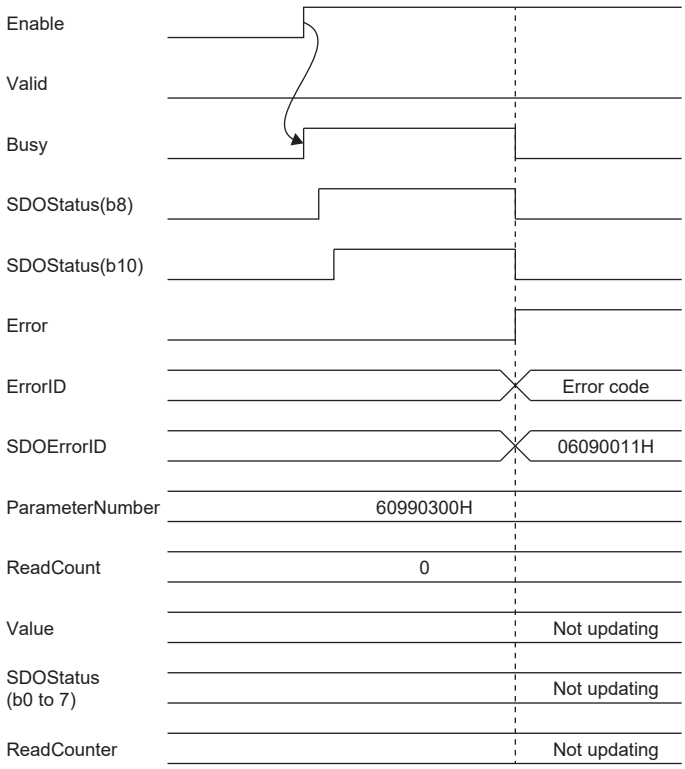
■Timing chart

- When the processing is completed successfully

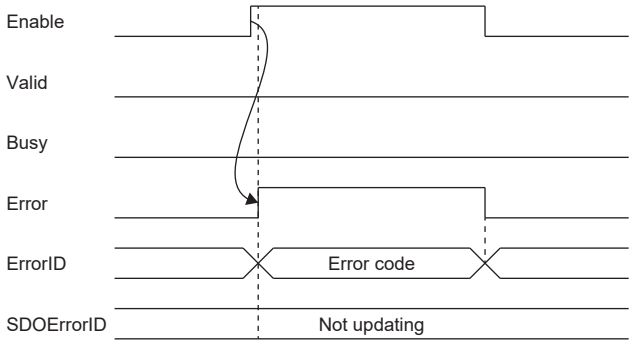


Reading is performed for the number of times set in the number of reads (ReadCount) until Enable becomes FALSE.
(When the number of reads (ReadCount) is "0", reading is continuously performed until Enable becomes FALSE.)

- When the processing is completed with an error
  - SDO communication error



  - I/O variable or input variable error

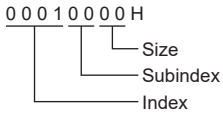


■Parameter No. (ParameterNumber)

• The classification of Parameter No. (ParameterNumber) set when reading parameters is shown below.

Parameter No.	Description
00010000H to FFFFFFFFH	The object of the device.

• The contents of Parameter No. (ParameterNumber) when setting the object of the device are shown below.



• When setting the size of the objects of the device, specify 1 byte (08H)/ 2 bytes (10H)/ 4 bytes (20H)/ 8 bytes (40H) in bits. If "00H" is specified, the parameter read is performed with default size. If a value other than the above setting values is specified, Out of Parameter No. Range (error code: 1BC2H) occurs.

• When the size specified with Parameter No. (ParameterNumber) differs from default size of the object data, the operations depend on the specifications of the drive unit.

Ex.

When setting object index "6099H", UNSIGNED32 object of subindex "02H", and a size of "00H".

Set "60990200H".

Point

For the objects of each device, refer to the device manual.

■Options (Options)

The function options for MC_ReadParameter (Parameter Read) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.

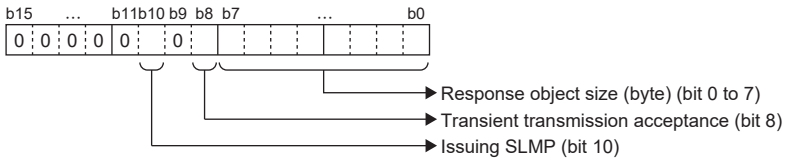


*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
16	Specifies unsigned	Whether to store the read value of the integer data with or without a sign is set. • 0: Signed • 1: Unsigned

■SDO transfer status (SDOStatus)

This variable stores the processing status of the transient request.



Bit	Name	Description
0 to 7	Response object size (byte)	The object size sent by the devices is stored when the processing is completed.
8	The transient transmission acceptance	The transient transmission status is stored. • 0: Unexecuted transient transmission • 1: Accepting transient transmission
10	Issuing SLMP	The SLMP status is stored. • 0: Unexecuted SLMP • 1: Issuing SLMP

Precautions

- Up to four transient transmissions can be executed toward the same axis. If MC_ReadParameter (Parameter Read) is executed for object data towards an axis which is already executing four transient transmissions, Parameter Reading And Writing FB Execution Disabled Error (error code: 1BC1H) occurs and the next FB is not executed. However, the FB being executed continues.
- For the driver homing method, home position data of the drive unit is obtained using the transient transmission function. Therefore, if a parameter read for object data is executed when MC_Home (OPR) is being executed, ABS Reference Point Read Error (error code: 1AF9H) may occur in MC_Home (OPR).
- When reading the integer type/BOOL type parameters or object data, the value read with MC_ReadParameter (Parameter Read) is converted into an LREAL type or BOOL type and output.

Ex.

Reading parameters while specifying two bytes size in MC_ReadParameter (Parameter Read).

Parameter value	Read value (Value) (Data type: LREAL)	
	Specifies unsigned (Options (Options): Bit 16) "0: Signed"	Specifies unsigned (Options (Options): Bit 16) "1: Unsigned"
10000	10000.0	10000.0
-20000	-20000.0	45536.0
65535	-1.0	65535.0
63356	0.0	0.0
131701	-1.0	65535.0

- Values changed with MC_ReadParameter (Parameter Read) are not saved. To use the changed values at the next power supply, save the parameters.
  For how to save parameters of objects of devices, refer to the manual for the devices.
- When devices in transient transmission are disconnected, all the transient requests for disconnected devices fail, and SDO Communication Error (error code: 1A48H) occurs.
- Before executing MC_ReadParameter (Parameter Read), ensure that Driver status (AxisName.Md.Driver_State) is not set to "-1: Axis variable not initialized/axis parameter error (Invalid)" or "0: Not Ready To Switch On (NotReadyToSwitchOn)". If executed in a status other than the above, Parameter Reading And Writing FB Execution Disabled Error (error code: 1BC1H) will occur.

Program example

The following is a program example. The parameter read command (bReadParameter) is set to TRUE, and the values in the objects (index "6099H", subindex "02H", and size "00H") of Axis1 (Axis0001) are read.

■Axis

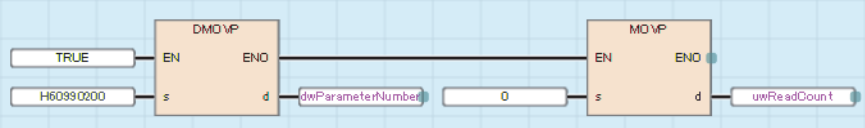
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

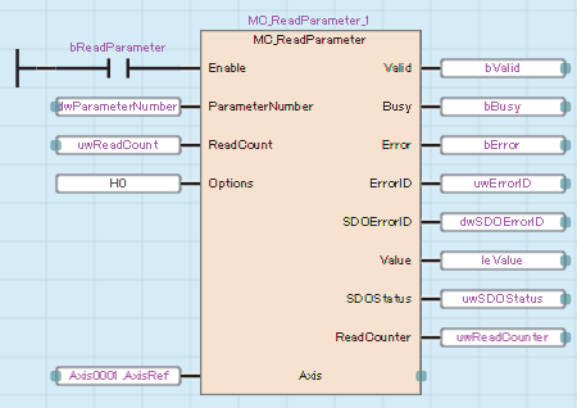
Label name	Data type	Comment
MC_ReadParameter_1	MC_ReadParameter	Parameter Read FB
bReadParameter	Bit	Program read command
dwParameterNumber	Double word [unsigned]/bit string [32-bit] (hexadecimal)	Parameter No.
uwReadCount	Word [unsigned]/bit string [16-bit]	Number of reading times
bValid	Bit	Output value valid
bBusy	Bit	Executing
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code
dwSDOErrorID	Double word [unsigned]/bit string [32-bit] (hexadecimal)	Transient error code
leValue	Double-precision real number	Read value
uwSDOStatus	Word [unsigned]/bit string [16-bit]	SDO transfer status
uwReadCounter	Word [unsigned]/bit string [16-bit]	Number of reading times

■FBD/LD program

• Data setting for parameter read



• Parameter read



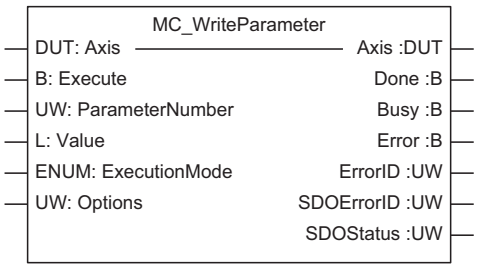
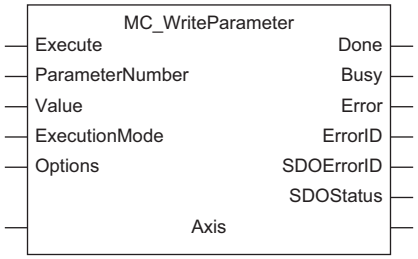
■ST program

```
//-----Data setting for parameter read-----
dwParameterNumber:= H60990200;
uwReadCount:= 0;

//-----Parameter read-----
MC_ReadParameter_1(
   Axis:= Axis0001.AxisRef ,
   Enable:= bReadParameter ,
   ParameterNumber:= dwParameterNumber ,
   ReadCount:= uwReadCount ,
   Options:= H00000000 ,
   Valid=> bValid ,
   Busy=> bBusy ,
   Error=> bError ,
   ErrorID=> uwErrorID ,
   SDOErrorID=> dwSDOErrorID ,
   Value=> leValue ,
   SDOStatus=> uwSDOStatus ,
   ReadCounter=> uwReadCounter
);
```

41.8 Parameter Write

MC_WriteParameter

This FB writes objects of a device.

Ladder	FBD/LD	ST
		MC_WriteParameter(Axis:= ?AXIS_REF? , Execute:= ?BOOL? , ParameterNumber:= ?DWORD? , Value:= ?LREAL? , ExecutionMode:= ?INT? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD? , SDOErrorID=> ?DWORD? , SDOStatus=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_WriteParameter (Parameter Write) is executed.
ParameterNumber	Parameter No.	DWORD(HEX)	At start	00010000H to FFFFFFFFH	00000000H	The object of a device is set. For details, refer to the following. ☞ Page 1524 Parameter No. (ParameterNumber)
Value	Setting value	LREAL	At start	—	0.0	The setting value of the set parameter is set.
ExecutionMode	Execution mode	INT (MC_EXECUTION_MODE)	At start	0, 1	0	The writing method of the set parameter is set. • 0: Execute immediately (mcImmediately) • 1: Execute at completion (mcQueued) For details, refer to the following. ☞ Page 1524 Execution mode (ExecutionMode)
Options	Options	DWORD(HEX)	At start	00000000H to 00010000H	00000000H	The function options for MC_WriteParameter (Parameter Write) are set in the specified bit position. For details, refer to the following. ☞ Page 1524 Options (Options)

■Output variables

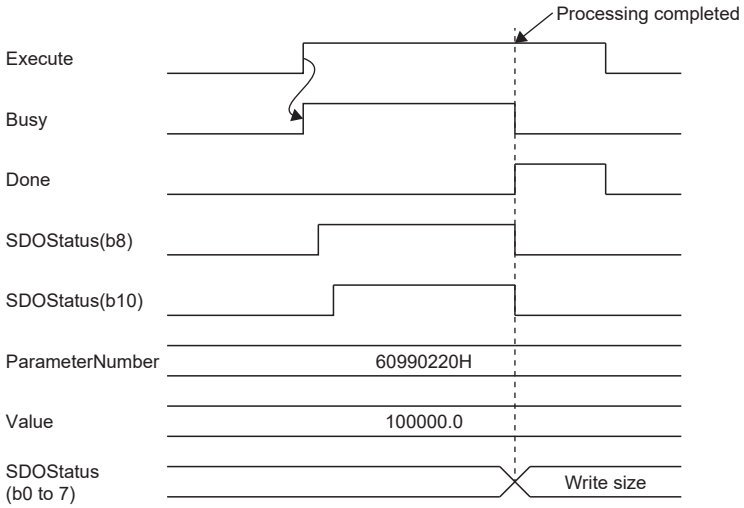
Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the parameter writing is completed.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_WriteParameter (Parameter Write) is executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used
SDOErrorID	Transient error code	DWORD(HEX)	00000000H	When an error occurs in SDO communication, this variable returns the response code (SDO Abort Code).
SDOStatus	SDO transfer status	WORD(UINT)	0	This variable stores the processing status of the transient request. For details, refer to the following. ☞ Page 1524 SDO transfer status (SDOStatus)

Processing details

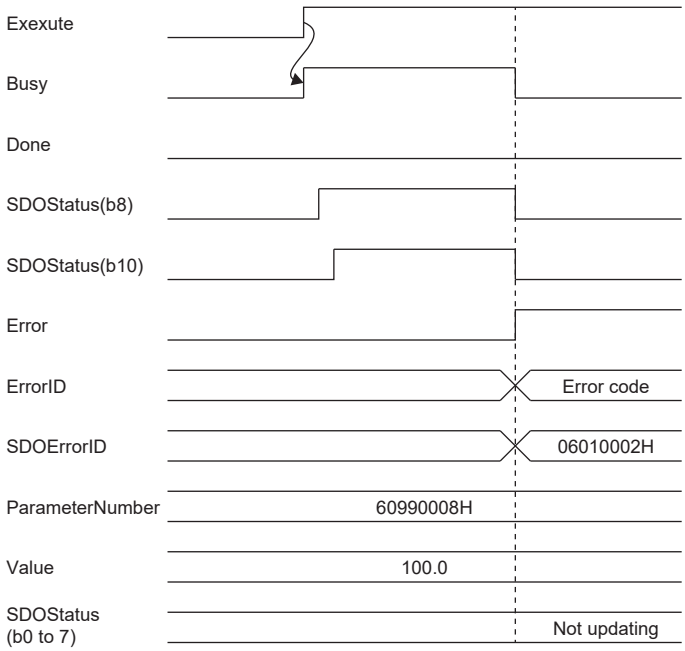
- This FB writes object data by setting a value later than 00010000H in Parameter No. (ParameterNumber). In this case, the object data is sent/received using the transient transmission function.
- When a transient request error (response code (SDO Abort Code)) is received from the device, Error (Error) becomes TRUE, SDO Communication Error (error code: 1A48H) is stored in Error code (ErrorID), and the response code (SDO Abort Code) from the device is stored in Transient error code (SDOErrorID).

■Timing chart

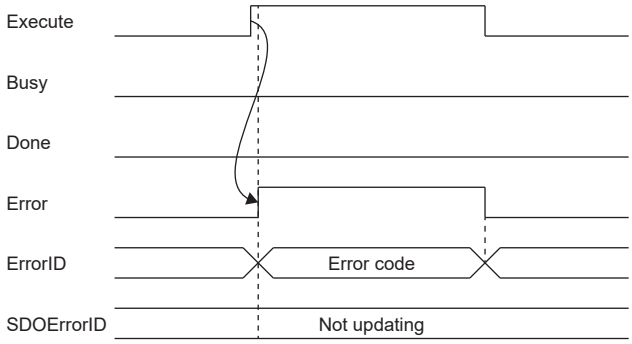
- When the processing is completed successfully



- When the processing is completed with an error
  - SDO communication error



  - I/O variable or input variable error

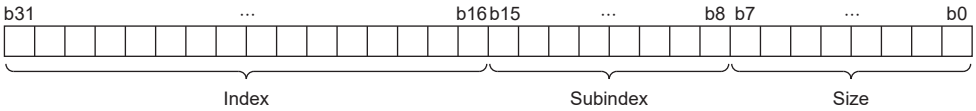


■Parameter No. (ParameterNumber)

- The classification of Parameter No. (ParameterNumber) set when writing parameters is shown below.

Parameter No.	Description
00010000H to FFFFFFFFH	The object of the device.

- The contents of Parameter No. (ParameterNumber) when setting the object of the device are shown below.



- When setting the size of the objects of the device, specify 1 byte (08H)/ 2 bytes (10H)/ 4 bytes (20H)/ 8 bytes (40H) in bits. If "00H" is specified, the parameter write is performed at 4 bytes. If a value other than the above setting values is specified, Out of Parameter No. Range (error code: 1BC2H) occurs.
- When the size specified with Parameter No. (ParameterNumber) differs from default size of the object data, the operations depend on the specifications of the drive unit.

Ex.

When setting object index "6099H", UNSIGNED32 object of subindex "02H", and a size of "00H".

Set "60990200H".

■Execution mode (ExecutionMode)

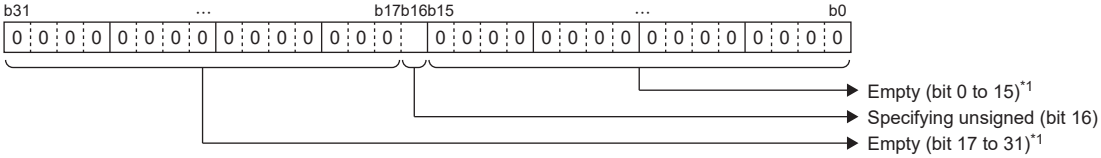
When executing a parameter write, set the timing of the write in Execution mode (ExecutionMode).

Setting value	Description
0: Execute immediately (mcImmediately)	The write is executed immediately. The axis operations might be affected because the reflection timing depends on each parameter.
1: Execute at completion (mcQueued)	The write is executed after the axis stops (Axis status (AxisName.Md.AxisStatus) is "0: Axis disabled (Disabled)" or "4: Standby (Standstill)").

■Options (Options)

The function options for MC_WriteParameter (Parameter Write) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.

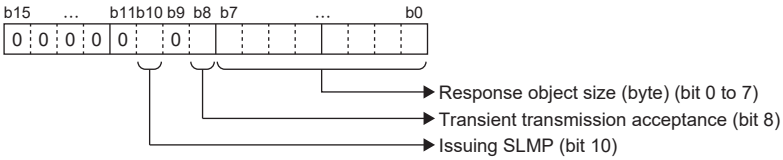


*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
16	Specifies unsigned	Whether to store the setting value of the integer data with or without a sign is set. • 0: Signed • 1: Unsigned

■SDO transfer status (SDOStatus)

This variable stores the processing status of the transient request.



Bit	Name	Description
0 to 7	Response object size (byte)	The object size sent by the devices is stored when the processing is completed.
8	The transient transmission acceptance	The transient transmission status is stored. • 0: Unexecuted transient transmission • 1: Accepting transient transmission
10	Issuing SLMP	The SLMP status is stored. • 0: Unexecuted SLMP • 1: Issuing SLMP

Precautions

- Up to four transient transmissions can be executed toward the same axis. If MC_WriteParameter (Parameter Write) is executed for object data towards an axis which is already executing four transient transmissions, Parameter Reading And Writing FB Execution Disabled Error (error code: 1BC1H) occurs and the next FB is not executed. (The FB being executed continues.)
- For the driver homing method, home position data of the drive unit is obtained using the transient transmission function. Therefore, if a parameter write for object data is executed when MC_Home (OPR) is being executed, ABS Reference Point Read Error (error code: 1AF9H) may occur in MC_Home (OPR).
- When writing the integer type/BOOL type parameters or object data, the setting value of MC_WriteParameter (Parameter Write) is converted into an integer type or BOOL type and imported.

Ex.

Writing parameters while specifying two bytes size in MC_WriteParameter (Parameter Write).

Setting value (Value) (Data type: LREAL)	Write value (2 bytes data)	
	Specifies unsigned (Options (Options): Bit 16) "0: Signed"	Specifies unsigned (Options (Options): Bit 16) "1: Unsigned"
10000.5	10000	10000
-10000.4	-10000	55536
65535.0	-1	65535
-65535.0	1	1
63356.0	0	0
131701.0	-1	65535

- Values changed with MC_WriteParameter (Parameter Write) are not saved. To use the changed values at the next power supply, save the parameters.
  For how to save parameters of objects of devices, refer to the manual for the devices.
- When devices in transient transmission are disconnected, all the transient requests for disconnected devices fail, and SDO Communication Error (error code: 1A48H) occurs.
- Before executing MC_WriteParameter (Parameter Write), ensure that Driver status (AxisName.Md.Driver_State) is not set to "-1: Axis variable not initialized/axis parameter error (Invalid)" or "0: Not Ready To Switch On (NotReadyToSwitchOn)". If executed in a status other than the above, Parameter Reading And Writing FB Execution Disabled Error (error code: 1BC1H) will occur.

Program example

The following is a program example. The parameter write command (bWriteParameter) is set to TRUE, and "100.0" is written to the objects (index "6099H", subindex "02H", and size "00H") of Axis1 (Axis0001).

■Axis

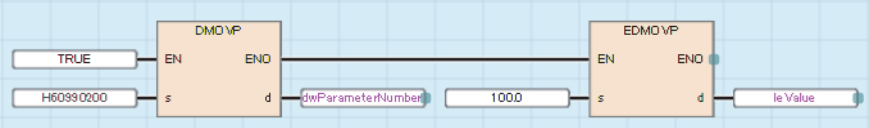
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

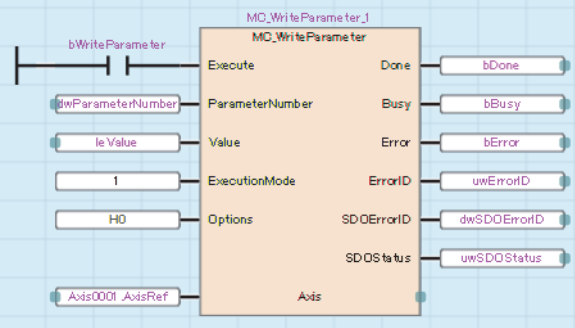
Label name	Data type	Comment
MC_WriteParameter_1	MC_WriteParameter	Parameter Write FB
bWriteParameter	Bit	Parameter write command
dwParameterNumber	Double word [unsigned]/bit string [32-bit] (hexadecimal)	Parameter No.
leValue	Double-precision real number	Setting value
bDone	Bit	Execution completion
bBusy	Bit	Executing
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code
dwSDOErrorID	Double word [unsigned]/bit string [32-bit] (hexadecimal)	Transient error code
uwSDOStatus	Word [unsigned]/bit string [16-bit]	SDO transfer status

■FBD/LD program

• Data setting for parameter write



• Parameter write



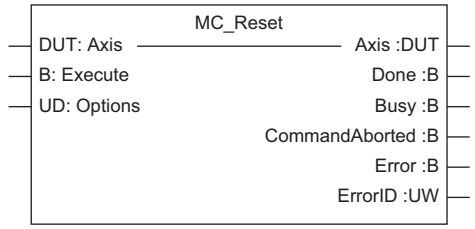
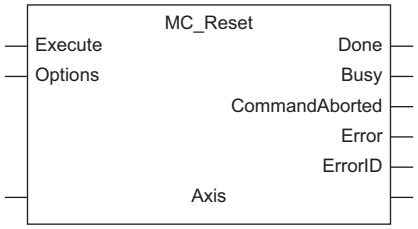
■ST program

```
//-----Data setting for parameter write-----
dwParameterNumber:= H60990200;
leValid:= 100.0;

//-----Parameter write-----
MC_WriteParameter_1(
    Axis:= Axis0001.AxisRef ,
    Execute:= bWriteParameter ,
    ParameterNumber:= dwParameterNumber ,
    Value:= leValue ,
    ExecutionMode:= MC_EXECUTION_MODE__mcQueued ,
    Options:= H00000000 ,
    Done=> bDone ,
    Busy=> bBusy ,
    Error=> bError ,
    ErrorID=> uwErrorID ,
    SDOErrorID=> dwSDOErrorID ,
    SDOStatus=> uwSDOStatus
);
```

41.9 Axis Error Reset

MC_Reset

This FB resets errors and warnings of the axis.

Ladder	FBD/LD	ST
		MC_Reset(Axis:= ?AXIS_REF? , Execute:= ?BOOL? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_Reset (Axis Error Reset) is executed.
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	Set "00000000H". *: If a value other than "00000000H" is set, Out of option range (error code: 1BC3H) occurs.

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the error reset is completed.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_Reset (Axis Error Reset) is executed.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when MC_Reset (Axis Error Reset) has been aborted because of timeout. The variable becomes FALSE when Execution command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

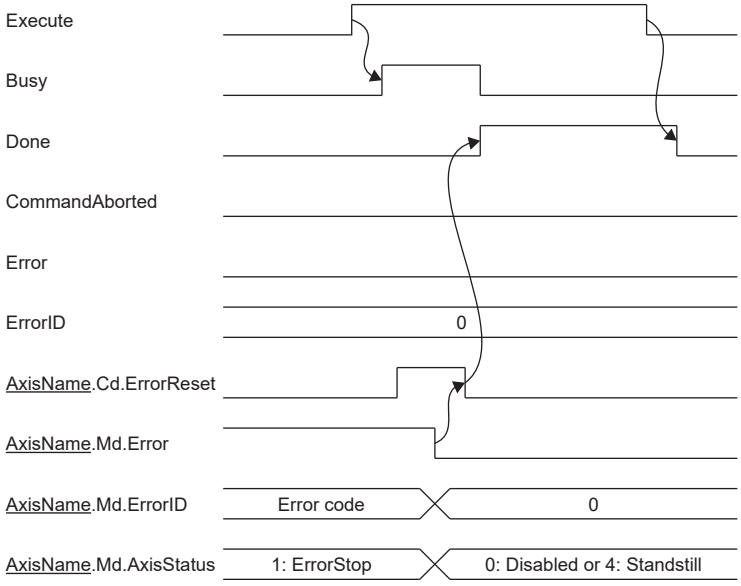
Processing details

- This FB is executed when Execution command (Execute) is set to TRUE. When the processing starts, Executing (Busy) becomes TRUE and Axis error reset (AxisName.Cd.ErrorReset) of the target axis also becomes TRUE.
- Executing (Busy) becomes FALSE and Execution completion (Done) becomes TRUE when clearing of the errors and warnings is completed.
- If executed when Axis status (AxisName.Md.AxisStatus) is other than "1: Stopping on error (ErrorStop)", Executing (Busy) becomes FALSE and Execution completion (Done) becomes TRUE after the warnings are cleared.
- Errors and warnings are not cleared if Execution command (Execute) is set to TRUE while the error and warning causes still remain. If the error is not cleared within one second after the instruction is executed, Abortion of execution (CommandAborted) becomes TRUE. Turn Execution command (Execute) to FALSE temporarily and clear the error and warning causes, then set Execution command (Execute) to TRUE again.
- When an error occurs in MC_Reset (Axis Error Reset), Error (Error) becomes TRUE and the error code is stored in Error code (ErrorID). For details on error codes, refer to the following.

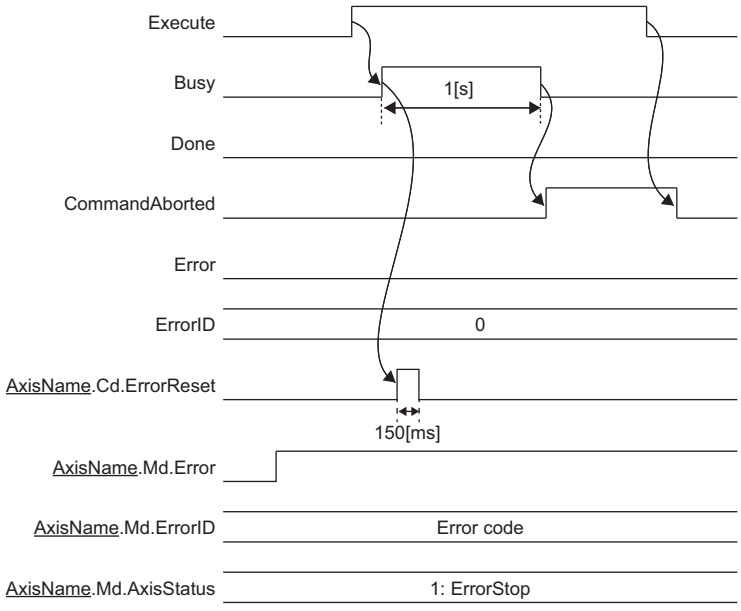
User's manual for the controller used

■Timing chart

- When the processing is completed successfully



- At timeout



Precautions

- Do not operate Axis error reset (AxisName.Cd.ErrorReset) directly while executing MC_Reset (Axis Error Reset).

Program example

The following is a program example. The error reset command (bErrorReset) is set to TRUE, and errors and warnings of Axis 1 (Axis0001) are reset.

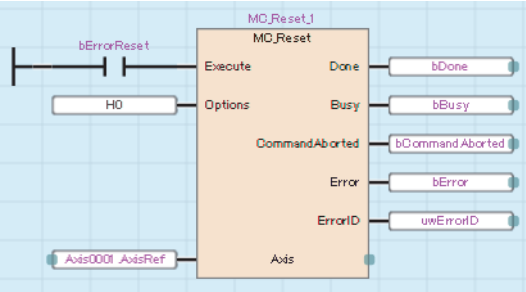
■Axis

Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

Label name	Data type	Comment
MC_Reset_1	MC_Reset	Axis Error Reset FB
bErrorReset	Bit	Error reset command
bDone	Bit	Execution completion
bBusy	Bit	Executing
bCommandAborted	Bit	Abortion of execution
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program



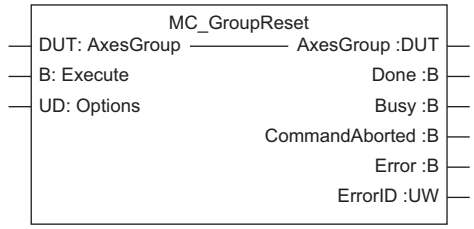
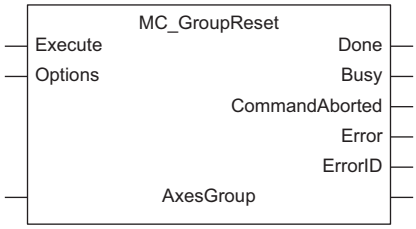
■ST program

```
MC_Reset_1(
   Axis:= Axis0001.AxisRef ,
   Execute:= bErrorReset ,
   Options:= H00000000 ,
   Done=> bDone ,
   Busy=> bBusy ,
   CommandAborted=> bCommandAborted ,
   Error=> bError ,
   ErrorID=> uwErrorID
);
```

41.10 Axes Group Error Reset

MC_GroupReset

This FB resets errors and warnings of the axes group and each axis in the axes group.

Ladder	FBD/LD	ST
		MC_GroupReset(AxesGroup:= ?AXES_GROUP_REF? , Execute:= ?BOOL? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
AxesGroup	Axes group information	AXES_GROUP_REF	At start	—	Omission not possible	An axes group is set. For the variable (AxesGroupName.AxesGroupRef.) used, refer to the following. ☞ Page 1423 AxesGroupName.AxesGroupRef. (Axes group information)

■Input variable

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_GroupReset (Axes Group Error Reset) is executed.
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	Set "00000000H". *: If a value other than "00000000H" is set, Out of option range (error code: 1BC3H) occurs.

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the error reset of all axes in the axes group is completed.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_GroupReset (Axes Group Error Reset) is executed.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when MC_GroupReset (Axes Group Error Reset) has been aborted because of timeout. The variable becomes FALSE when Execution command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

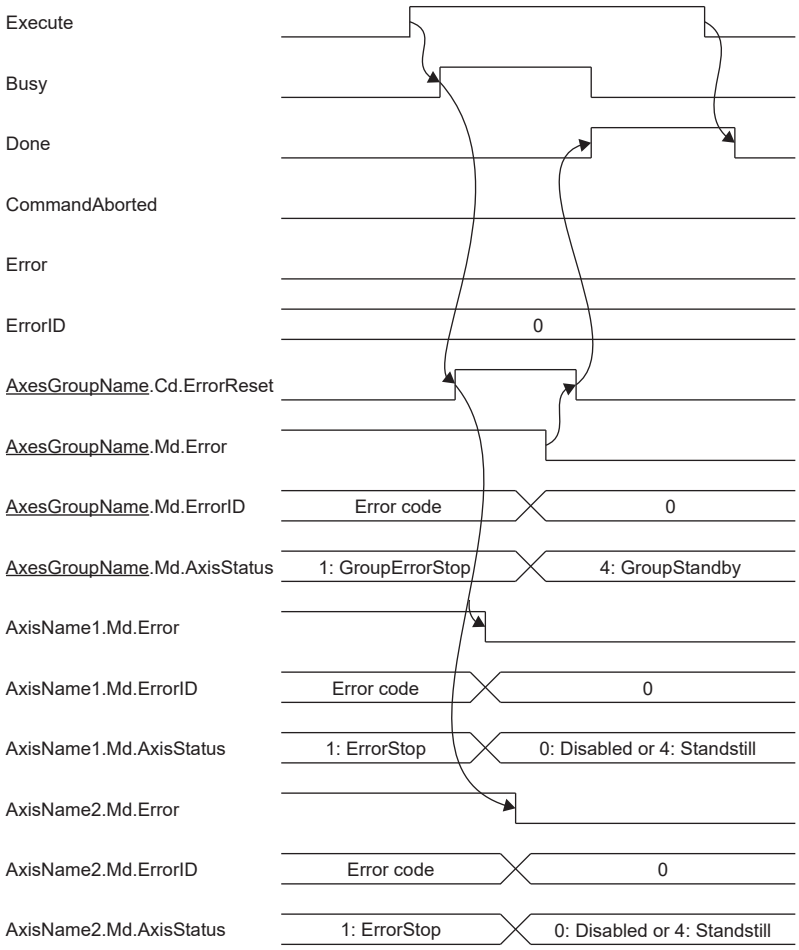
Processing details

- This FB is executed at the rising edge of Execution command (Execute). When the processing starts, Executing (Busy) becomes TRUE and Axes group error reset (AxesGroupName.Cd.ErrorReset) of the target axes group also becomes TRUE.
- Executing (Busy) becomes FALSE and Execution completion (Done) becomes TRUE when clearing of the errors and warnings is completed for the axes group or individual axis.
- If executed when Axes group status (AxesGroupName.Md.GroupStatus) is other than "1: Stopping on error (GroupErrorStop)", Executing (Busy) becomes FALSE and Execution completion (Done) becomes TRUE after the warnings are cleared.
- Axis errors and warnings are not cleared if Execution command (Execute) is set to TRUE while the error and warning causes still remain. In this case, if the errors are not cleared within one second after the FB is executed, Abortion of execution (CommandAborted) becomes TRUE and Axes group error reset (AxesGroupName.Cd.ErrorReset) of the target axes group becomes FALSE. Set Execution command (Execute) to FALSE temporarily and clear the error and warning causes, then set Execution command (Execute) to TRUE again.
- When an error occurs in MC_GroupReset (Axes Group Error Reset), Error (Error) becomes TRUE and the error code is stored in Error code (ErrorID). For details on error codes, refer to the following.

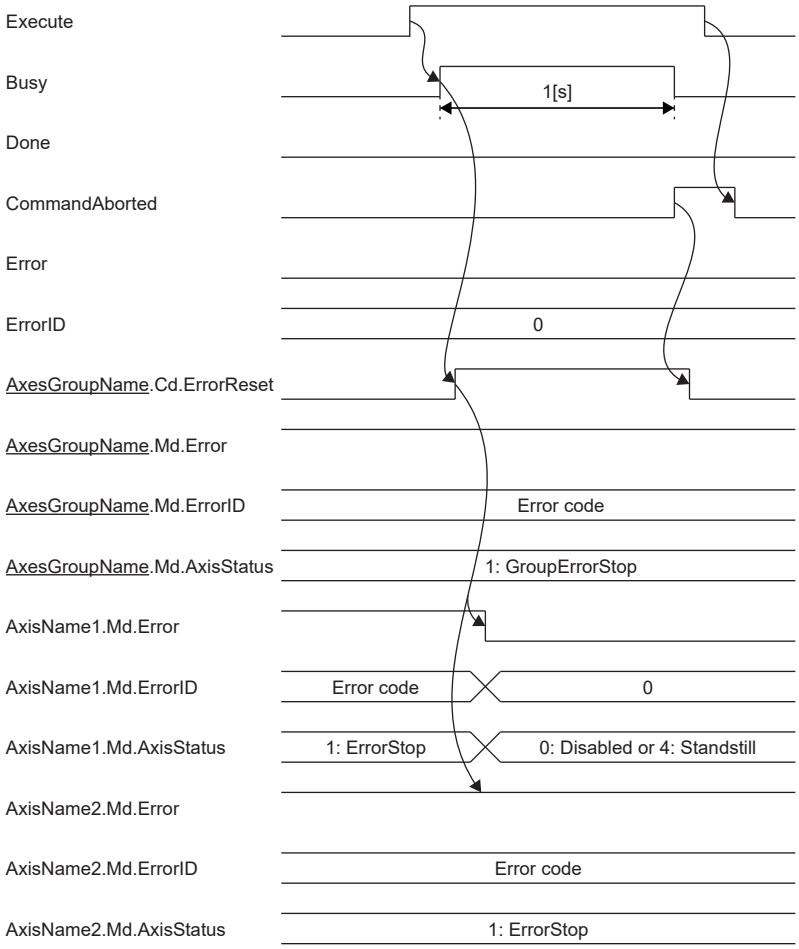
User's manual for the controller used

■Timing chart

- When the processing is completed successfully



- At timeout



Precautions

- Do not operate Axes group error reset (AxesGroupName.Cd.ErrorReset) directly while executing MC_GroupReset (Axes Group Error Reset).

Program example

The following is a program example. The axes group error reset command (bGroupErrorReset) is set to TRUE, and errors and warnings of Axes group 1 (AxesGroup001) are reset.

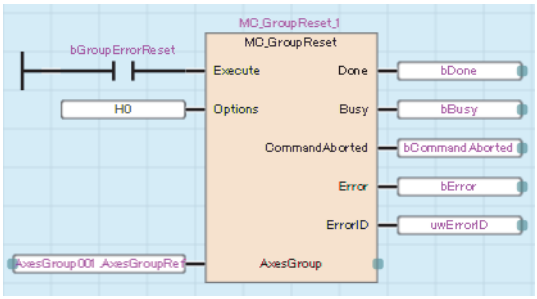
■Axes group

Axes group No.	Label name	Data type	Comment
1	AxesGroup001	AXES_GROUP_REF	Axes group 1

■Labels used

Label name	Data type	Comment
MC_GroupReset_1	MC_GroupReset	Axes Group Error Reset FB
bGroupErrorReset	Bit	Axes group error reset command
bDone	Bit	Execution completion
bBusy	Bit	Executing
bCommandAborted	Bit	Abortion of execution
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program



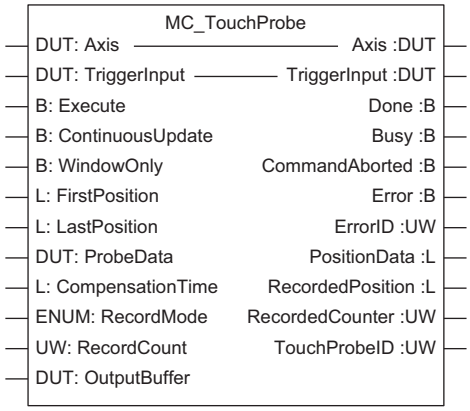
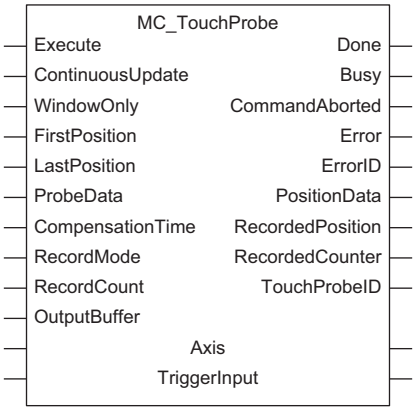
■ST program

```
MC_GroupReset_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bGroupErrorReset ,
   Options:= H00000000 ,
   Done=> bDone ,
   Busy=> bBusy ,
   CommandAborted=> bCommandAborted ,
   Error=> bError ,
   ErrorID=> uwErrorID
);
```

41.11 Touch Probe Enabled

MC_TouchProbe

This FB records optional data when a trigger event occurs.

Ladder	FBD/LD	ST
		MC_TouchProbe(Axis:= ?AXIS_REF? , TriggerInput:= ?MC_TRIGGER_REF? , Execute:= ?BOOL? , ContinuousUpdate:= ?BOOL? , WindowOnly:= ?BOOL? , FirstPosition:= ?LREAL? , LastPosition:= ?LREAL? , ProbeData:= ?TARGET_REF? , CompensationTime:= ?LREAL? , RecordMode:= ?INT? , RecordCount:= ?WORD? , OutputBuffer:= ?TARGET_REF? , Done=> ?BOOL? , Busy=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD? , PositionData=> ?LREAL? , RecordedPosition=> ?LREAL? , RecordedCounter=> ?WORD? , TouchProbeID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)
TriggerInput	Trigger input signal	MC_TRIGGER_REF	At start	—	Omission not possible	The trigger input signal is set. For details, refer to the following. ☞ Page 1540 Trigger input signal (TriggerInput)

■Input variables

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_TouchProbe (Touch Probe Enabled) is executed.
ContinuousUpdate	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	Whether to enable or disable continuous change of Enabled window (WindowOnly), Lower limit position (FirstPosition), and Upper limit position (LastPosition) is set. • FALSE: Disabled • TRUE: Enabled
WindowOnly	Enabled window	BOOL	At start/Retrigger possible/ Continuous update possible	TRUE, FALSE	FALSE	Whether to detect trigger events only in the area defined by the lower limit position and the upper limit position or not is set. • FALSE: Detected in entire area • TRUE: Detected in set area only For details, refer to the following. ☞ Page 1541 Enabled window (WindowOnly)/ Lower limit position (FirstPosition)/Upper limit position (LastPosition)
FirstPosition	Lower limit position	LREAL	At start/Retrigger possible/ Continuous update possible	—	0.0	The lower limit position of the trigger event valid area is set. The set value is included in the valid area. For details, refer to the following. ☞ Page 1541 Enabled window (WindowOnly)/ Lower limit position (FirstPosition)/Upper limit position (LastPosition)
LastPosition	Upper limit position	LREAL	At start/Retrigger possible/ Continuous update possible	—	0.0	The upper limit position of the trigger event valid area is set. The set value is included in the valid area. For details, refer to the following. ☞ Page 1541 Enabled window (WindowOnly)/ Lower limit position (FirstPosition)/Upper limit position (LastPosition)
ProbeData	Probe data	TARGET_REF	At start	—	—	The probe data is set. For details, refer to the following. ☞ Page 1542 Probe data (ProbeData)
CompensationTime	Compensation time	LREAL	At start	-5.0 to 5.0 [s]	0.0	The value to compensate for a delay of the touch probe processing is set. Set a positive value to compensate for a delay. For details, refer to the following. ☞ Page 1543 Compensation time (CompensationTime)
RecordMode	Latch mode	INT (MC_RECORD_MODE)	At start	0 to 2	0	The data latch method is set. • 0: One shot mode (OneShot) • 1: Frequency specified mode (RecordCount) • 2: Ring buffer mode (RingBuffer) For details, refer to the following. ☞ Page 1544 Latch mode (RecordMode)
RecordCount	Number of latch times	WORD(UINT)	At start	1 to 65535	1	The number of latch times is set. For details, refer to the following. ☞ Page 1544 Number of latch times (RecordCount)
OutputBuffer	Storage location of latch data	TARGET_REF	At start	—	—	The storage location of the latch data is set. When omitted, latch data is output only to Latch position (RecordedPosition). For details, refer to the following. ☞ Page 1545 Storage location of latch data (OutputBuffer)

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Completion	BOOL	FALSE	This variable becomes TRUE when the control is completed.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_TouchProbe (Touch Probe Enabled) is executed.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MC_TouchProbe (Touch Probe Enabled) is aborted because of an error or multiple start.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used
PositionData	Current value monitor data	LREAL	0.0	The current value of the latch target data is stored.
RecordedPosition	Latch position	LREAL	0.0	The current value when a trigger event occurred is set. When latch is executed multiple times, the data is overwritten by the latest data.
RecordedCounter	Number of latch times	WORD(UINT)	0	The number of latch times is stored.
TouchProbeID	Probe ID	WORD(UINT)	0	An ID unique to MC_TouchProbe (Touch Probe Enabled) is stored.

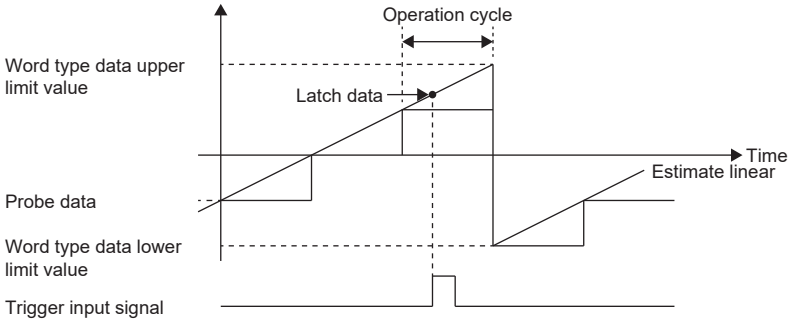
Processing details

- This FB estimates and calculates the latch data on the rising edge/falling edge/both directions of the signal specified in Trigger input signal (TriggerInput).
- When Enabled window (WindowOnly) is TRUE, checks whether the latch data at trigger event occurrence is within the range of Lower limit position (FirstPosition) and Upper limit position (LastPosition) or not. Data outside the range are not latched.
- The latch data is stored to the storage location set in Latch position (RecordedPosition) or Storage location of latch data (OutputBuffer) according to Latch mode (RecordMode), and Number of latch times (RecordedCounter) is updated.
- The probe data during operation cycle interval is calculated by estimation. The value calculated by estimation when Trigger input signal (TriggerInput) is input is the latch data. The value is calculated as shown in the figure below.

■When the probe data is WORD type

• When using an external signal high-accuracy input

The trigger input timing is the time when the device station detects the signal.



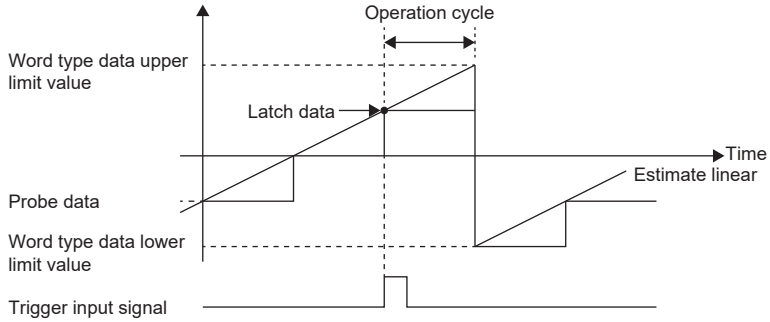
*: For how to set high-accuracy input for selected external signals, refer to the following.

User's manual for the controller used

*: For the accuracy of signal detection time, refer to the manual of the device to be used.

• When not using an external signal high-accuracy input

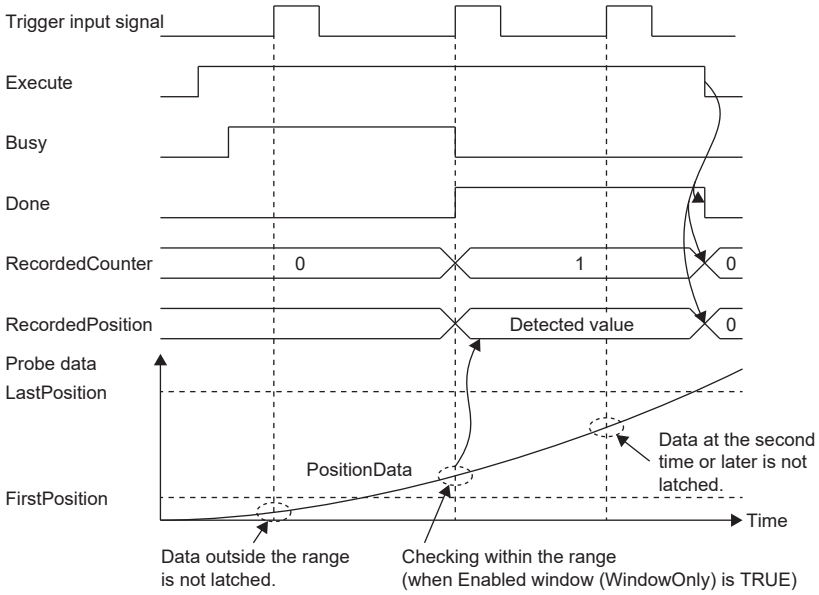
The trigger input timing is the operation cycle.



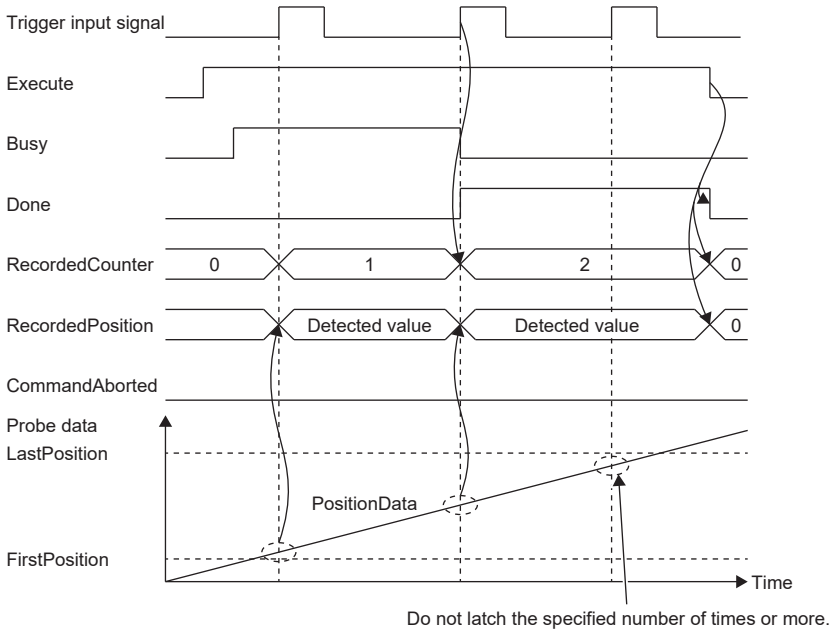
■Timing chart

- When the processing is completed successfully

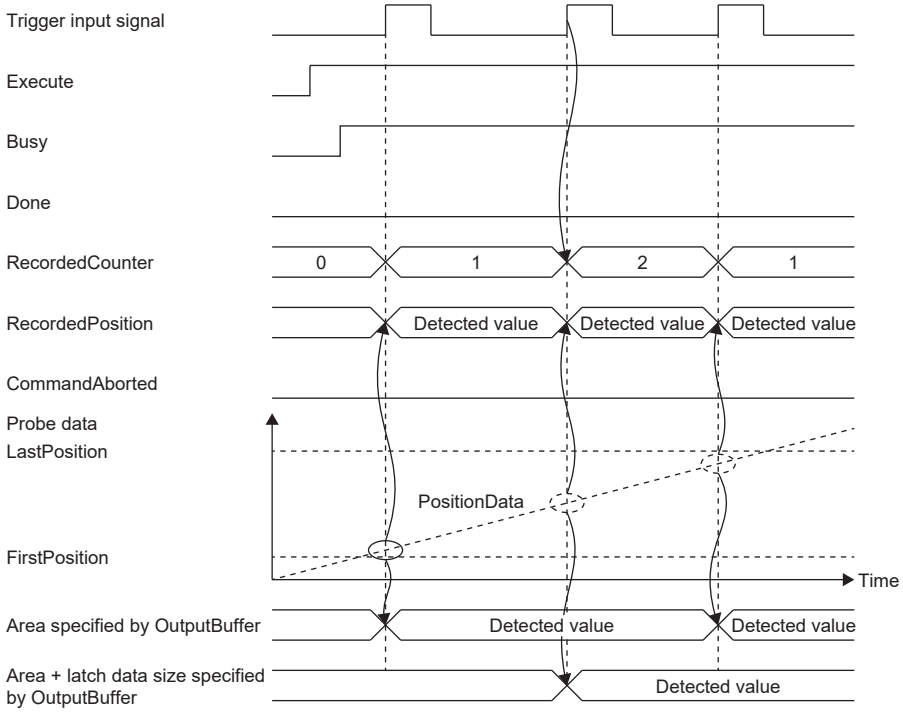
• When Latch mode (RecordMode) is "0: One shot mode (OneShot)



• When Latch mode (RecordMode) is "1: Frequency specified mode (RecordCount)"
When Number of latch times (RecordCount) is two



• When Latch mode (RecordMode) is "2: Ring buffer mode (RingBuffer)"
When Number of latch times (RecordCount) is two



• When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Trigger input signal (TriggerInput)

Trigger input signal is set with an MC_TRIGGER_REF structure. For the MC_TRIGGER_REF structure, refer to the following.

☞ Page 1442 MC_TRIGGER_REF

The setting range for Trigger input signal (Signal) of the MC_TRIGGER_REF structure set in Trigger input signal (TriggerInput) is shown below.

The type for Trigger signal (Signal) is SIGNAL_SELECT structure. For the SIGNAL_SELECT structure, refer to the following.

☞ Page 1447 SIGNAL_SELECT (Signal select)

Structures	Variable name	Type	Setting range
SIGNAL_SELECT (Signal select)	Source (Signal)	TARGET_REF ☞ Page 1444 TARGET_REF (Input signal)	■Type • BOOL ■Data category • [OBJ]*1*2 • [LINK] • [VAR] • [DEV] • [CONST]
	Detection (Signal detection method)	INT (MC_SIGNAL_LOGIC) ☞ Page 1478 MC_SIGNAL_LOGIC	• 2: Detection at FALSE → TRUE (rising edge) (RisingEdge) • 3: Detection at TRUE → FALSE (falling edge) (FallingEdge) • 4: Detection at rising edge/falling edge (BothEdges)
	CompensationTime (Compensation time)	LREAL	-5.0 to 5.0 [s]
	FilterTime (Filter time)	LREAL	0.0 to 5.0 [s]

*1 When the target modification is omitted, the object of Axis Information (Axis) axis is referred. However, if the Axis Information (Axis) axis does not have a station address, or if a station address has not been set, External signal selection incorrect (error code: 1BD6H) occurs.
When the station address for the target modification has been specified, the Axis Information (Axis) is ignored.
When the referred station has the Axis Emulate function enabled and also has a set station address, the emulating object will be referred.
To use external signal high-accuracy input, set Detection (Signal detection method) to 4: Detection at rising edge/falling edge (BothEdges).

*2 External signal high-accuracy input can be used. For details, refer to the following.
📖 User's manual for the controller used

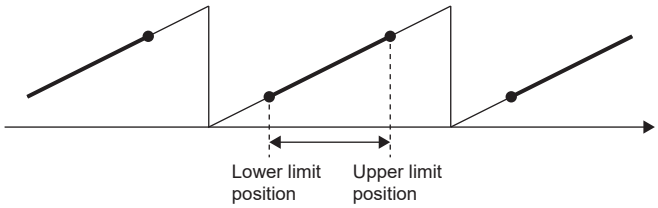
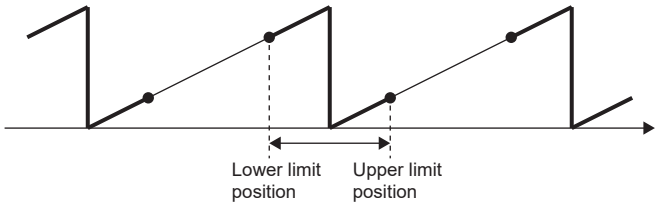
When a signal that cannot be used for Signal name (TriggerInput.Signal.Source.Target) is used, External signal selection string incorrect (error code: 1BC5H), External signal selection incorrect (error code: 1BD6H or 1BD7H), or Add-on Signal IO internal error (error code: 1BD8H) is output.

Point

When using an external signal high-accuracy input, set the Filter time (TriggerInput.Signal.FilterTime) to "0". If set to a value other than "0", the signal detection time will not be correctly obtained. If the device has an input filter, use this device filter. If the device does not have an input filter, use the Window enabled (WindowOnly), Lower limit position (FirstPosition), and Upper limit position (LastPosition) of Touch Probe Enabled (MC_TouchProbe) together to prevent a false detection of the trigger signal due to noise.

■Enabled window (WindowOnly)/Lower limit position (FirstPosition)/Upper limit position (LastPosition)

When Enabled window (WindowOnly) is valid (TRUE), triggers are detected only within the range between Lower limit position (FirstPosition) and Upper limit position (LastPosition) of the probe data. When it is invalid (FALSE), the touch probe processing is executed for the entire range.

Lower limit position (FirstPosition)/Upper limit position (LastPosition) relation	Touch probe processing
Lower limit position (FirstPosition) ≤ Upper limit position (LastPosition)	The touch probe processing is executed when the probe data is equal to or more than Lower limit position (FirstPosition) and equal to or less than Upper limit position (LastPosition). 
Lower limit position (FirstPosition) > Upper limit position (LastPosition)	The touch probe processing is executed when the probe data is equal to or less than Upper limit position (LastPosition) or equal to or more than Lower limit position (FirstPosition). 

■Probe data (ProbeData)

The data to latch is set with a TARGET_REF structure. For the TARGET_REF structure, refer to the following.

☞ Page 1444 TARGET_REF (Input signal)

The settable data categories are shown below.

Data classification	Description
[AXIS]	The axis data of Axis information (Axis) is referred. The target modification (such as @Position) needs to be set.
[VAR]	Axis information (Axis) is ignored.
[DEV]	
[OBJ] (which has been mapped to cyclic data)	• When the station address for the target modification has been specified, the Axis Information (Axis) is ignored. • When the target modification is omitted, the object of Axis Information (Axis) axis is referred. However, if the Axis Information (Axis) axis does not have a station address, or if a station address has not been set, an error will occur. • When the referred station has the Axis Emulate function enabled and also has a set station address, the emulating object will be referred.

When a signal that cannot be used for Signal name (TARGET_REF.Target) is used, External signal selection string incorrect (error code: 1BC5H), External signal selection incorrect (error code: 1BD6H or 1BD7H), or Add-on Signal IO internal error (error code: 1BD8H) is output.

- The data is updated as the ring counter. The upper limit value and lower limit value of the ring counter differ depending on the data classification and data type of Probe data (ProbeData).

Data classification	Description
[AXIS]	■When the target modification is set to "@Position" • The ring counter upper limit value and ring counter lower limit value of the axis are used. ■When the target modification is set to "@CumulativePos" • The positioning range upper limit value and positioning range lower limit value of the axis are used.
[VAR]	The maximum value and minimum value of the data type are used. Example: When [VAR] is set to a Word type label. • Ring counter upper limit value: 65535 • Ring counter lower limit value: 0
[DEV]	
[OBJ]	

Restriction

Ensure that the change amount per operation cycle satisfies the following formula. Failing to do so may cause the actual change amount of the probe data and the latched change amount to not match.

$$\text{Variation amount of one operation cycle} < \frac{|\,(\text{ring counter maximum value}) - (\text{ring counter minimum value}) + 1}{2}$$

■Compensation time (CompensationTime)

Compensates for the delay time of the touch probe processing. Set this variable to compensate for delays specific to the touch probe processing separately from the compensation time of the trigger input signal. To compensate for delays, set a positive number. To compensate for the progression, set a negative number.
The total compensation time = Compensation time (CompensationTime) - Compensation time (TriggerInput.Signal.CompensationTime)

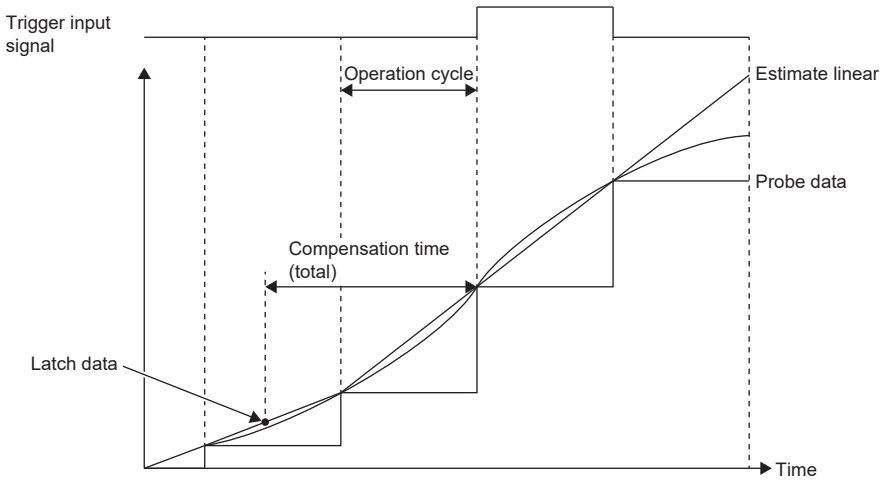
Point

When compensating for the delay time, the past value is used for latch data estimate calculation. As such, the maximum accuracy is reached when the compensation time is passed after Executing (Busy) becomes TRUE. In addition, memory is consumed as shown below depending on the compensation time and the operation cycle.

- Memory consumption = 24 bytes × (total compensation time ÷ operation cycle)

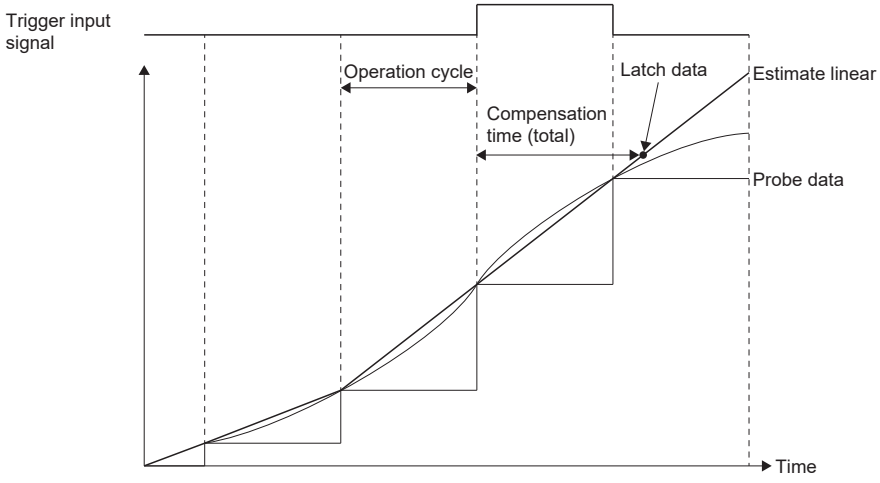
- When compensating for delays (when the total compensation time is a positive number)

The estimate is calculated based on the operation cycle data after latching and before the compensation time is passed.



- When compensating for the progression (when the total compensation time is a negative number)

The estimate is calculated based on the operation cycle data at latching.



■Latch mode (RecordMode)

The latch method for the data is set. The latch methods are shown below.

Setting value	Description
0: One shot mode (OneShot)	Latch data is stored one time only.
1: Frequency specified mode (RecordCount)	• Latch data is stored the number of times equal to the setting value of Number of latch times (RecordCount). • Starting from the storage location specified in Storage location of latch data (OutputBuffer), stores latch data in an area shifted for the latch data size each latch. • To stop the touch probe function before latching completes for the number of times set in Number of latch times (RecordCount), execute MC_AbortTrigger (Touch Probe Disabled).
2: Ring buffer mode (RingBuffer)	• Latch data is stored to the ring buffer the number of times equal to the setting value of Number of latch times (RecordCount). The latch operation is always performed. • Starting from the storage location specified in Storage location of latch data (OutputBuffer), shifts for the latch data size each latch and stores latch data to the ring buffer of Number of latch times (RecordCount). • To stop the touch probe function before latching completes for the number of times set in Number of latch times (RecordCount), execute MC_AbortTrigger (Touch Probe Disabled).

- Based on the setting of Latch mode (RecordMode), the following values are stored in Number of latch times (RecordedCounter) or Latch position (RecordedPosition).

Latch mode (RecordMode)	Number of latch times (RecordedCounter)	Latch position (RecordedPosition)
0: One shot mode (OneShot)	Becomes "1" at latch operation.	Latch data is stored one time only.
1: Frequency specified mode (RecordCount)	Adds "+1" at each latch operation.	The latest latch data is stored at each latch operation.
2: Ring buffer mode (RingBuffer)	Adds "+1" at each latch operation. Returns to "0" when the value of Number of latch times (RecordCount) is exceeded.	The latest latch data is stored at each latch operation.

■Number of latch times (RecordCount)

Based on the setting of Latch mode (RecordMode), the following values are stored in Number of latch times (RecordCount).

Latch mode (RecordMode)	Number of latch times (RecordCount)
0: One shot mode (OneShot)	Ignored.
1: Frequency specified mode (RecordCount)	The number of latch times is set. When the setting is out of range, Latch Count Incorrect (error code: 1B8AH) occurs.
2: Ring buffer mode (RingBuffer)	The number of ring buffer times is set. When the setting is out of range, Latch Count Incorrect (error code: 1B8AH) occurs.

■Storage location of latch data (OutputBuffer)

The storage location of the latched data is set with a TARGET_REF structure. For the TARGET_REF structure, refer to the following.

☞ Page 1444 TARGET_REF (Input signal)

- The usable data classifications are [VAR] and [DEV].
- As latched data will be stored as LREAL type, when setting non-LREAL type data in Target (Target) of the TARGET_REF structure, set "(LREAL)" explicitly to the data type.
- The storage location of the latched data is shown below.

Latch mode (RecordMode)	Storage location
0: One shot mode (OneShot)	The area specified in Storage location of latch data (OutputBuffer).
1: Frequency specified mode (RecordCount)	Following area Area specified in Storage location of latch data (OutputBuffer) + Number of latch times (RecordedCounter) × latch data size
2: Ring buffer mode (RingBuffer)	

- Storage location of latch data (OutputBuffer) can be omitted. When omitted, the latch data is only stored in Latch position (RecordedPosition).
- A setting example in which Storage location of latch data (OutputBuffer) is set in Target (Target) of the TARGET_REF structure is shown below.

Ex.

When the following settings are used

Example	Description	Setting
1	• When Latch mode (RecordMode) is "0: One shot mode (OneShot)" • When Number of latch times (RecordCount) is "1" while Latch mode (RecordMode) is "1: Frequency specified mode (RecordCount)" or "2: Ring buffer mode (RingBuffer)".	• [VAR]OutputBufferData1*1
2	• When Number of latch times (RecordCount) is "100" while Latch mode (RecordMode) is "1: Frequency specified mode (RecordCount)" or "2: Ring buffer mode (RingBuffer)".	• [VAR]OutputBufferData2*2 • [VAR](LREAL)OutputBufferData3*3

*1 "OutputBufferData1" is an LREAL type instance

*2 "OutputBufferData2" is an LREAL[0..99] type instance.

*3 "OutputBufferData3" is a WORD[0..399] type instance.

- When the number of elements set in Storage location of latch data (OutputBuffer) is less than the value of Number of latch times (RecordCount), Latch Data Storage Shortage (error code: 1BBFH) occurs. Latch Data Storage Shortage (error code: 1BBFH) also occurs when access outside the device or label range is made during latch operation.
- The relationship between the setting value and operation of each item is shown below.

Latch mode (RecordMode)	Number of latch times (RecordCount)	Storage location of latch data (OutputBuffer)	Detection operation	Data storage location
0: One shot mode (OneShot)	Within the setting range/ outside the setting range	Omit	One time	Latch position (RecordedPosition)
		Set		Latch position (RecordedPosition)/buffer
1: Frequency specified mode (RecordCount)	Outside the setting range	Omit/Set	Error	—
	Within the setting range	Omit	Setting number	Latch position (RecordedPosition)
		Set		Latch position (RecordedPosition)/buffer
2: Ring buffer mode (RingBuffer)	Outside the setting range	Omit/Set	Error	—
	Within the setting range	Omit	Always	Latch position (RecordedPosition)
		Set		Latch position (RecordedPosition)/buffer

Precautions

- The touch probe is controlled (detection and latch of trigger input signals) in the following operation cycles, regardless of the FB call task (scan/fixed cycle/event). The detection cycle of trigger input signals differs depending on the data category set in TriggerInput.

Category	Period
[VAR]	First operation cycle
[DEV]	
[CONST]	
[OBJ]	Communication cycle of the target station
[LINK]	
[VAR] With a network label specified	

The data set in the probe data (ProbeData) is latched at the following cycles.

Category	Period
[VAR]	First operation cycle
[DEV]	
[OBJ]	Communication cycle of the target station
[LINK]	
[VAR] With a network label specified	
[AXIS]	Control operation cycle of the target axis

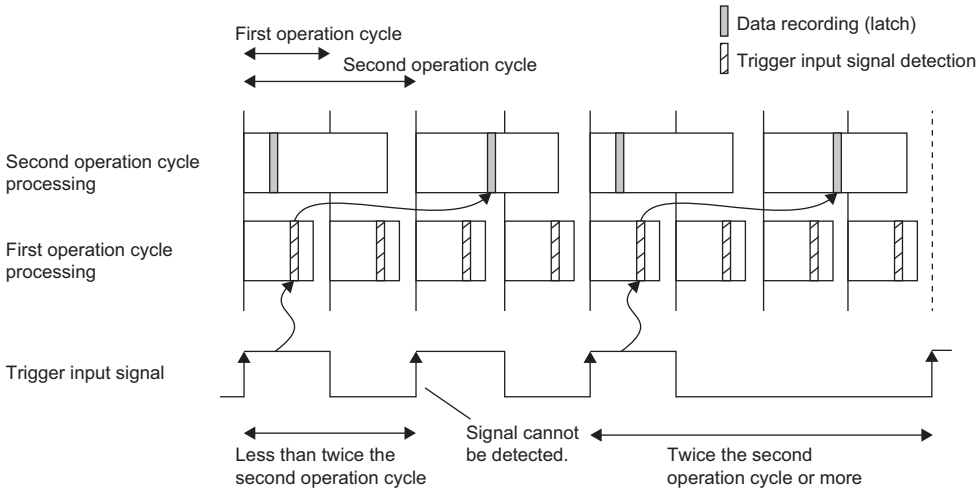
- MC_TouchProbe (Touch Probe Enabled) is executed even if the target axis is unconnected or an error occurs.
- The time for each TRUE and FALSE of the trigger input signal must be longer than the following. If the time is short, the signal cannot be detected correctly.

External signal high-accuracy input	The time for each TRUE and FALSE of the trigger input signal
Use	• Device station: Device station filter time • Input module: Input module input response time
Not use	Trigger input signal detection cycle

- The interval for trigger input signals must be at least twice as long as the longer of either the trigger input signal detection cycle or latch cycle. If the interval is shorter than that, the signal cannot be detected correctly. If multiple trigger input signals are input before latching, the first input signal is used.

Ex.

If the detection cycle of the trigger input signal is the first operation cycle and the latch cycle is the second operation cycle



Program example

The following is a program example. The touch probe start (bTouchProbe) is set to TRUE, and the actual position of Axis 1 (Axis0001) is estimated and calculated on each rising edge of the touch probe signal (G_bTouchProbeSignal) in the motion system. After the actual position is stored in Latch position (leRecordedPosition), the Touch Probe Disabled start (bAbortTrigger) is set to TRUE and the latch operation is stopped.

■Axis

Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■I/O data

Data name	Data category	Data type	Source category	Source	Signal detection method
SignalData0001	Trigger signal	MC_TRIGGER_REF	Global label	G_bTouchProbeSignal	Detection at FALSE → TRUE (rising edge) (RisingEdge)

■Global label used

Label name	Data type	Comment
G_bTouchProbeSignal	Bit	Touch probe signal
G_leLatchPos_array	Double-precision real number (0..1)	Variable for storing latch position

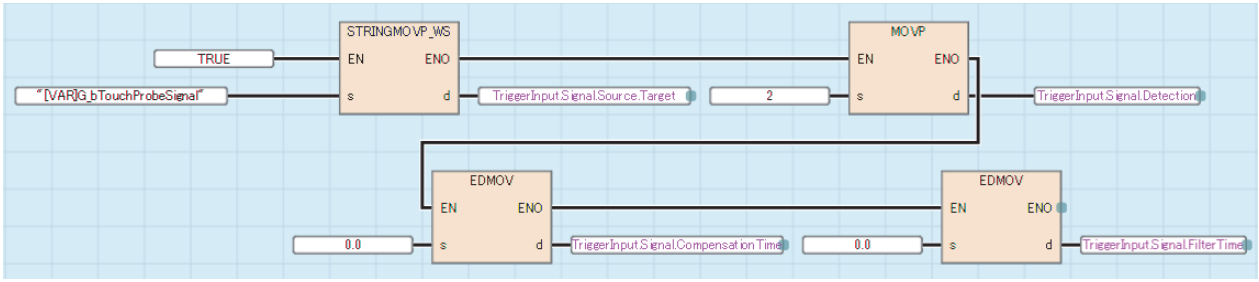
■Labels used

Label name	Data type	Comment
MC_TouchProbe_1	MC_TouchProbe	Touch Probe Enabled FB
bTouchProbe	Bit	Touch probe start
leFirstPosition	Double-precision real number	Lower limit position
leLastPosition	Double-precision real number	Upper limit position
leCompensationTime	Double-precision real number	Compensation time
ProbeData	TARGET_REF	Probe data
uwRecordCount	Word [unsigned]/bit string [16-bit]	Number of latch times
OutputBuffer	TARGET_REF	Storage location of latch data
bTPDone	Bit	Touch Probe Enabled Completion
bTPBusy	Bit	Touch Probe Enabled Executing
bTPCommandAborted	Bit	Touch Probe Enabled Abortion of execution
bTPError	Bit	Touch Probe Enabled Error
uwTPErrorID	Word [unsigned]/bit string [16-bit]	Touch Probe Enabled Error code
lePositionData	Double-precision real number	Current value monitor data
leRecordedPosition	Double-precision real number	Latch position
uwRecordedCounter	Word [unsigned]/bit string [16-bit]	Number of latch times
uwTouchProbeID	Word [unsigned]/bit string [16-bit]	Probe ID
MC_AbortTrigger_1	MC_AbortTrigger	Touch Probe Disabled FB
bAbortTrigger	Bit	Touch Probe Disabled start
bATDone	Bit	Touch Probe Disabled Completion
bATBusy	Bit	Touch Probe Disabled processing
bATCommandAborted	Bit	Touch Probe Disabled Abortion of execution
bATError	Bit	Touch Probe Disabled Error
uwATErrorID	Word [unsigned]/bit string [16-bit]	Touch Probe Disabled Error code
TriggerInput*1	MC_TRIGGER_REF	Trigger input signal

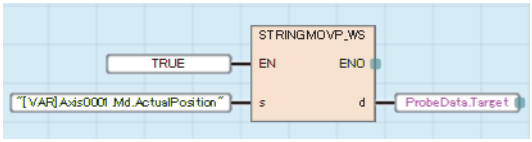
*1 Only used in FBD/LD programs. (Required to use the I/O data in FBD/LD programs.)

■FBD/LD program

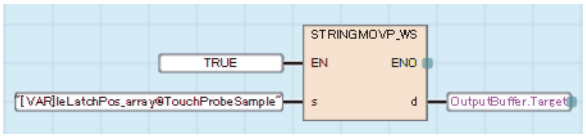
- Trigger input signal (I/O data)



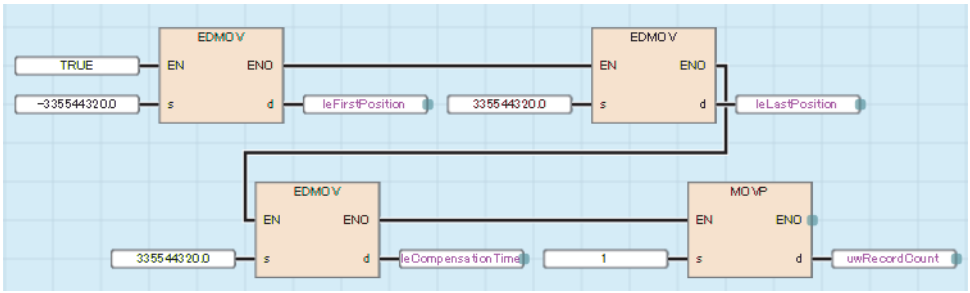
- Data for latch target



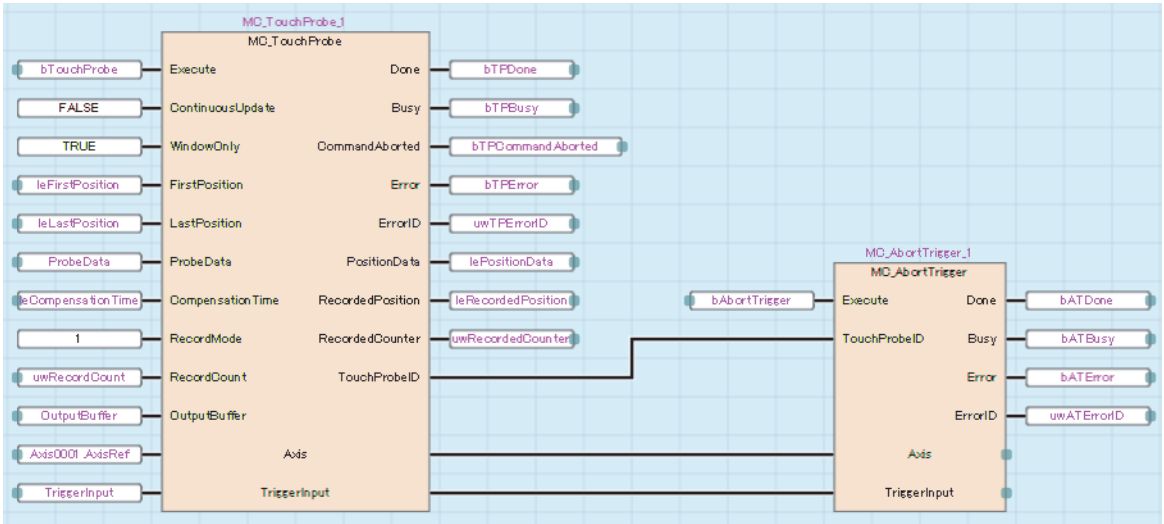
- Touch probe processing



- Data setting for touch probe



- Touch Probe Enabled/Disabled



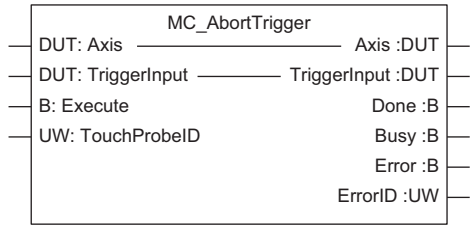
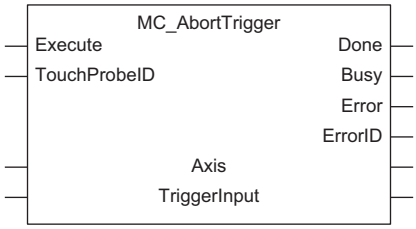
■ST program

```
//-----Data for latch target-----
ProbeData.Target:= "[VAR]Axis0001.Md.ActualPosition";

//-----Touch probe processing-----
OutputBuffer.Target:= "[VAR]G_leLatchPos_array";

//-----Data setting for touch probe-----
leFirstPosition:= -335544320.0;
leLastPosition:= 335544320.0;
leCompensationTime:= 0.0;
uwRecordCount:= 1;

//-----Touch Probe Enabled-----
MC_TouchProbe_1(
   Axis:= Axis0001.AxisRef ,
   TriggerInput:= SignalData0001 ,
   Execute:= bTouchProbe ,
   ContinuousUpdate:= FALSE ,
   WindowOnly:= TRUE ,
   FirstPosition:= leFirstPosition ,
   LastPosition:= leLastPosition ,
   ProbeData:= ProbeData ,
   CompensationTime:= leCompensationTime ,
   RecordMode:= MC_RECORD_MODE__RecordCount ,
   RecordCount:= uwRecordCount ,
   OutputBuffer:= OutputBuffer ,
   Done=> bTPDone ,
   Busy=> bTPBusy ,
   CommandAborted=> bTPCommandAborted ,
   Error=> bTPError ,
   ErrorID=> uwTPErrorID ,
   PositionData=> lePositionData ,
   RecordedPosition=> leRecordedPosition ,
   RecordedCounter=> uwRecordedCounter ,
   TouchProbeID=> uwTouchProbeID
);

//-----Touch Probe Disabled-----
MC_AbortTrigger_1(
   Axis:= Axis0001.AxisRef ,
   TriggerInput:= SignalData0001 ,
   Execute:= bAbortTrigger ,
   TouchProbeID:= MC_TouchProbe_1.TouchProbeID ,
   Done=> bATDone ,
   Busy=> bATBusy ,
   Error=> bATError ,
   ErrorID=> uwATErrorID
);
```

41.12 Touch Probe Disabled

MC_AbortTrigger

This FB disables the latch that is being executed.

Ladder	FBD/LD	ST
		MC_AbortTrigger(Axis:= ?AXIS_REF? , TriggerInput:= ?MC_TRIGGER_REF? , Execute:= ?BOOL? , TouchProbeID:= ?WORD? , Done=> ?BOOL? , Busy=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)
TriggerInput	Trigger input signal	MC_TRIGGER_REF	At start	—	Omission possible	The trigger input signal is set. The setting cannot be configured with this FB. If the setting is configured, it is ignored for operation.

■Input variables

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Start	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_AbortTrigger (Touch Probe Disabled) is executed.
TouchProbeID	Touch probe ID	WORD(UINT)	At start	0 to 65535	0	The specific ID of the disable target touch probe is set. The touch probe ID to be stored in Probe ID (TouchProbeID) that was started by Execution command (Execute) of the disable target MC_TouchProbe (Touch Probe Enabled) is set.

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the control is completed.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_AbortTrigger (Touch Probe Disabled) is executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

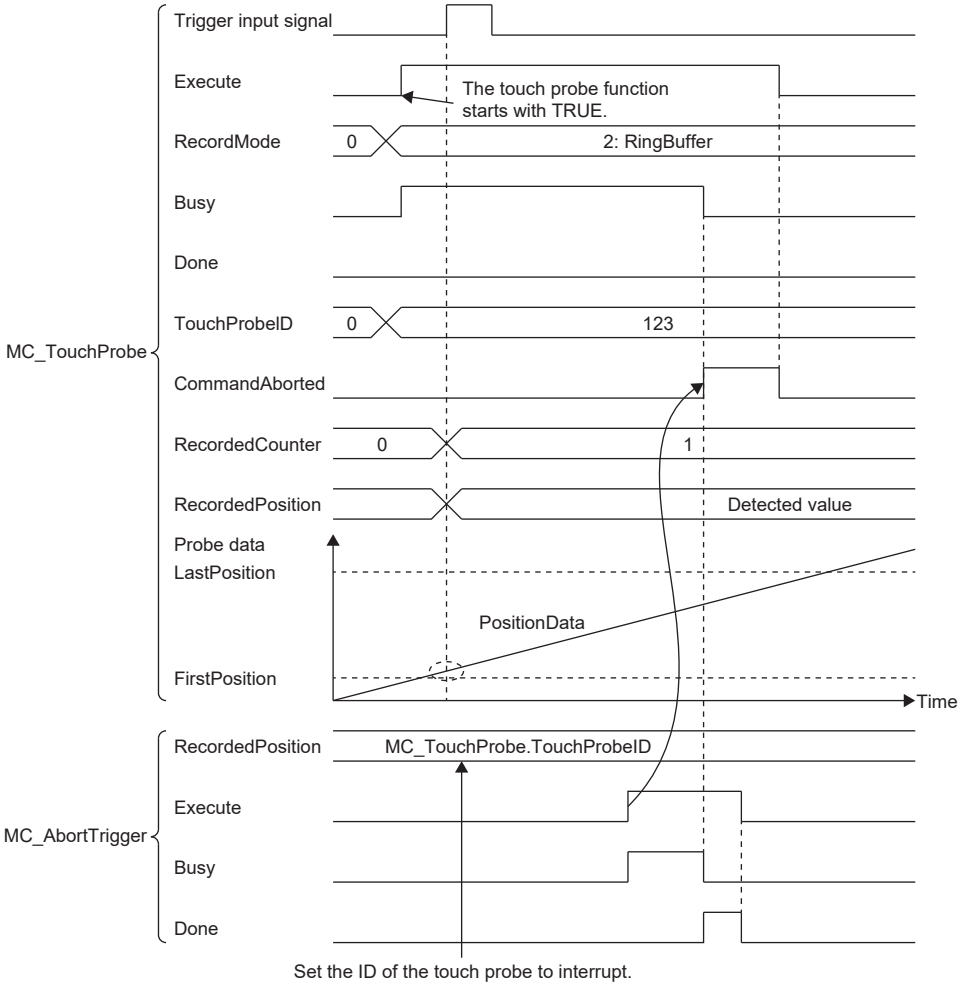
Processing details

- This FB disables the MC_TouchProbe (Touch Probe Enabled) set in Touch probe ID (TouchProbeID).
- When MC_AbortTrigger (Touch Probe Disabled) is executed, if the MC_TouchProbe (Touch Probe Enabled) set in Touch probe ID (TouchProbeID) is not in operation, Execution completion (Done) immediately becomes TRUE.
- Axis information (Axis) can be omitted because it is ignored.

■Timing chart

- When the processing is completed successfully

When Latch mode (RecordMode) is "2: Ring buffer mode (RingBuffer)"



- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

Program example

The following is a program example. The touch probe start (bTouchProbe) is set to TRUE, and the actual position of Axis 1 (Axis0001) is estimated and calculated on each rising edge of the touch probe signal (G_bTouchProbeSignal) in the motion system. After the actual position is stored in Latch position (leRecordedPosition), the Touch Probe Disabled start (bAbortTrigger) is set to TRUE and the latch operation is stopped.

■Axis

Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■I/O data

Data name	Data category	Data type	Source category	Source	Signal detection method
SignalData0001	Trigger signal	MC_TRIGGER_REF	Global label	G_bTouchProbeSignal	Detection at FALSE → TRUE (rising edge) (RisingEdge)

■Global label used

Label name	Data type	Comment
G_bTouchProbeSignal	Bit	Touch probe signal
G_leLatchPos_array	Double-precision real number (0..1)	Variable for storing latch position

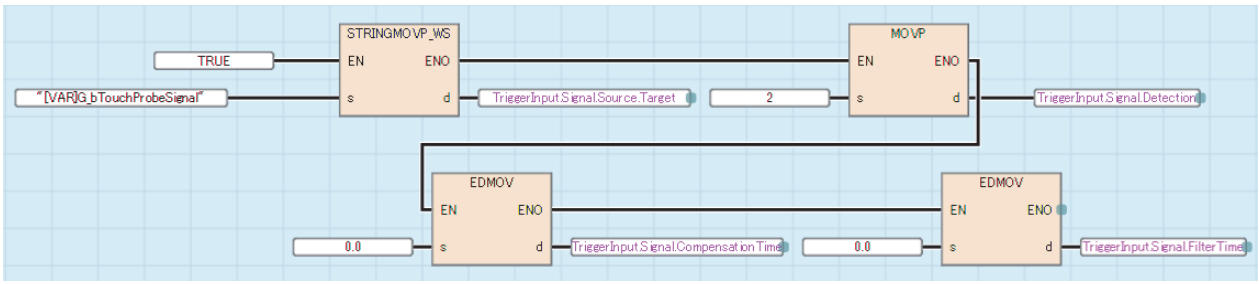
■Labels used

Label name	Data type	Comment
MC_TouchProbe_1	MC_TouchProbe	Touch Probe Enabled FB
bTouchProbe	Bit	Touch probe start
leFirstPosition	Double-precision real number	Lower limit position
leLastPosition	Double-precision real number	Upper limit position
leCompensationTime	Double-precision real number	Compensation time
ProbeData	TARGET_REF	Probe data
uwRecordCount	Word [unsigned]/bit string [16-bit]	Number of latch times
OutputBuffer	TARGET_REF	Storage location of latch data
bTPDone	Bit	Touch Probe Enabled Completion
bTPBusy	Bit	Touch Probe Enabled Executing
bTPCommandAborted	Bit	Touch Probe Enabled Abortion of execution
bTPError	Bit	Touch Probe Enabled Error
uwTPErrorID	Word [unsigned]/bit string [16-bit]	Touch Probe Enabled Error code
lePositionData	Double-precision real number	Current value monitor data
leRecordedPosition	Double-precision real number	Latch position
uwRecordedCounter	Word [unsigned]/bit string [16-bit]	Number of latch times
uwTouchProbeID	Word [unsigned]/bit string [16-bit]	Probe ID
MC_AbortTrigger_1	MC_AbortTrigger	Touch Probe Disabled FB
bAbortTrigger	Bit	Touch Probe Disabled start
bATDone	Bit	Touch Probe Disabled Completion
bATBusy	Bit	Touch Probe Disabled processing
bATCommandAborted	Bit	Touch Probe Disabled Abortion of execution
bATError	Bit	Touch Probe Disabled Error
uwATErrorID	Word [unsigned]/bit string [16-bit]	Touch Probe Disabled Error code
TriggerInput*1	MC_TRIGGER_REF	Trigger input signal

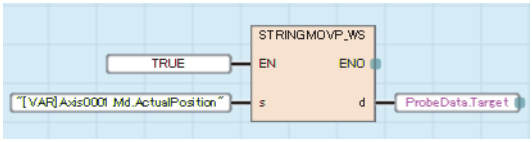
*1 Only used in FBD/LD programs. (Required to use the I/O data in FBD/LD programs.)

■FBD/LD program

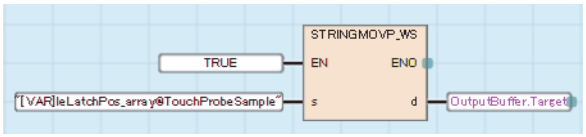
- Trigger input signal (I/O data)



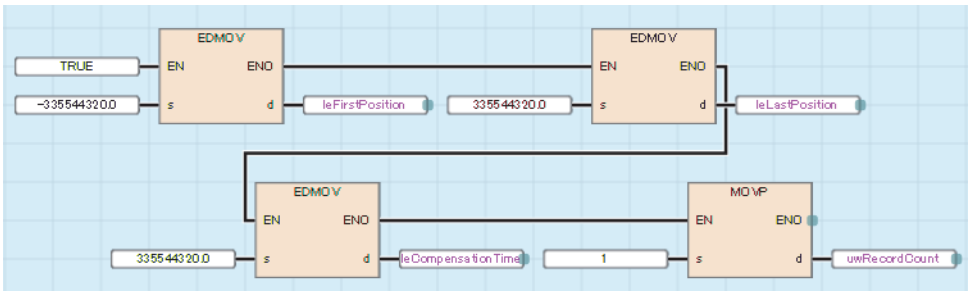
- Data for latch target



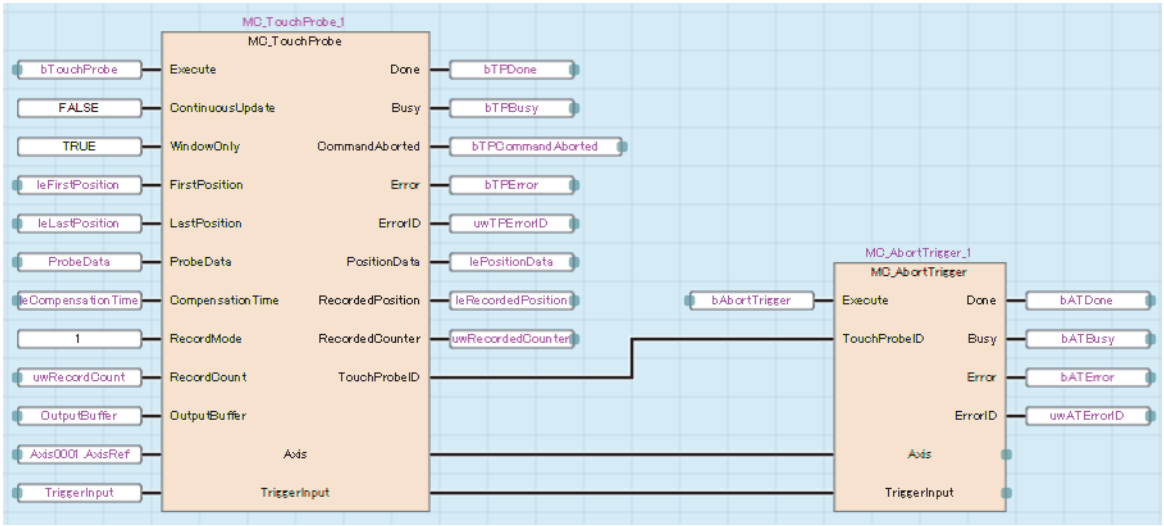
- Touch probe processing



- Data setting for touch probe



- Touch Probe Enabled/Disabled



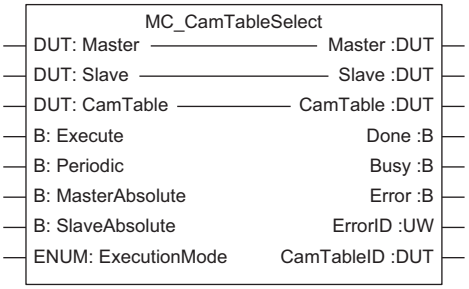
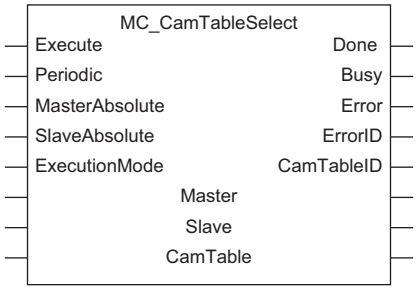
■ST program

```
//-----Data for latch target-----
ProbeData.Target:= "[VAR]Axis0001.Md.ActualPosition";

//-----Touch probe processing-----
OutputBuffer.Target:= "[VAR]G_leLatchPos_array";

//-----Data setting for touch probe-----
leFirstPosition:= -335544320.0;
leLastPosition:= 335544320.0;
leCompensationTime:= 0.0;
uwRecordCount:= 1;

//-----Touch Probe Enabled-----
MC_TouchProbe_1(
   Axis:= Axis0001.AxisRef ,
   TriggerInput:= SignalData0001 ,
   Execute:= bTouchProbe ,
   ContinuousUpdate:= FALSE ,
   WindowOnly:= TRUE ,
   FirstPosition:= leFirstPosition ,
   LastPosition:= leLastPosition ,
   ProbeData:= ProbeData ,
   CompensationTime:= leCompensationTime ,
   RecordMode:= MC_RECORD_MODE__RecordCount ,
   RecordCount:= uwRecordCount ,
   OutputBuffer:= OutputBuffer ,
   Done=> bTPDone ,
   Busy=> bTPBusy ,
   CommandAborted=> bTPCommandAborted ,
   Error=> bTPError ,
   ErrorID=> uwTPErrorID ,
   PositionData=> lePositionData ,
   RecordedPosition=> leRecordedPosition ,
   RecordedCounter=> uwRecordedCounter ,
   TouchProbeID=> uwTouchProbeID
);

//-----Touch Probe Disabled-----
MC_AbortTrigger_1(
   Axis:= Axis0001.AxisRef ,
   TriggerInput:= SignalData0001 ,
   Execute:= bAbortTrigger ,
   TouchProbeID:= MC_TouchProbe_1.TouchProbeID ,
   Done=> bATDone ,
   Busy=> bATBusy ,
   Error=> bATError ,
   ErrorID=> uwATErrorID
);
```

41.13 Cam Table Selection

MC_CamTableSelect

This FB stores the specified operation profile data (cam data) into the open area.

Ladder	FBD/LD	ST
		MC_CamTableSelect(Master:= ?AXIS_REF? , Slave:= ?AXIS_REF? , CamTable:= ?MC_CAM_REF? , Execute:= ?BOOL? , Periodic:= ?BOOL? , MasterAbsolute:= ?BOOL? , SlaveAbsolute:= ?BOOL? , ExecutionMode:= ?INT? , Done=> ?BOOL? , Busy=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD? , CamTableID=> ?MC_CAM_ID?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
Master	Master axis	AXIS_REF	At start	—	Omission possible	The setting value is ignored.
Slave	Slave axis	AXIS_REF	At start	—	Omission possible	The setting value is ignored.
CamTable	Cam table	MC_CAM_REF	At start	—	Omission not possible	The operation profile data (cam data) is set. When MC_CamTableSelect (Cam Table Selection) is executed, the setting of Profile ID (MC_CAM_ID.ProfileID) is not required. The setting value of PROFILE_DATA.ID (MC_CAM_REF.ProfileData.ID) is referred to. However, the setting of I/O No. (CamTable.ProfileData.ID.StartIO) is not required. For details, refer to the following. ☞ Page 1558 Cam table (CamTable)

■Input variables

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Start	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_CamTableSelect (Cam Table Selection) is executed.
Periodic	Periodic	BOOL	At start	TRUE, FALSE	FALSE	The operation to execute the operation profile data (cam data) is set. • FALSE: Non periodic • TRUE: Periodic For details, refer to the following. ☞ Page 1558 Periodic (Periodic)
MasterAbsolute	Master axis absolute coordinate	BOOL	At start	FALSE	FALSE	The coordinate of the master axis is set. • FALSE: Relative coordinate • TRUE: Setting not possible For details, refer to the following. ☞ Page 1559 Master axis absolute coordinate (MasterAbsolute) *: If set to "TRUE", Out of MasterAbsolute Range (error code: 1B94H) occurs.
SlaveAbsolute	Slave axis absolute coordinate	BOOL	At start	TRUE, FALSE	FALSE	The coordinate of the slave axis is set. • FALSE: Relative coordinate • TRUE: Absolute coordinate For details, refer to the following. ☞ Page 1559 Slave axis absolute coordinate (SlaveAbsolute)
ExecutionMode	Execution mode	INT (MC_EXECUTION_MODE)	At start	0, 1, 3	0	The execution timing of MC_CamTableSelect (Cam Table Selection) is set. • 0: Execute immediately (mcImmediately) • 1: Execute at completion (mcQueued) • 3: Execute speculatively (mcSpeculatively) For details, refer to the following. ☞ Page 1560 Execution mode (ExecutionMode)

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Completion	BOOL	FALSE	This variable becomes TRUE when the control is completed. The value differs depending on the status of Start (Execute) at operation completion. ■When Start (Execute) is TRUE This variable remains TRUE until Start (Execute) is set to FALSE. ■When Start (Execute) is FALSE This variable becomes TRUE only for one cycle.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_CamTableSelect (Cam Table Selection) is executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used
CamTableID	Cam table ID	MC_CAM_ID	0	The profile ID is output.

Processing details

- This FB opens the operation profile data (cam data) file set in Operation profile data storage location (Location) of the PROFILE_DATA structure to the open area by using MC_CamTableSelect (Cam Table Selection).
- Data opened in the open area is assigned the Profile ID (Number) of the PROFILE_ID structure.

Point

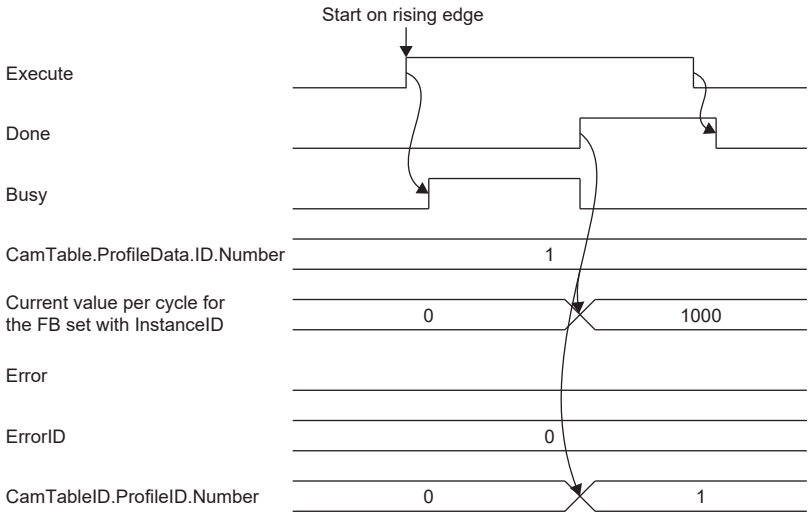
- For details on creating operation profile data (cam data), refer to the following.

User's manual for the controller used

- MC_CamTableSelect (Cam Table Selection) is performed with motion service processing. It may take time to open depending on the processing details and the operation profile data.
- When Profile_ID (ID) of the PROFILE_DATA structure, Periodic (Periodic), Master axis absolute coordinate (MasterAbsolute), and Slave axis absolute coordinate (SlaveAbsolute) are present as input variables for MC_CamTableSelect (Cam Table Selection), the file setting values are ignored and the open is performed while referencing the input variable settings. (The setting values of the file will not be overwritten.)

■Timing chart

- When the processing is completed successfully



- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Cam table (CamTable)

The operation profile data is set with an MC_CAM_REF type PROFILE_DATA structure. For the PROFILE_DATA structure, refer to the following.

☞ Page 1447 PROFILE_DATA

References operation profile data storage location (Location) of the PROFILE_DATA structure and performs open for the operation profile data that File name (FileName) and Folder specification (Path) indicate.

Profile ID No. (Number) of the PROFILE_ID structure indicated by Profile ID (ID) of the PROFILE_DATA structure is assigned in the open area.

The operation changes depending on the value set in Profile ID No. (Number).

Profile ID No. (Number)	Operation
When set to "0"	An available ID is automatically assigned and stored in Profile ID No. (Number). If an MC_CamTableSelect (Cam Table Selection) with Profile ID No. (Number) set to "0" is executed multiple times to the same operation profile data, the data will be opened in multiple different open areas. When no ID is available, Insufficient Operation Profile Data ID (error code: 1BA0H) will occur and open processing will not be performed.
When set to "1 to 60000"	The set ID is opened without changing the ID. If specifying an ID that has already been opened, the open data will be overwritten.

The relationship between the value of Profile ID No. (Number) at open and the open location area is shown below.

Profile ID No. (Number) value		Open location area
Before executing MC_CamTableSelect (Cam Table Selection)	After executing MC_CamTableSelect (Cam Table Selection)	
0	1 to 60000 is stored.	New area
1 to 60000		Open data is overwritten if already existing. When open data does not exist, a new open area is assigned.

- For the function specific type, the same value as PROFILE_DATA.ID (such as MC_CAM_REF.ProfileData.ID) is stored in the function specific type profile ID (such as MC_CAM_ID.ProfileID). For input variables of operation profile data control FBs (such as MC_CamIn), set a function specific type profile ID.

For details on profile ID types, refer to the following.

📖 User's manual for the controller used

- When executing an open FB, MC_CAM_REF.CamID.ProfileID does not need to be set. The set value for MC_CAM_REF.ProfileData.ID is referred to.

For the list of open FBs, refer to the following.

📖 User's manual for the controller used

- If the operation profile data set in Operation profile data storage location (Location) does not exist, No Operation Profile Data (error code: 1B8EH) occurs and the open processing is not performed.
- If the operation profile data set in Operation profile data storage location (Location) is broken or there is a problem in the format, Operation Profile Data Error (error code: 1B91H) occurs and the open processing is not performed.
- If the operation profile data setting value specified in Operation profile data storage location (Location) is outside the range, Operation Profile Data Incorrect (error code: 1B8FH) occurs and the open processing is not performed.

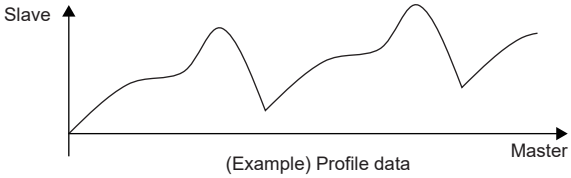
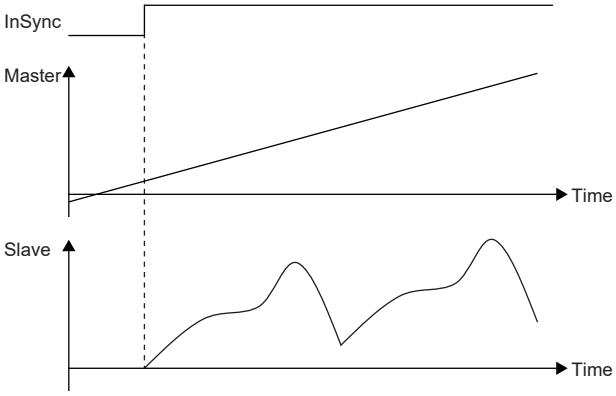
■Periodic (Periodic)

The operation performed by Periodic (Periodic) at operation profile data control execution is set.

Setting value	Description
FALSE (Non periodic)	The control ends when it executes until the end point of the operation profile data.
TRUE (Periodic)	Operation profile data execution is repeated continuously.

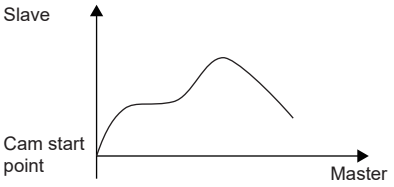
■Master axis absolute coordinate (MasterAbsolute)

The operation performed by Master axis absolute coordinate (MasterAbsolute) at operation profile data control execution is set. (When the input unit is time specification, it is ignored.)

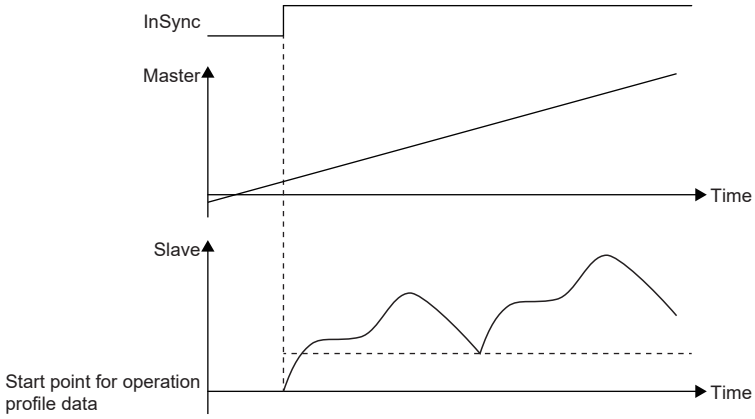
Setting value	Description
FALSE (Relative coordinate)	The current Master axis (Master) value becomes the start point of the operation profile data. When In synchronization (InSync) becomes TRUE, Slave axis (Slave) is operated according to the relative movement amount of Master axis (Master).  (Example) Profile data 
TRUE	TRUE cannot be set. If set to TRUE, Out of MasterAbsolute Range (error code: 1B94H) occurs.

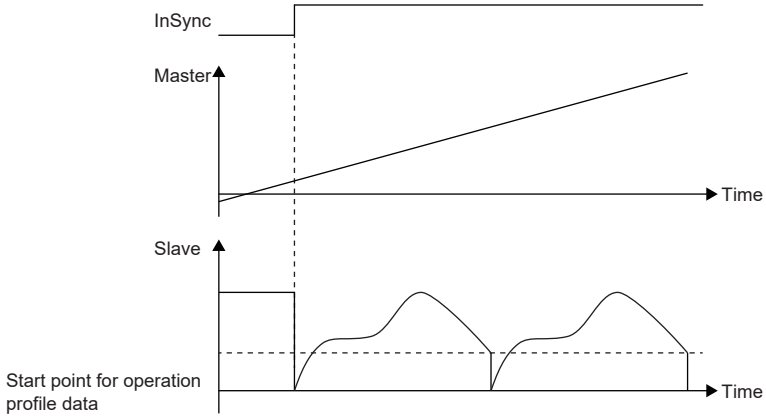
■Slave axis absolute coordinate (SlaveAbsolute)

The operation performed by Slave axis absolute coordinate (SlaveAbsolute) at operation profile data control execution is set.



<Example> Start point for operation profile data

Setting value	Description
FALSE (Relative coordinate)	When In synchronization (InSync) becomes TRUE, Slave axis (Slave) is operated based on the current value. 

Setting value	Description
TRUE (Absolute coordinate)	Operation is performed so that Slave axis (Slave) at the time that In synchronization (InSync) becomes TRUE is always the start point at the start of each cycle of the operation profile data. When there is a limit on the value of Master axis (Master) of the operation profile data, such as the cam data length per cycle, and the start point and end point of the operation profile data are different when Periodic (Periodic) is "TRUE: Periodic", the command is output in one operation cycle to return to the first Slave axis (Slave) at the start of the next cycle. 

■Execution mode (ExecutionMode)

This variable sets the timing for executing MC_CamTableSelect (Cam Table Selection).

Setting value	Description
0: Execute immediately (mcImmediately)	Immediately writes the contents of the open area. The writing may affect the control in execution. Note that if the operation profile data format and resolution do not match when performing open during FB execution, Operation Profile Data Being Operated (error code: 1B90H) occurs.
1: Execute at completion (mcQueued)	Opens after waiting for the FB being executed to complete. (Done = TRUE) When multiple FBs are waiting, the FBs will be executed in order of task priority. FBs of the same priority are executed according to the start order.
3: Execute speculatively (mcSpeculatively)	Operation profile data being operated (error code: 1B90H) occurs and the open area is not changed.

Point

When no FB is in execution, the FB is executed immediately regardless of the setting of Execution mode (ExecutionMode).
When FBs in execution are accessing the open area, Executing (Busy) of each FB accessing the open area becomes TRUE. Use Execution mode (ExecutionMode) for the interlock as needed.

Precautions

- When rewriting the open area during control, some waveform patterns may cause the stroke value may change rapidly and apply shock to the machine. Create operation profile data so that the waveform does not become discontinuous before or after the stroke value changes.

Program example

The following is a program example. The cam table select command (bCamTableSelect) is set to TRUE, and the operation profile data (Cam data 1) is opened to the open area and a profile ID is assigned.

■Axis

Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Operation profile data

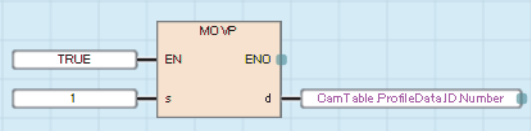
Profile ID	Label name	Data type	Comment
1	ProfileData0001	MC_CAM_REF	Cam data 1

■Labels used

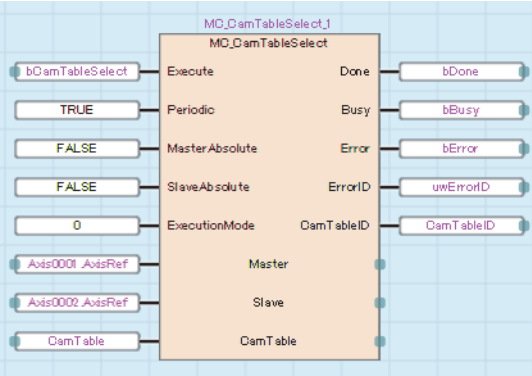
Label name	Data type	Comment
MC_CamTableSelect_1	MC_CamTableSelect	Cam Table Selection FB
CamTable	MC_CAM_REF	Cam table
bCamTableSelect	Bit	Cam table select command
bDone	Bit	Execution completion
bBusy	Bit	Executing
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code
CamTableID	MC_CAM_ID	Cam table ID

■FBD/LD program

- Operation profile data (cam data) settings



- Cam Table Selection



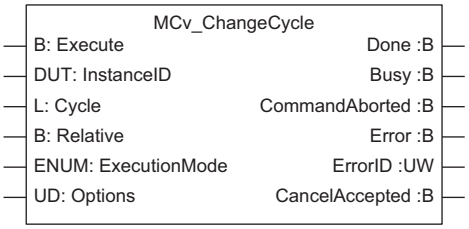
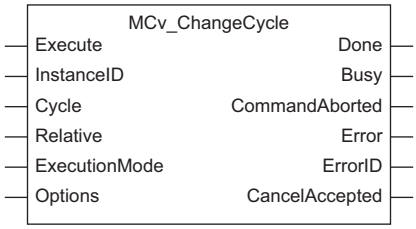
■ST program

```
//-----Operation profile data (cam data) settings-----
CamTable.ProfileData:= ProfileData0001.ProfileData;

//-----Cam Table Selection-----
MC_CamTableSelect_1(
   CamTable:= CamTable ,
   Execute:= bCamTableSelect ,
   Periodic:= TRUE ,
   MasterAbsolute:= FALSE ,
   SlaveAbsolute:= FALSE ,
   ExecutionMode:= MC_EXECUTION_MODE__mcImmediately ,
   Done=> bDone ,
   Busy=> bBusy ,
   Error=> bError ,
   ErrorID=> uwErrorID ,
   CamTableID=> CamTableID
);
```

41.14 Current Value Change per Cycle

MCv_ChangeCycle

This FB changes the current value per cycle of the specified operation profile data control FB.

Ladder	FBD/LD	ST
		MCv_ChangeCycle(Execute:= ?BOOL? , InstanceID:= ?INSTANCE_ID? , Cycle:= ?LREAL(0..2)? , Relative:= ?BOOL? , ExecutionMode:= ?INT? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD? , CancelAccepted=> ?BOOL?);

Setting data

■Input variables

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Start	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_ChangeCycle (Current Value Change per Cycle) is executed.
InstanceID	Instance ID	INSTANCE_ID	At start	1 to 65535	0	The instance ID of the operation profile data control FB whose current value per cycle will be changed is set. The instance ID is valid until "PLC Ready (MotionSystem.Cd.SequenceReady)" changes from on to off. When MCv_ChangeCycle (Current Value Change per Cycle) is used, set the I/O number (StartIO).
Cycle	Current value per cycle	LREAL[0..2]	At start	—	0.0	The change target current value per cycle is set. • When Relative selection (Relative) is set to FALSE, the absolute position is set. • When Relative selection (Relative) is set to TRUE, the relative distance is set. For details, refer to the following. ☞ Page 1564 Current value per cycle (Cycle)
Relative	Relative selection	BOOL	At start	TRUE, FALSE	FALSE	The method used to change the current value per cycle is set. • FALSE: Absolute coordinate • TRUE: Relative distance For details, refer to the following. ☞ Page 1565 Relative Selection (Relative)
ExecutionMode	Execution mode	INT (MC_EXECUTION_MODE)	At start	0	0	The execution timing of MCv_ChangeCycle (Current Value Change per Cycle) is set. • 0: Execute immediately (mcImmediately) For details, refer to the following. ☞ Page 1566 Execution mode (ExecutionMode)

Input variable	Name	Data type	Input	Setting range	Default value	Description
Options	Options	DWORD(HEX)	At start	00000000H to 00010000H	00000000H	The function options for MCv_ChangeCycle (Current Value Change per Cycle) are set in the specified bit position. For details, refer to the following. ☞ Page 1567 Options (Options)

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Completion	BOOL	FALSE	This variable becomes TRUE when the control is completed. The value differs depending on the status of Start (Execute) at operation completion. ■When Start (Execute) is TRUE This variable remains TRUE until Start (Execute) is set to FALSE. ■When Start (Execute) is FALSE This variable becomes TRUE only for one cycle.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_ChangeCycle (Current Value Change per Cycle) is executed.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MCv_ChangeCycle (Current Value Change per Cycle) is aborted because of an error or multiple start.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. User's manual for the controller used
CancelAccepted	Cancel acceptance	BOOL	FALSE	This variable becomes TRUE when MCv_ChangeCycle (Current Value Change per Cycle) accepts the cancel.

Processing details

- The current value per cycle of the operation profile data control FB is changed to the value set at execution of control.
- Instance ID (InstanceID) of the public variable of an operation profile data control FB (such as MC_CamIn) is set in InstanceID (InstanceID).
- Upon performing the current value per cycle change, the public variable of the operation profile data control FB (such as MC_CamIn) changes as shown below.

Public variable	Updated value	Remark
Current value per cycle (InputPerCycle)	Value set in Current value per cycle (Cycle)	—
Reference value (Reference)	Stroke value equivalent to Current value per cycle (Cycle)	Reference value (Reference) is updated to fix Output value (OutputData).
Output value (OutputData)	No value updated	—

Point

Reference value (Reference) is updated to fix Output value (OutputData) when performing current value per cycle change. This occurs even when Slave axis absolute coordinate (SlaveAbsolute) of MC_CamTableSelect (Cam Table Selection) is set to TRUE (Absolute coordinate).

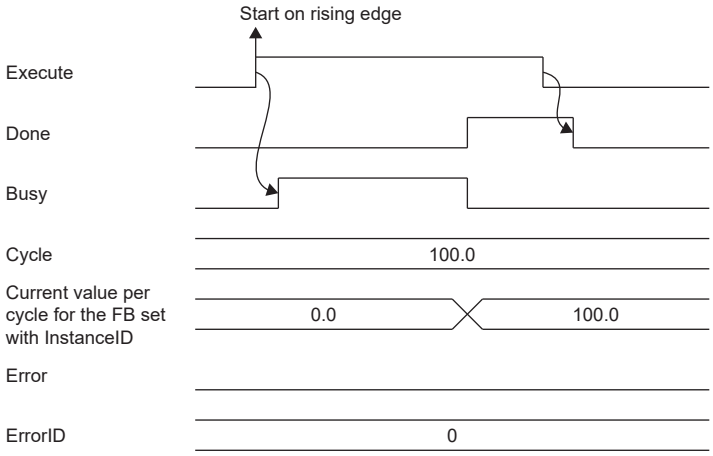
• The method for current value per cycle change is as follows.

Method	Description
MCv_ChangeCycle (Current Value Change per Cycle)	The value is changed to the one specified at control execution.

• For specification of the current value per cycle to be changed, "absolute specification" or "relative specification" can be selected by Relative position selection (Relative).
• When a current value change per cycle is performed for an operation profile data control FB other than MC_CamIn (Cam Operation Start), Current Value Per Cycle Change Unsupported (error code: 1BAFH) occurs.

■Timing chart

• When the processing is completed successfully



• When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Current value per cycle (Cycle)

The current value per cycle to be changed is set.

Values settable for Current value per cycle (Cycle) differ depending on the setting of Relative selection (Relative). Set the variable within the setting range shown below. If the variable is set outside the range, MCv_ChangeCycle Instruction Error (error code: 1BB9H (Cycle is out of range (detail code: 1))) occurs.

Current value per cycle (Cycle)	Setting value of Relative selection (Relative)	Setting range of Current value per cycle (Cycle)
Cycle[0]	FALSE (Absolute position)	0.0 ≤ Setting value < Current value per cycle
	TRUE (Relative distance)	- (Length per cycle) / 2 ≤ Setting value ≤ (Length per cycle) / 2
Cycle[1], Cycle[2]	FALSE (Absolute position)	0.0
	TRUE (Relative distance)	

■Relative Selection (Relative)

The value of Current value per cycle (Cycle) is changed as shown below based on the setting of Relative selection (Relative).

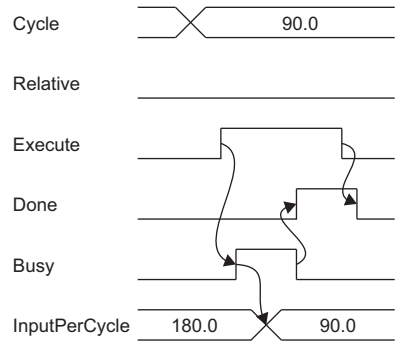
Setting value of Relative selection (Relative)	Description
FALSE (Absolute position)	Current value per cycle (InputPerCycle)*1 is changed to a value equal to Current value per cycle (Cycle) added to the presently set Current value per cycle (InputPerCycle)*1.
TRUE (Relative distance)	Current value per cycle (InputPerCycle)*1 is changed to the value of Current value per cycle (Cycle).

*1 Public variable of operation profile data control FBs (such as MC_CamIn)

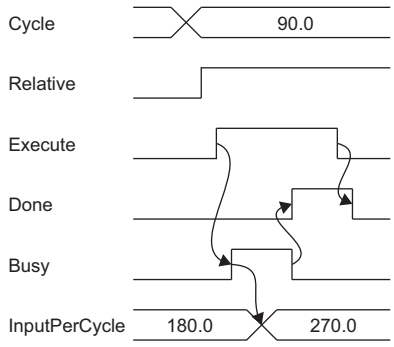
Ex.

When public variable Current value per cycle (InputPerCycle) is "180.0" at stop, and a current value per cycle change of "90.0" is set in Current value per cycle (Cycle)

• When Relative selection (Relative) is FALSE (Absolute position)



• When Relative selection (Relative) is TRUE (Relative distance)



The following table shows whether or not exceeding the current value per cycle is possible when the current value per cycle is changed.

○: Exceeding possible, ×: Exceeding not possible

Setting value of Relative selection (Relative)	Relationship between 1 cycle and amount moved by the current value per cycle	Exceeding Possible
FALSE (Absolute position)	1 cycle > movement amount of the current value per cycle	×
TRUE (Relative distance)	1 cycle > movement amount of the current value per cycle	○
	1 cycle ≤ movement amount of the current value per cycle	○*1

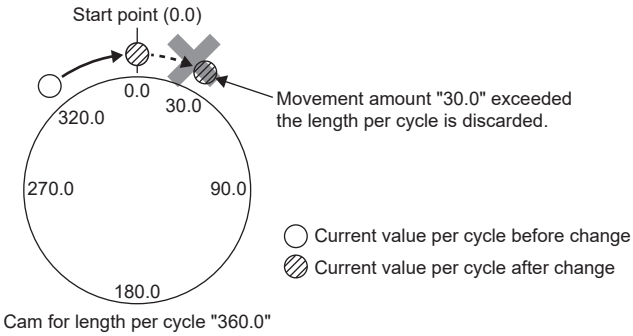
*1 The operation when exceeding occurs will differ based on the set values for Periodic (Periodic) set under MC_CamTableSelect (Cam Table Selection)

Setting value of Periodic (Periodic)	Description
FALSE (Non periodic)	Movement amount exceeding 1 cycle is disposed.
TRUE (Periodic)	The current value per cycle is changed when exceeding 1 cycle.

Ex.

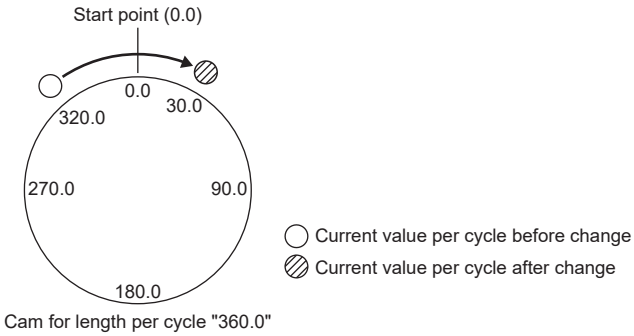
When the Current Value per Cycle (Cycle) changes from "320.0" to "30.0"

• When Periodic (Periodic) is FALSE (Non periodic).



Cam for length per cycle "360.0"

• When Periodic (Periodic) is TRUE (Periodic).

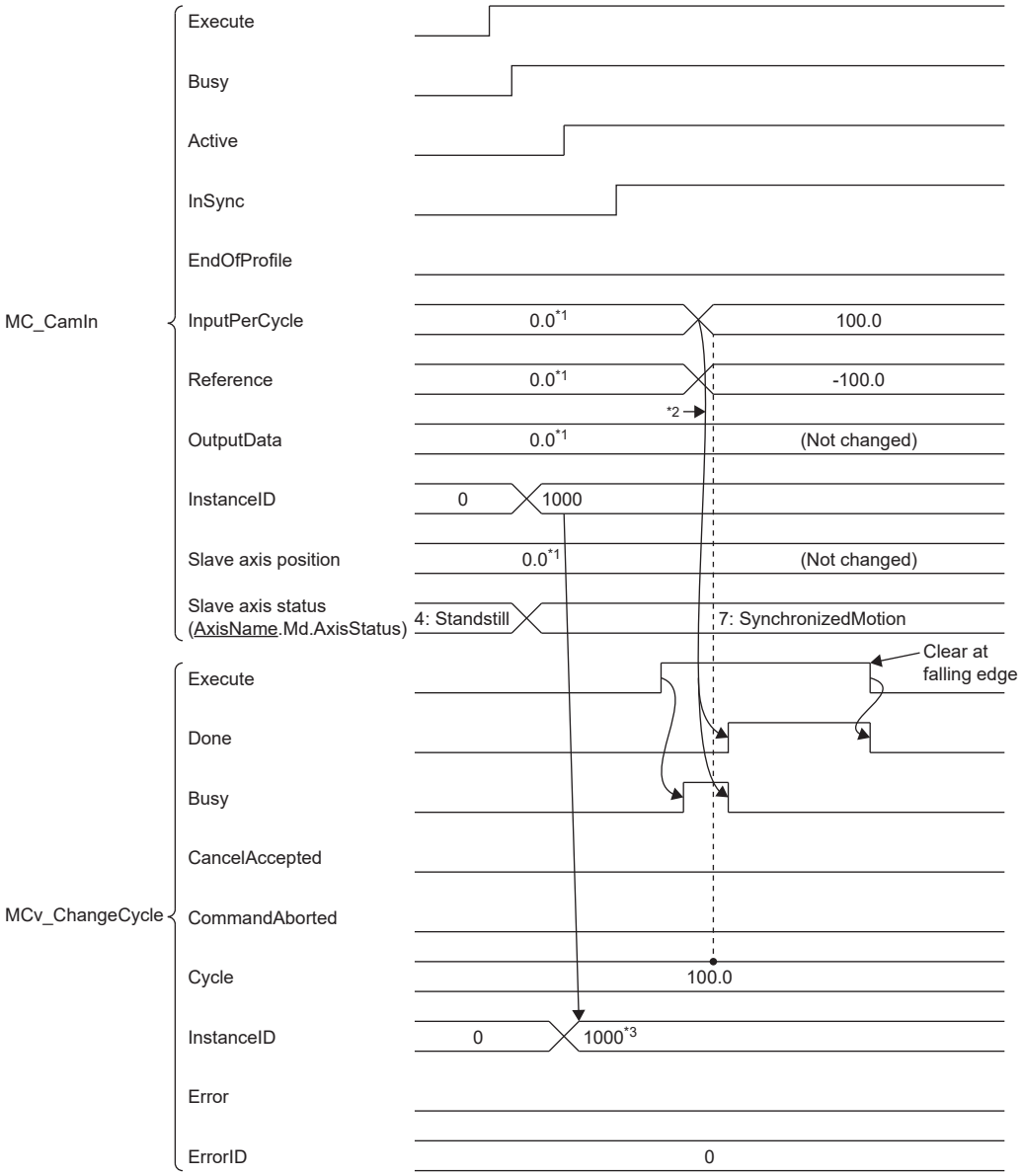


Cam for length per cycle "360.0"

■Execution mode (ExecutionMode)

When MCv_ChangeCycle (Current Value Change per Cycle) is executed, the timing for the current value per cycle change differs depending on the setting of Execution mode (ExecutionMode). The timing for Execution mode (ExecutionMode) is shown below.

Setting value	Description
0: Execute immediately (mcImmediately)	It is immediately executed using Start (Execute) rising edge detection regardless of whether Executing (Busy) of the operation profile data control FB (such as MC_CamIn) is TRUE. Example: When current value per cycle change is executed while Executing (Busy) of MC_CamIn (Cam Operation Start) is TRUE and "100.0" is set in Current value per cycle (Cycle) (The current value for Master axis (Master) of MC_CamIn (Cam Operation Start) is considered as not being changed.)

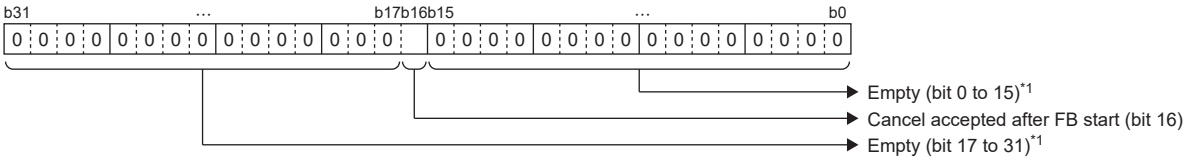


*1: Does not change because the master axis does not move.
*2: At current value change per cycle completion, Completion (Done) becomes TRUE and Executing (Busy) becomes FALSE.
*3: The instance ID of the FB that executes the current value change per cycle is set manually.

■Options (Options)

The function options for MCv_ChangeCycle (Current Value Change per Cycle) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
16	Cancel accepted after FB start	Whether or not to allow a cancel after MCv_ChangeCycle (Current Value Change per Cycle) is executed is set. • 0: Disabled • 1: Enabled

- A current value per cycle change that is in standby status can be canceled after FB execution by setting Cancel accepted after FB start.
  - The cancel is started at falling edge detection of Start (Execute).
  - The cancel is accepted only when Executing (Busy) is TRUE.
  - When the FB accepts the cancel, Cancel acceptance (CancelAccepted) becomes TRUE.
  - When the cancel is completed, Abortion of execution (CommandAborted) becomes TRUE.
  - When the cancel was performed, Current value per cycle (InputPerCycle) is not changed.

Program example

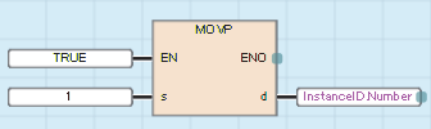
The following is a program example. The current value change per cycle command (bChangeCycle) is set to TRUE, and the current value per cycle of the Instance ID (InstanceID) set to "1" is changed to "90.0".

■Labels used

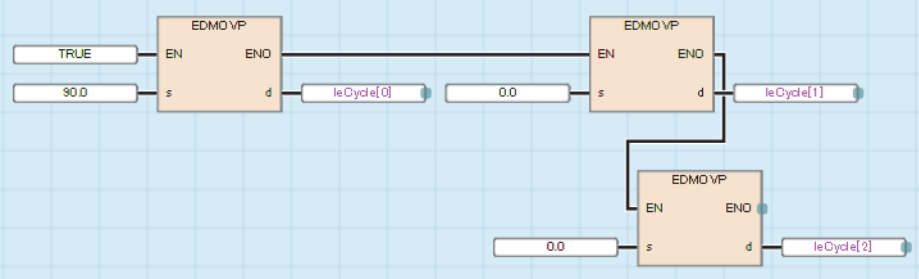
Label name	Data type	Comment
MCv_ChangeCycle_1	MCv_ChangeCycle	Current Value Change per Cycle FB
bChangeCycle	Bit	Current value change per cycle command
InstanceID	INSTANCE_ID	Instance ID
leCycle	Double-precision real number (0..2)	Current value per cycle
bDone	Bit	Execution completion
bBusy	Bit	Executing
bCommandAborted	Bit	Abortion of execution
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code
bCancelAccepted	Bit	Cancel acceptance

■FBD/LD program

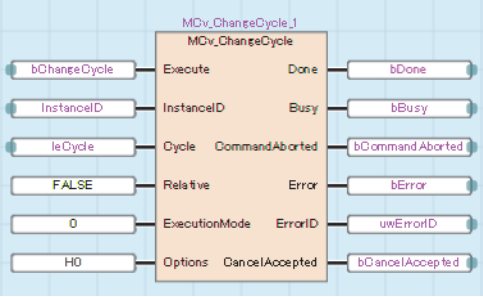
• Operation profile data setting



• Data setting for Current Value Change per Cycle



• Current Value Change per Cycle



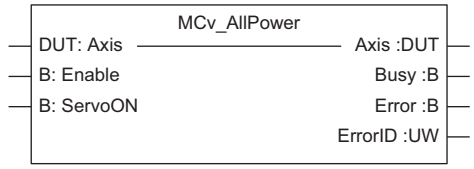
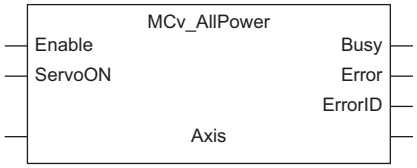
■ST program

```
//-----Operation profile data setting-----
InstanceID.Number:= 1;

//-----Data setting for Current Value Change per Cycle-----
leCycle[0]:= 90.0;
leCycle[1]:= 0.0;
leCycle[2]:= 0.0;

//-----Current Value Change per Cycle-----
MCv_ChangeCycle_1(
   Execute:= bChangeCycle ,
   InstanceID:= InstanceID ,
   Cycle:= leCycle ,
   Relative:= FALSE ,
   ExecutionMode:= MC_EXECUTION_MODE__mcImmediately ,
   Options:= H00000000 ,
   Done=> bDone ,
   Busy=> bBusy ,
   CommandAborted=> bCommandAborted ,
   Error=> bError ,
   ErrorID=> uwErrorID ,
   CancelAccepted=> bCancelAccepted
);
```

41.15 All Axes Operation Available

MCv_AllPower

This FB switches every axis to the operation possible status.

Ladder	FBD/LD	ST
		MCv_AllPower(Axis:= ?AXIS_REF? , Enable:= ?BOOL? , ServoON:= ?BOOL? , Busy=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission possible	The setting value is ignored.

■Input variables

Input variable	Name	Data type	Input	Setting range	Default value	Description
Enable	Enable	BOOL	Always	TRUE, FALSE	FALSE	When this variable is set to TRUE, control of the specified axis is enabled (operation possible status). When this variable is set to FALSE, control of the specified axis is disabled.
ServoON	Servo ON request	BOOL	Always	TRUE, FALSE	FALSE	When this variable is set to TRUE, the servo ON is requested to the specified axis.

■Output variables

Output variable	Name	Data type	Default value	Description
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_AllPower (All Axes Operation Available) is executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

- This FB initializes the information of all axes and switches the axis status to the operation possible status.
- When the Enable (Enable) and Servo ON request (ServoON) inputs are set to TRUE, every axis is switched to the operation possible status.
- Starting the processing causes Executing (Busy) to become TRUE.
- When an error occurs in MCv_AllPower (All Axes Operation Available), Error (Error) becomes TRUE and the error code is stored in Error code (ErrorID). For details on error codes, refer to the following.

User's manual for the controller used

- Inputting Enable (Enable) and Servo ON request (ServoON) switches the servo ON/OFF status and the drive unit status of all real axes as shown below.

Input variables		Servo ON/OFF status	Driver status (AxisName.Md.Driver_State)
Enable (Enable)	Servo ON request (ServoON)		
TRUE	TRUE	Servo ON	6: Operation Enable (OperationEnable)
	FALSE	Servo OFF	5: Switched On (SwitchedOn)
FALSE	TRUE	Servo OFF	3: Switched On Disabled (SwitchOnDisabled)
	FALSE	Servo OFF	3: Switched On Disabled (SwitchOnDisabled)

- If the real axis is rotated by external force during the servo OFF status, follow up processing is performed.
- The servo ON/OFF control can be operated regardless of the control mode. The control mode used during servo OFF depends on the specification of the drive unit.
- MCv_AllPower (All Axes Operation Available) is sent to the drive unit while a drive unit error is occurring, so Enable (Enable) and Servo ON request (ServoON) do not need to be turned from FALSE to TRUE again.

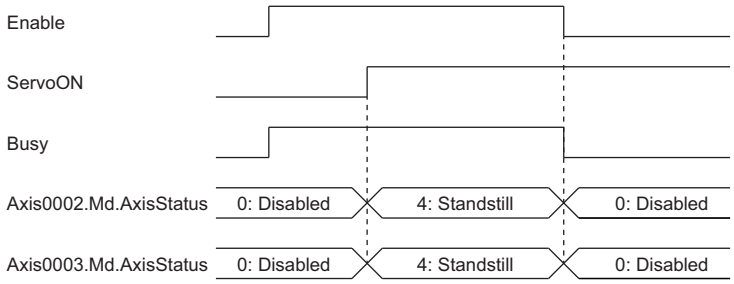
Point

Use MCv_AllPower (All Axes Operation Available) and MC_Power (Operation Available) together if executing servo OFF individually.
When MCv_AllPower (All Axes Operation Available) and MC_Power (Operation Available) are used together, the MC_Power (Operation Available) command is given priority.

■Timing chart

Ex.

When Axis 2 (Axis0002) and Axis 3 (Axis0003) are set as real drive axes



Precautions

- For axes setting MC_Power (Operation Available), the MC_Power (Operation Available) command is given priority for switching to operable status.
- The servo ON/OFF status when stopping the program is maintained while Built-in motion ready (X20) is off.

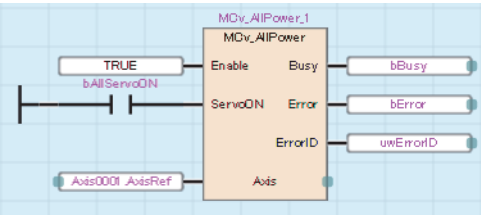
Program example

The following is a program example. All the axes servo ON/OFF (bAllServoON) is set to TRUE, and all axes are switched to operation possible status.

■Labels used

Label name	Data type	Comment
MCv_AllPower_1	MCv_AllPower	All axes servo ON FB
bAllServoON	Bit	All axes servo ON/OFF
bBusy	Bit	Executing
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program



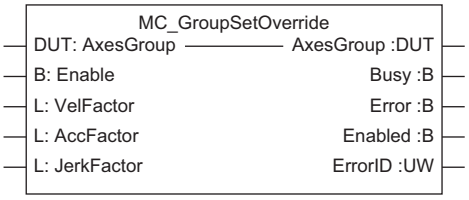
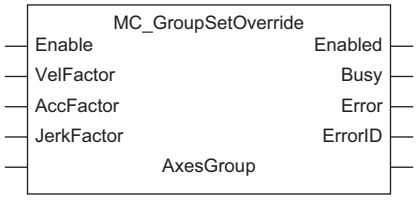
■ST program

```
MCv_AllPower_1(
   Enable:= TRUE ,
   ServoON:= bAllServoON ,
   Busy=> bBusy ,
   Error=> bError ,
   ErrorID=> uwErrorID
);
```

41.16 Axes Group Override Value Setting

MC_GroupSetOverride

This FB changes the target velocity, the target acceleration, and the target deceleration of the specified axes group.

Ladder	FBD/LD	ST
		MC_GroupSetOverride(AxesGroup:= ?AXES_GROUP_REF? , Enable:= ?BOOL? , VelFactor:= ?LREAL? , AccFactor:= ?LREAL? , JerkFactor:= ?LREAL? , Enabled=> ?BOOL? , Busy=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
AxesGroup	Axes group information	AXES_GROUP_REF	At start	—	Omission not possible	An axes group is set. When MC_GroupSetOverride (Axes Group Override Value Setting) is used, I/O No. (StartIO) is set. For the variable (AxesGroupName.AxesGroupRef.) used, refer to the following. ☞ Page 1423 AxesGroupName.AxesGroupRef. (Axes group information)

■Input variable

Input variable	Name	Data type	Input	Setting range	Default value	Description
Enable	Enable	BOOL	Always	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_GroupSetOverride (Axes Group Override Value Setting) is executed.
VelFactor	Velocity override factor	LREAL	Always	0.00 to 10.00	0.00	The velocity override factor is set. When Enable (Enable) is TRUE, the value is always imported.
AccFactor	Acceleration override factor	LREAL	Always	0.00, 0.01 to 10.00	0.00	The acceleration override factor is set. When Enable (Enable) is TRUE, the value is always imported. When "0.00" is set, the acceleration override factor is not changed and the control is performed with the previous value.
JerkFactor	Jerk override factor	LREAL	Always	0.0	0.0	"0.0" must be set. *: If a value other than "0.0" is set, Out of Jerk Override Coefficient (JerkFactor) Range (error code: 1BBEH) occurs.

■Output variables

Output variable	Name	Data type	Default value	Description
Enabled	Enabled	BOOL	FALSE	This variable becomes TRUE when the override values are set correctly.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_GroupSetOverride (Axes Group Override Value Setting) is executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

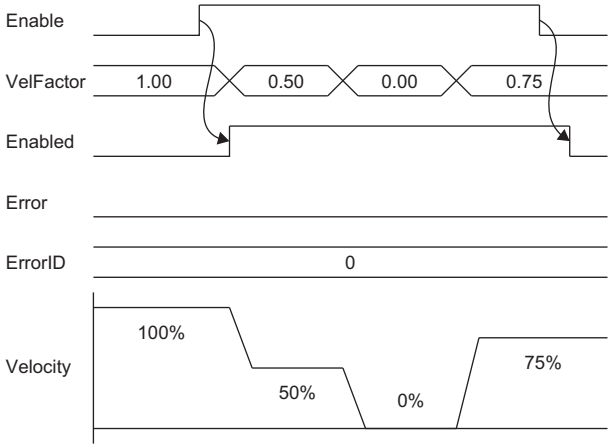
- This FB changes the target velocity, the target acceleration, and the target deceleration of the specified axes group.
- The current operating target velocity, target acceleration, and target deceleration values are changed by multiplying each value by the override factor.
- MC_GroupSetOverride (Axes Group Override Value Setting) is executed when Enable (Enable) becomes TRUE. Enabled (Enabled) becomes TRUE while the override factor is valid.
- If the override factor value is changed while Enable (Enable) is TRUE, the new override factor is reflected.
- When an error occurs in MC_GroupSetOverride (Axes Group Override Value Setting), Error (Error) becomes TRUE and the error code is stored in Error code (ErrorID). For details on error codes, refer to the following.

📖 User's manual for the controller used

- When the value for Velocity override factor (VelFactor) is set to "0.00", the axis stops without Axis status (AxisName.Md.AxisStatus) changing to "4: Standby (Standstill)".
- When the value for Acceleration override factor (AccFactor) is set to "0.00", the acceleration override factor is not changed and the previous acceleration override factor is maintained.

■Timing chart

- When the processing is completed successfully



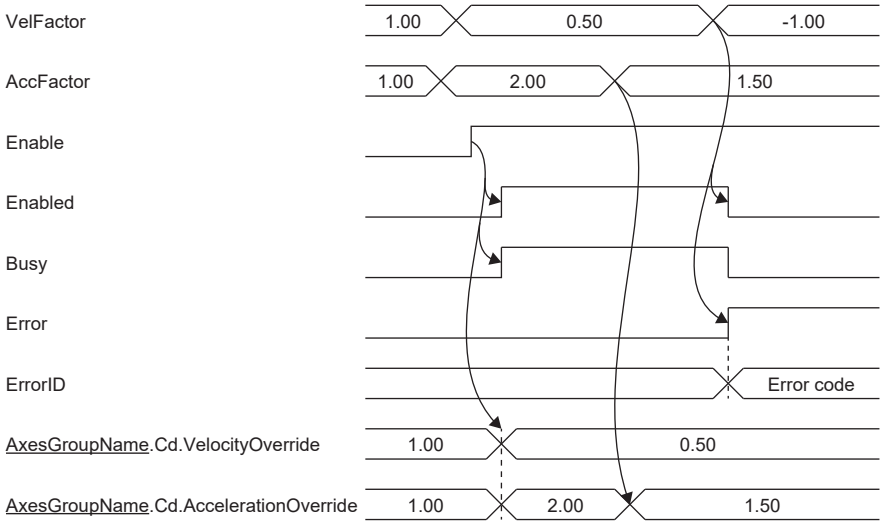
- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1390 Basic operation of Enable (Enable) type motion control FBs

■Operation overview

- In MC_GroupSetOverride (Axes Group Override Value Setting), Velocity override factor (AxesGroupName.Cd.VelocityOverride) and Acceleration override factor (AxesGroupName.Cd.AccelerationOverride) are changed.



- If the override factor is set to a value outside the range, an error will occur in MC_GroupSetOverride (Axes Group Override Value Setting) and subsequent values will not be imported. To import values again, restart Enable (Enable).

Precautions

- Do not assign two or more MC_GroupSetOverride (Axes Group Override Value Setting) FBs in the same axes group. The operation is not guaranteed if two or more of this FB are set.
- For axes group control, only the override factor set with the axes group control data has an effect.
- While executing MC_GroupSetOverride (Axes Group Override Value Setting), do not perform direct operation of Velocity override factor (AxesGroupName.Cd.VelocityOverride) and Acceleration override factor (AxesGroupName.Cd.AccelerationOverride).
- When the speed after Speed override is out of range, refer to "Velocity Range" in the following.

📖 User's manual for the controller used

- When the acceleration, deceleration, acceleration time, and deceleration time after acceleration override is out of range, refer to "Acceleration/deceleration processing function" in the following.

📖 User's manual for the controller used

Program example

The following is a program example. The axes group override value change command (bGroupSetOverride) is set to TRUE, and the target velocity, target acceleration, and target deceleration of Axes group 1 (AxesGroup001) are changed to values multiplied by a velocity override factor of "1.0" and an acceleration override factor of "2.0".

■Axes group

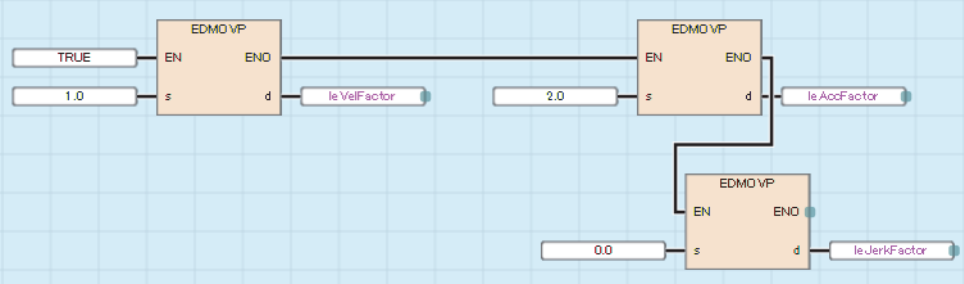
Axes group No.	Label name	Data type	Comment
1	AxesGroup001	AXES_GROUP_REF	Axes group 1

■Labels used

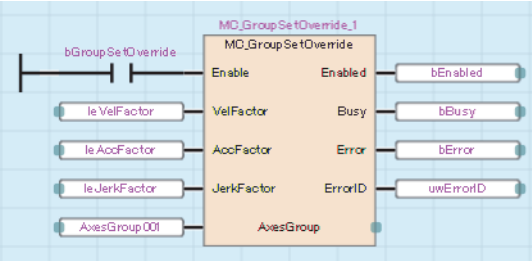
Label name	Data type	Comment
MC_GroupSetOverride_1	MC_GroupSetOverride	Axes Group Override Value Setting FB
bGroupSetOverride	Bit	Axes group override value change command
leVelFactor	Double-precision real number	Velocity override factor
leAccFactor	Double-precision real number	Acceleration/deceleration override factor
leJerkFactor	Double-precision real number	Jerk override factor
bEnabled	Bit	Enabled
bBusy	Bit	Executing
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data setting for Axes Group Override Value Setting



• Axes group override value change



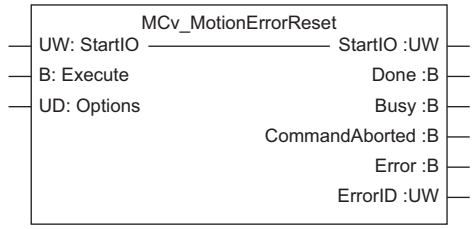
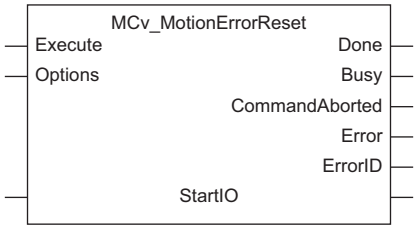
■ST program

```
//-----Data setting for Axes Group Override Value Setting-----
leVelFactor:= 1.00;
leAccFactor:= 2.00;
leJerkFactor:= 0.0;

//-----Axes group override value change-----
MC_GroupSetOverride_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Enable:= bGroupSetOverride ,
   VelFactor:= leVelFactor ,
   AccFactor:= leAccFactor ,
   JerkFactor:= leJerkFactor ,
   Enabled=> bEnabled ,
   Busy=> bBusy ,
   Error=> bError ,
   ErrorID=> uwErrorID
);
```

41.17 Motion Error Reset

MCv_MotionErrorReset

This FB resets all errors and warnings of the motion system.

Ladder	FBD/LD	ST
		MCv_MotionErrorReset(StartIO:= ?WORD? , Execute:= ?BOOL? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
StartIO	I/O No.	WORD(HEX)	At start	000H to 0FEH	Omission possible	The start I/O No. (first 3 digits when displayed as 4 hexadecimal digits) is set.

■Input variables

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_MotionErrorReset (Motion Error Reset) is executed.
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	Set "00000000H". *: If a value other than "00000000H" is set, Out of option range (error code: 1BC3H) occurs.

■Output variables

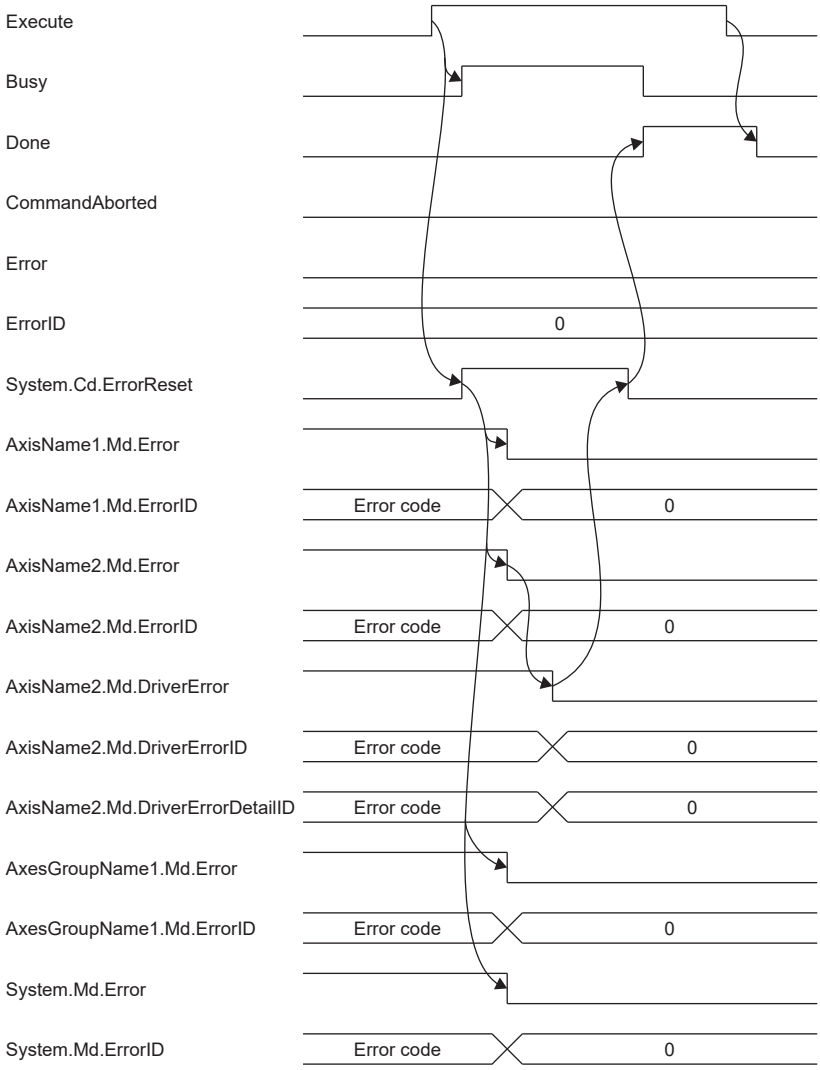
Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the error reset is completed.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_MotionErrorReset (Motion Error Reset) is executed.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when MCv_MotionErrorReset (Motion Error Reset) has been aborted because of timeout. The variable becomes FALSE when Execution command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. User's manual for the controller used

Processing details

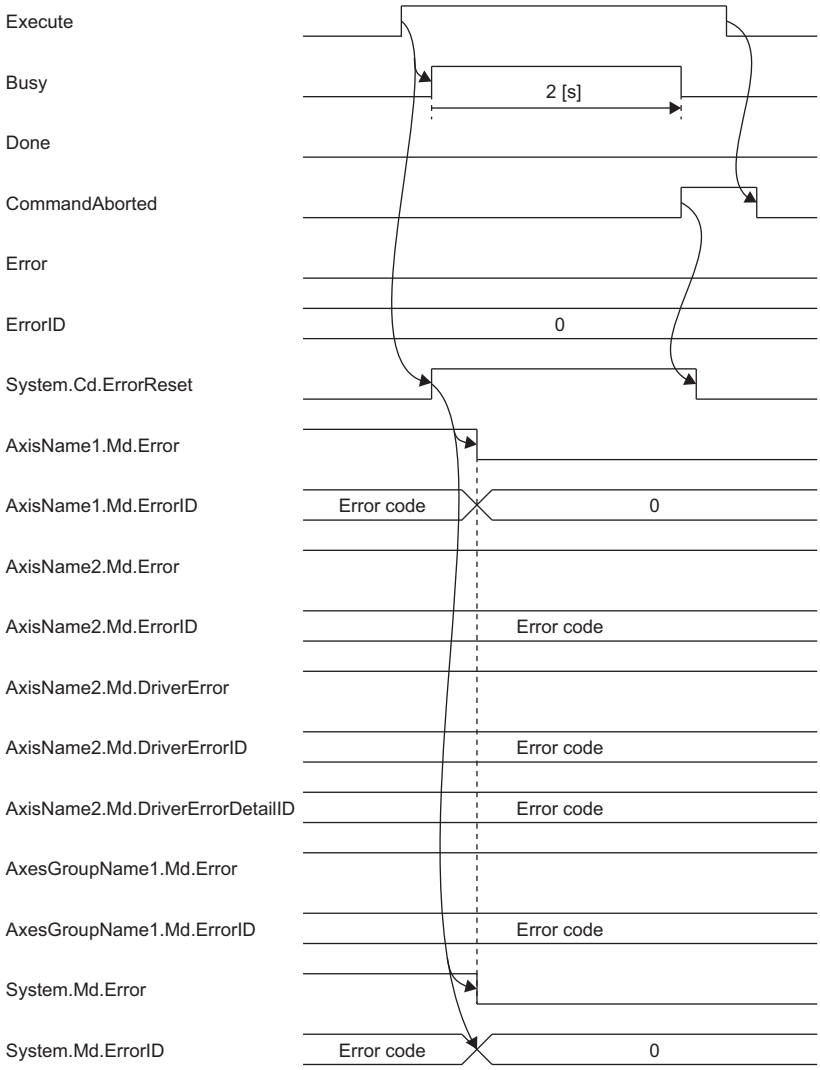
- MCv_MotionErrorReset (Motion Error Reset) is executed when Execution command (Execute) is set to TRUE. When the processing starts, Executing (Busy) becomes TRUE and System error reset (MotionSystem.Cd.ErrorReset) also becomes TRUE.
- Once errors and warnings have been cleared for the motion system, Executing (Busy) becomes FALSE and Execution completion (Done) becomes TRUE.
- Setting Execution command (Execute) to TRUE while the causes of the errors and warnings remain does not clear the errors and warnings. In addition, if the errors are not cleared within two seconds after this FB is executed, Abortion of execution (CommandAborted) becomes TRUE and changes System error reset (MotionSystem.Cd.ErrorReset) to FALSE. Turn Execution command (Execute) to FALSE temporarily and clear the error and warning causes, then set Execution command (Execute) to TRUE again.

■Timing chart

- When the processing is completed successfully



• At timeout



Precautions

- For MCv_MotionErrorReset (Motion Error Reset), the special relay (SM0) and the special register (SD0) are not cleared, and the ERROR LED of the module is not turned off. For how to perform these operations, check the user's manual for the controller used.
- Do not directly operate System error reset (MotionSystem.Cd.ErrorReset) while MCv_MotionErrorReset (Motion Error Reset) is being executed.

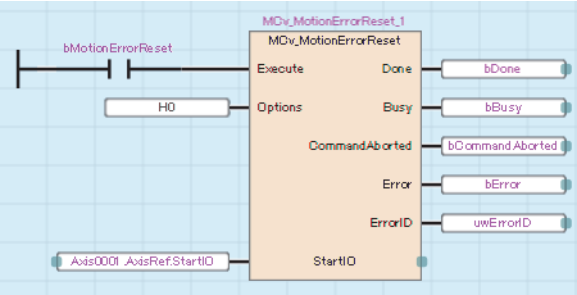
Program example

The following is a program example. The error reset (bMotionErrorReset) is set to TRUE, and all errors and warnings of the motion system are reset.

■Labels used

Label name	Data type	Comment
MCv_MotionErrorReset_1	MCv_MotionErrorReset	Motion Error Reset FB
bMotionErrorReset	Bit	Error reset
bDone	Bit	Execution completion
bBusy	Bit	Executing
bCommandAborted	Bit	Abortion of execution
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program



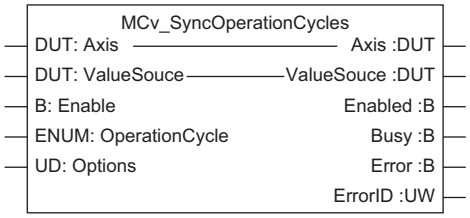
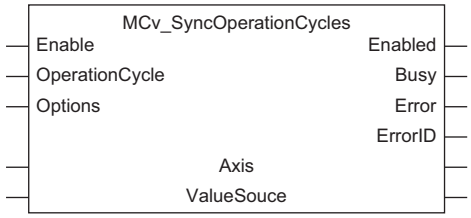
■ST program

```
MCv_MotionErrorReset_1(
   Execute:= bMotionErrorReset ,
   Options:= H00000000 ,
   Done=> bDone ,
   Busy=> bBusy ,
   CommandAborted=> bCommandAborted ,
   Error=> bError ,
   ErrorID=> uwErrorID
);
```

41.18 Control Operation Cycle Synchronization

MCv_SyncOperationCycles

The control operation cycle of the axis is synchronized with the specified cycle.

Ladder	FBD/LD	ST
		MCv_SyncOperationCycles(Axis:= ?AXIS_REF? , ValueSource:= ?SIGNAL_SELECT? , Enable:= ?BOOL? , OperationCycle:= ?INT? , Options:= ?DWORD? , Enabled=> ?BOOL? , Busy=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)
ValueSource	Data source selection	SIGNAL_SELECT	At start	—	Omission possible	The input is ignored.

■Input variable

Input variable	Name	Data type	Input	Setting range	Default value	Description
Enable	Enable	BOOL	Always	TRUE, FALSE	FALSE	While this variable remains TRUE, the control operation cycle synchronization is enabled.
OperationCycle	Control operation cycle	INT	At start	1 to 3	1	The control operation cycle of the synchronization target is set. • 1: First operation cycle • 2: Second operation cycle • 3: Third operation cycle
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	Set "00000000H". An error occurs if a value other than "00000000H" is set.

■Output variable

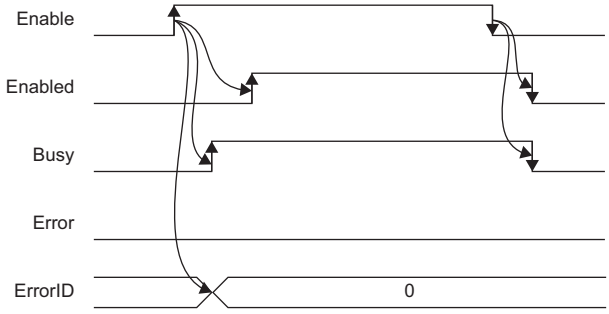
Output variable	Name	Data type	Default value	Description
Enabled	Enabled	BOOL	FALSE	This variable becomes TRUE when the control operation cycle synchronization is operating correctly.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_SyncOperationCycles (Control Operation Cycle Synchronization) is being executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

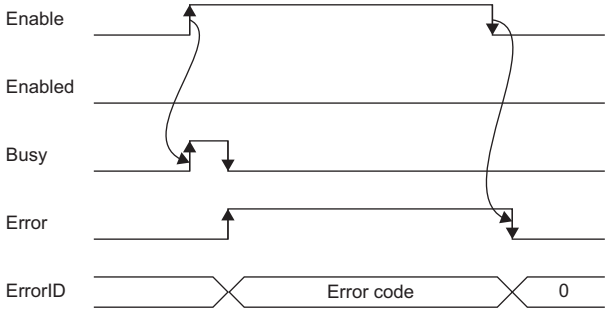
- MCv_SyncOperationCycles (Control Operation Cycle Synchronization) synchronizes with the cycle of the value to which the control operation cycle of the selected data (axis or data source) is set.
- This function block is used if the control operation cycles of the master axis and slave axis are not the same in the single axis synchronous control FB.
- When Enable (Enable) becomes TRUE, control operation cycle synchronization is enabled. While the control operation cycle synchronization is enabled, Enabled (Enabled) is TRUE.

■Timing chart

- When the processing is completed successfully



- When the processing is completed with an error



Precautions

- This FB synchronizes the selected data with the operation cycle during the specified control time but does not change the control operation cycle setting.
- When the control operation cycles are not the same between the input and the output (input: master axis, output: slave axis in synchronous control), start this FB before starting synchronous control. If synchronous control is started without synchronization, the operation cycle mismatch (error code: 1B29H) occurs.
- When this FB is executed, the operation time increases.
- The input argument refresh for a motion control FB is performed when the FB is read, but this FB is refreshed by the operation cycle, regardless of the FB call task (scan/fixed cycle/event).

Program example

The following is a program example. Axes in different control cycles are synchronously controlled by using Control Operation Cycle Synchronization (MCv_SyncOperationCycles).
This program example starts MC_GearIn while handling Axis0001 to be controlled in the first operation cycle as the master axis and Axis0002 to be controlled in the second operation cycle as the slave axis.
The program that executes servo ON of the MR-J5-G is omitted.

■CC-Link IE TSN configuration

Station No.	Type name	IP address	Network synchronous communication setting	Communication period setting
1	MR-J5-G	192.168.4.1	Synchronize	Basic cycle
2	MR-J5-G	192.168.4.2	Synchronize	Medium speed

■Axes

Axis No.	Label name	Station address setting	Axis type	Control cycle setting
1	Axis0001	192.168.4.1	Real drive axis	Operate in the first operation cycle
2	Axis0002	192.168.4.2	Real drive axis	Operate in the second operation cycle

■Labels used

Label name	Data type	Comment
bSyncStart	Bit	Synchronous control start
bMCGearInExecute	Bit	Gear operation start
MCv_SyncOperationCycles_1	MCv_SyncOperationCycles	Control operation cycle synchronization FB
MC_GearIn_1	MC_GearIn	Gear Operation Start FB

■ST program

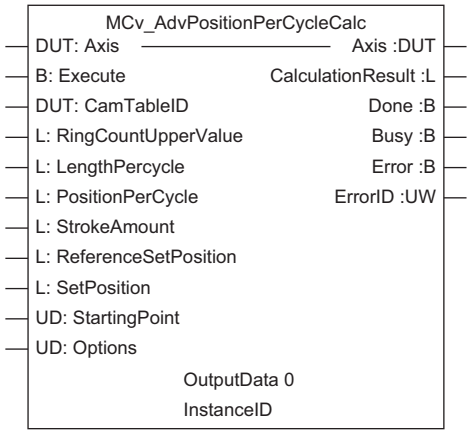
```
SET(MCv_SyncOperationCycles_1.Enabled, bMCGearInExecute);
//-----Control operation cycle synchronization-----
MCv_SyncOperationCycles_1(
   Axis:= Axis0001.AxisRef ,
//ValueSource:= ?SIGNAL_SELECT? ,//Omitted
   Enable:= bSyncStart ,
   OperationCycle:= 2 ,//2: Second operation cycle
   Options:= 0
);

//-----Gear Operation Start-----
MC_GearIn_1(
   Master:= Axis0001.AxisRef ,
   Slave:= Axis0002.AxisRef ,
   Execute:= bMCGearInExecute ,
   ContinuousUpdate:= TRUE ,
   RatioNumerator:= 1 ,
   RatioDenominator:= 1 ,
   MasterValueSource:= MC_SOURCE__mcSetValue ,
   Acceleration:= 1000.0 ,
   Deceleration:= 1000.0 ,
   Jerk:= 0.0 ,
   BufferMode:= MC_BUFFER_MODE__mcAborting ,
   Options:= 0
);
```

41.19 Advanced Synchronous Control One Cycle Current Position Calculation

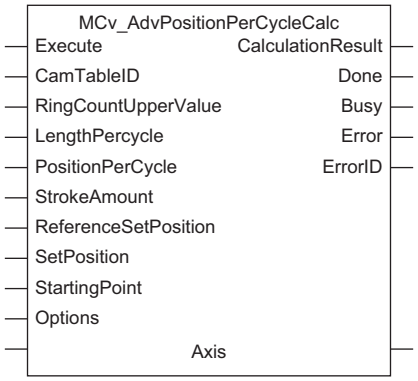
MCv_AdvPositionPerCycleCalc

This FB calculates the current position per cycle based on the specified cam data.

Ladder



FBD/LD



ST

```
MCv_AdvPositionPerCycleCalc(
  Axis:= ?AXIS_REF? ,
  Execute:= ?BOOL? ,
  CamTableID:= ?MC_CAM_ID? ,
  RingCountUpperValue:= ?LREAL? ,
  LengthPercycle:= ?LREAL? ,
  PositionPerCycle:= ?LREAL? ,
  StrokeAmount:= ?LREAL? ,
  ReferenceSetPosition:= ?LREAL? ,
  SetPosition:= ?LREAL? ,
  StartingPoint:= ?DWORD? ,
  Options:= ?DWORD? ,
  CalculationResult=> ?LREAL? ,
  Done=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission possible	The setting is ignored.

■Input variables

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	This FB is executed when the value is TRUE.
CamTableID	Cam table ID	MC_CAM_ID	At start	0 to 6000	0	The operation cam table ID of the calculation target is set. When "0" is set in Profile ID No. (Number) of the PROFILE_ID structure, the target is calculated as a linear cam.
RingCountUpperValue	Ring counter upper limit	LREAL	At start	0.0, 2.0 to 2147483647.0	0.0	The ring counter upper limit value for the calculation target output axis is set when the ring counter upper limit is to be considered in cam position calculations. When "0.0" is set, the calculation is performed with the ring counter set as "-10000000000.0 to 10000000000.0".
LengthPerCycle	Cam position one cycle length	LREAL	At start	1.0 to 2147483647.0	1.0	The length per cycle required for the cam calculation is set.
PositionPerCycle	Cam position calculation current position per cycle	LREAL	At start	0.0 to 2147483647.0	0.0	The current position per cycle required for the cam calculation is set. When a value greater than or equal to Cam position one cycle length (LengthPerCycle) is set, the value is converted within the range of "0.0 to (Cam position one cycle length (LengthPerCycle) - 0.00001)" for calculation.
StrokeAmount	Cam position calculation stroke amount	LREAL	At start	-2147483648.0 to 2147483647.0	0.0	The cam stroke amount required for the cam calculation is set.

Input variable	Name	Data type	Input	Setting range	Default value	Description
ReferenceSetPosition	Cam position calculation cam reference position	LREAL	At start	-10000000000.0 to 10000000000.0	0.0	The cam reference position required for the cam calculation is set.
SetPosition	Cam position calculation cam set position	LREAL	At start	-10000000000.0 to 10000000000.0	0.0	The output axis current position required for the cam calculation is set.
StartingPoint	Cam position calculation cam start point	DWORD	At start	0 to 65535	0	This variable is set when the cam start point is to be considered in the cam calculation.
Options	Options	DWORD(HEX)	At start	Fixed to 0	0	This variable cannot be set for this FB. Set 0. If a value other than 0 is set, Out of options range (error code: 1ABBH) occurs.

■Output variables

Output variable	Name	Data type	Default value	Description
CalculationResult	Operation result	LREAL	0.0	The calculation result of the current position per cycle is stored.
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the calculation is completed.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when this FB is executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. User's manual for the controller used

■Public variables

Public variable	Name	Data type	Default value	Description
OutputData	Output data	LREAL	0.0	• The stroke amount corresponding to the calculation result is stored when the target is a linear cam, or when Specifying interpolation method (Interpolate) of the operation profile data is set to "0: Linear interpolation". • The stroke ratio of the cam data corresponding to the calculation result (current position per cycle) is stored in units of percentage when Interpolation method specification of the operation profile data is set to "1: Specifying for each section" or "2: Spline interpolation", or when the data format type is set to "rotary cutter". • The value of the output data (OutputData) is stored when Execution completion (Done) becomes TRUE.
InstanceID	Instance ID	INSTANCE_ID	0	The instance ID is stored. This variable is automatically set by the system when an instance is created.

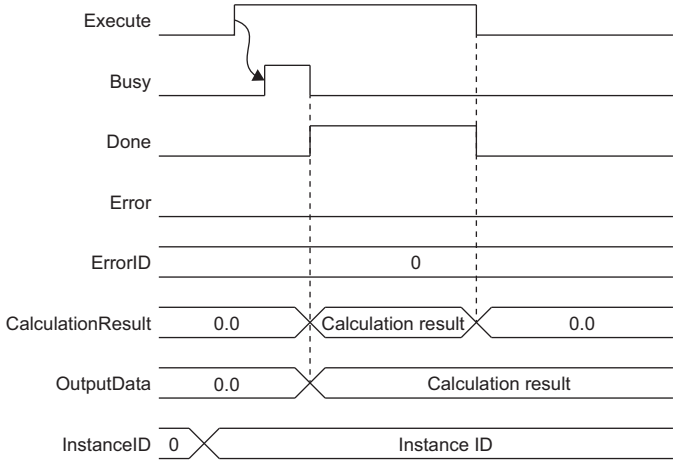
Processing details

- The current position per cycle is calculated from cam data based on Cam position calculation cam reference position (ReferenceSetPosition) and Cam position calculation cam set position (SetPosition). The calculation result is output to Calculation result (CalculationResult).
- For details on the current position per cycle calculation, refer to the following.

📖 User's manual for the controller used

■Timing chart

- When the processing is completed successfully



- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

Precautions

- Before using the cam data, configure the open setting to automatically open the cam data to the open area, or open the data to the area using MC_CamTableSelect (Cam Table Selection).
- Apply the calculation result of the current position per cycle calculation (MCv_AdvPositionPerCycleCalc) only to MCv_AdvancedSync (Advanced synchronous control). If the result is applied to a synchronous control FB (such as MC_CamIn) other than MCv_AdvancedSync (Advanced synchronous control), a mismatch occurs between the calculation result and the actual operation.
- For the current position per cycle calculation of coordinate data, the calculation result of the cam data last point may be as follows depending on the length per cycle. When coordinate data is used, matching the cam data last point with the length per cycle is recommended.

Cam data last point	Description
Greater than the length per cycle	The calculation result may differ from the synchronous control path because the entire range of the cam data is searched regardless of the length per cycle setting value.
Less than the length per cycle	The current value per cycle may be greater than the length per cycle due to the relationship between the cam set position and the cam reference position.

Program example

For a program example of this FB, refer to the following.

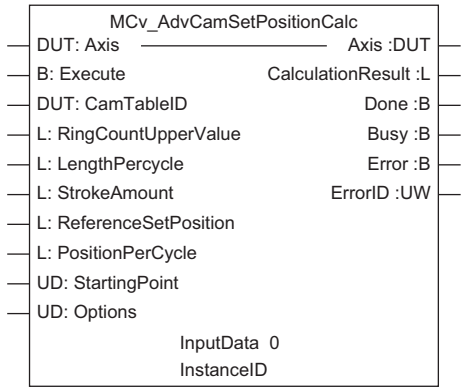
📖 User's manual for the controller used

41.20 Cam Set Position Calculation in Advanced Synchronous Control

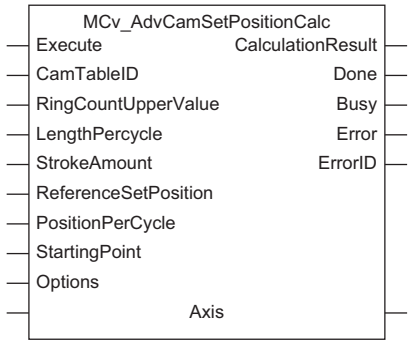
MCv_AdvCamSetPositionCalc

This FB calculates the cam set position based on the specified cam data.

Ladder



FBD/LD



ST

```
MCv_AdvCamSetPositionCalc(
  Axis:= ?AXIS_REF? ,
  Execute:= ?BOOL? ,
  CamTableID:= ?MC_CAM_ID? ,
  RingCountUpperValue:= ?LREAL? ,
  LengthPercycle:= ?LREAL? ,
  StrokeAmount:= ?LREAL? ,
  ReferenceSetPosition:= ?LREAL? ,
  PositionPerCycle:= ?LREAL? ,
  StartingPoint:= ?DWORD? ,
  Options:= ?DWORD? ,
  CalculationResult=> ?LREAL? ,
  Done=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission possible	The setting is ignored.

■Input variables

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	This FB is executed when the value is TRUE.
CamTableID	Cam table ID	MC_CAM_ID	At start	0 to 6000	0	The operation cam table ID of the calculation target is set. When "0" is set in Profile ID No. (Number) of the PROFILE_ID structure, the target is calculated as a linear cam.
RingCountUpperValue	Ring counter upper limit	LREAL	At start	0.0, 2.0 to 2147483647.0	0.0	The ring counter upper limit value for the calculation target output axis is set when the ring counter upper limit is to be considered in cam position calculations. When "0.0" is set, the calculation is performed with the ring counter set as "-10000000000.0 to 10000000000.0".
LengthPerCycle	Cam position one cycle length	LREAL	At start	1.0 to 2147483647.0	1.0	The length per cycle required for the cam calculation is set.
StrokeAmount	Cam position calculation stroke amount	LREAL	At start	-2147483648.0 to 2147483647.0	0.0	The cam stroke amount required for the cam calculation is set.
ReferenceSetPosition	Cam position calculation cam reference position	LREAL	At start	-10000000000.0 to 10000000000.0	0.0	The cam reference position required for the cam calculation is set.
PositionPerCycle	Cam position calculation current position per cycle	LREAL	At start	0.0 to LengthPerCycle	0.0	The current position per cycle required for the cam calculation is set.

Input variable	Name	Data type	Input	Setting range	Default value	Description
StartingPoint	Cam position calculation cam start point	DWORD	At start	0 to 65535	0	This variable is set when the cam start point is to be considered in the cam calculation.
Options	Options	DWORD(HEX)	At start	Fixed to 0	0	This variable cannot be set for this FB. Set 0. If a value other than 0 is set, Out of options range (error code: 1ABBH) occurs.

■Output variables

Output variable	Name	Data type	Default value	Description
CalculationResult	Operation result	LREAL	0.0	The calculation result of the cam set position is stored.
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the calculation is completed.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when this FB is executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. User's manual for the controller used

■Public variables

Public variable	Name	Data type	Default value	Description
InputData	Input data	LREAL	0.0	• The current position per cycle corresponding to the calculation result is stored when the target is a linear cam. • The current position per cycle corresponding to the calculation result (cam set position) is set when Specifying interpolation method (Interpolate) of the operation profile data is set to "0: Linear interpolation". • The resolution of the cam data corresponding to the calculation result (cam set position) is set when Interpolation method specification of the operation profile data is set to "1: Specifying for each section" or "2: Spline interpolation", or when the data format type is set to "rotary cutter". • The value of the input data (InputData) is stored when Execution completion (Done) becomes TRUE.
InstanceID	Instance ID	INSTANCE_ID	0	The instance ID is stored. This variable is automatically set by the system when an instance is created.

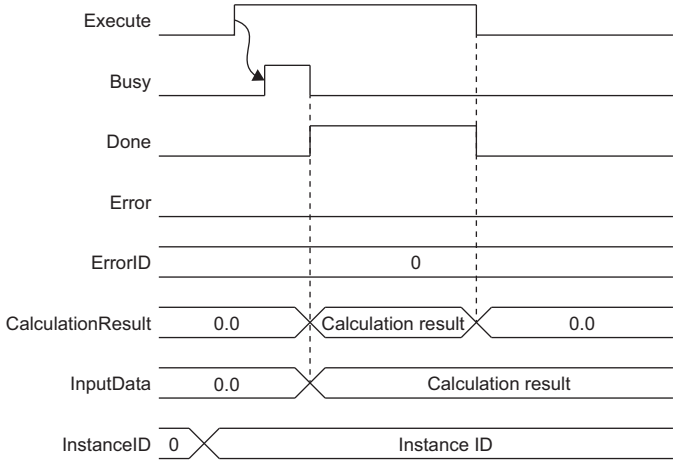
Processing details

- The cam set position is calculated based on Cam position calculation current position per cycle (PositionPerCycle) and Cam position calculation cam reference position (ReferenceSetPosition). The calculation result is output to Calculation result (CalculationResult).
- For details on the cam set position calculation, refer to the following.

📖 User's manual for the controller used

■Timing chart

- When the processing is completed successfully



- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

Precautions

- Before using the cam data, configure the open setting to automatically open the cam data to the open area, or open the data to the area using MC_CamTableSelect (Cam Table Selection).
- Apply the calculation result of the cam set position calculation (MCv_AdvCamSetPositionCalc) only to MCv_AdvancedSync (Advanced synchronous control). If the result is applied to a synchronous control FB (such as MC_CamIn) other than MCv_AdvancedSync (Advanced synchronous control), a mismatch occurs between the calculation result and the actual operation.

Program example

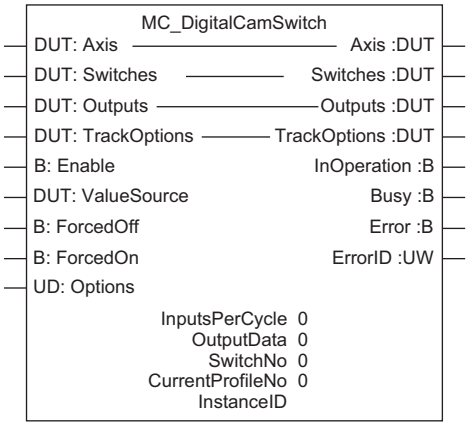
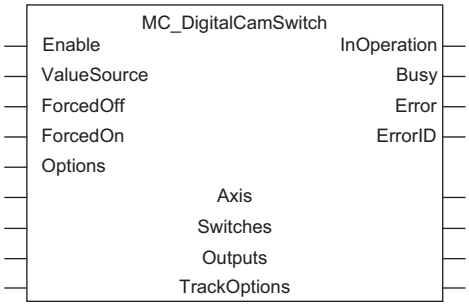
For a program example of this FB, refer to the following.

📖 User's manual for the controller used

41.21 Digital Cam Switch Output

MC_DigitalCamSwitch

This FB outputs the on/off signals according to a given data value.

Ladder	FBD/LD	ST
		MC_DigitalCamSwitch(Axis:= ?AXIS_REF? , Switches:= ?MC_CAMSWITCH_DATA_REF? , Outputs:= ?MC_OUTPUT_REF? , TrackOptions:= ?MC_TRACK_REF? ,Enable:= ?BOOL? , ValueSource:= ?MC_CAMSWITCH_SOURCE_REF? , ForcedOff:= ?BOOL? ,ForcedOn:= ?BOOL? , Options:= ?DWORD? ,InOperation=> ?BOOL? , Busy=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
Axis	Axis	AXIS_REF	At start	—	Omission possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)
Switches	Switch	MC_CAMSWITCH_DATA_REF ☞ Page 1443 MC_CAMSWITCH_REF	Always	1 to 60000	Omission not possible	The switch is set using the operation profile data. ☞ Page 1591 Switches (Switches)
Outputs	Output signal	MC_OUTPUT_REF	At start	—	Omission not possible	The output signal is set. For details, refer to the following. ☞ Page 1592 Output signal (Outputs)
TrackOptions	Track options	MC_TRACK_REF	Always	—	—	The track options are set. For details, refer to the following. ☞ Page 1593 Track options (TrackOptions)

■Input variables

Input variable	Name	Data type	Input	Setting range	Default value	Description
Enable	Enable	BOOL	Always	TRUE, FALSE	FALSE	This FB is executed while the value is TRUE.
ValueSource	Data source selection	MC_CAMSWITCH_SOURCE_REF	At start	—	—	A data source is specified. For details, refer to the following. ☞ Page 1594 Data source selection (ValueSource)
ForcedOff	Forced off	BOOL	Always	TRUE, FALSE	FALSE	The output signals are forcibly turned off while the value is TRUE. For details, refer to the following. ☞ Page 1595 Forced off (ForcedOff)
ForcedOn	Forced on	BOOL	Always	TRUE, FALSE	FALSE	The output signals are forcibly turned on while the value is TRUE. For details, refer to the following. ☞ Page 1595 Forced on (ForcedOn)
Options	Options	DWORD(HEX)	At start	—	00000000H	The function options are set. For details, refer to the following. ☞ Page 1595 Options (Options)

■Output variables

Output variable	Name	Data type	Default value	Description
InOperation	In operation	BOOL	FALSE	This variable becomes TRUE when the output signal is valid.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when this FB is being executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

■Public variables

Public variable	Name	Data type	Default value	Description
InputsPerCycle	Current position per cycle	LREAL	0	The data source value is stored.
OutputData	Output value	BOOL	0	The output value to the output signal is stored.
SwitchNo	Switch number	WORD (UINT)	0	The switch number of the switch that is on is stored.
CurrentProfileNo	Executing profile number	WORD (UINT)	0	The profile ID of the profile currently being executed is stored.
InstanceID	Instance ID	INSTANCE_ID	0	The instance ID is stored. This variable is automatically set by the system when an instance is created.

Processing details

This FB judges whether a data source set for Data source selection (ValueSource) satisfies the switch setting condition and outputs the output signal. For details on the digital cam switch function, refer to the following.

📖 User's manual for the controller used

■Timing chart

For the timing chart of this FB, refer to the following.

☞ Page 1390 Basic operation of Enable (Enable) type motion control FBs

■Switches (Switches)

The operation profile ID is specified with an MC_CAMSWITCH_DATA_REF structure.

For the setting method, refer to the following.

📖 User's manual for the controller used

If Switches (Switches) is changed to outside the range during execution or an open area does not exist, a program warning (event code: 00F19H) occurs, and the operation continues with the value before the change.

■Output signal (Outputs)

Output signals are set with an MC_OUTPUT_REF structure. For details on the MC_OUTPUT_REF structure, refer to the following.

☞ Page 1442 MC_OUTPUT_REF

The setting range for Output signal (Outputs) of the MC_OUTPUT_REF structure set in Output signal (Outputs) is shown below.

The type for Output signal (Outputs) is the SIGNAL_SELECT structure. For the SIGNAL_SELECT structure, refer to the following.

☞ Page 1447 SIGNAL_SELECT (Signal select)

Structure	Variable name	Type	Setting range
SIGNAL_SELECT (Signal select)	Source (Signal)	TARGET_REF ☞ Page 1444 TARGET_REF (Input signal)	■Type • BOOL ■Data category • [LINK]*1
	Detection (Signal detection method)	INT (MC_SIGNAL_LOGIC) ☞ Page 1478 MC_SIGNAL_LOGIC	The signal logic of the bit output signal cannot be selected.
	CompensationTime (Compensation time)*2	LREAL	-5.0 to 5.0 [s]
	FilterTime (Filter time)	LREAL	0.0 [s]*3

*1 When [LINK] is specified, writing to the corresponding network labels is ignored. The setting is only possible when the external signal high-accuracy output is used. For more details, check the user's manual for the controller used.

*2 Set this variable when compensating for delays, such as those caused by the device response time. For details on time compensation, check the user's manual for the controller used.

*3 If a value other than 0.0 is set, Out of filter time setting range of signal (error code: 1BC8H) occurs. To eliminate chattering, set Hysteresis (Switching disabled distance) of Track options (TrackOptions).

When an identical output signal is set for multiple FB instances, the logical sum of the output results is output.

The output signal is refreshed in the following timing, regardless of the program execution cycle used.

Data category	Refresh timing
[LINK]	Communication cycle of the target station

■Track options (TrackOptions)

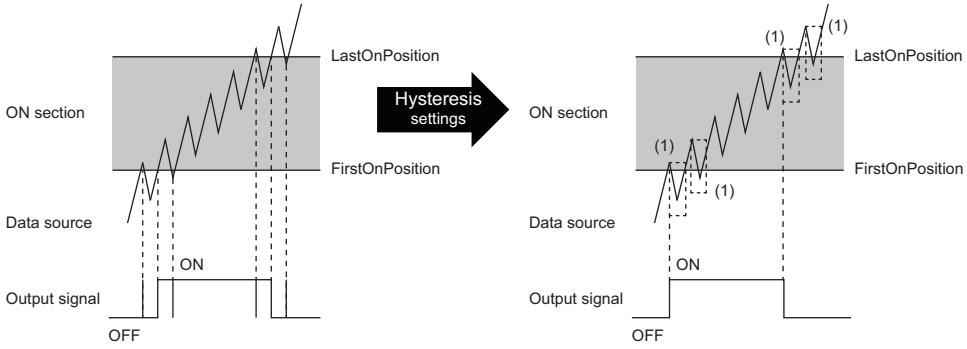
Track options are set with an MC_TRACK_REF structure. For the MC_TRACK_REF structure, refer to the following.

☞ Page 1443 MC_TRACK_REF

The setting range for Track options (TrackOptions) of the MC_TRACK_REF structure to be set for Track options (TrackOptions) is shown below.

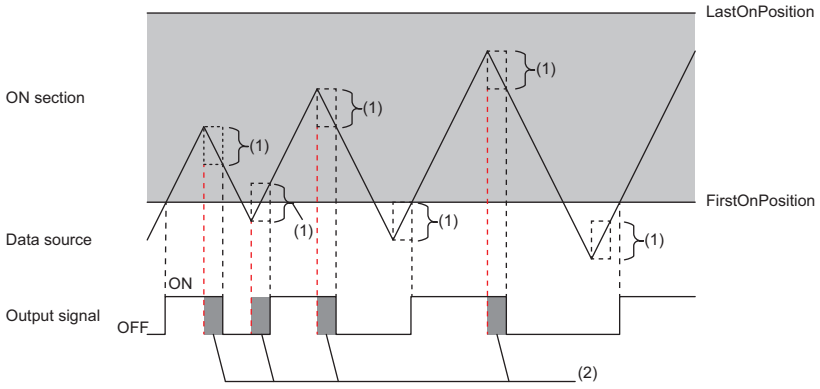
Item	Setting range	Description
Hysteresis (Switching disabled distance)	0.0 to 10000000000.0	Set the dead band to prevent chattering of the output signal when the speed of the data source is unstable.*1 Use less than half of the data source ring counter range for the setting. If half or more of the data source ring counter range is used for the setting, an MC_DigitalCamSwitch instruction error (error code: 1BCAH) occurs. If the setting value is changed to a value outside the range during execution, a program warning (event code: 00F19H) occurs, and the operation continues with the value before the change. While the movement direction of the data source is within Hysteresis from the inverted position, changes in Hysteresis are not reflected to the control. The changes will be reflected to the control when out of Hysteresis.

*1 For details, refer to the following.



(1) Hysteresis (Switching disabled distance)

- When Switch mode selection (CamSwitchMode) in the switch settings is set to "1: Position reference", the output signal is not switched between on and off within the range from the position where the data source change direction is reversed to Hysteresis (Switching disabled distance). When Direction selection (Direction) in the switch settings is set to "1: Positive Direction", the operation is as follows.



(1) Hysteresis (Switching disabled distance)
(2) The on/off switching of the output signal is not performed.

■Data source selection (ValueSource)

The following items are specified in the MC_CAMSWITCH_SOURCE_REF structure.

- Data source
- Upper limit value and lower limit value of the data source

For details on the MC_CAMSWITCH_SOURCE_REF structure, refer to the following.

☞ Page 1443 MC_CAMSWITCH_SOURCE_REF

Variable name	Type	Setting range	Description
Source (Data source)	TARGET_REF ☞ Page 1444 TARGET_REF (Input signal)	■Type • LREAL*1 ■Data category • [AXIS] • [VAR] • [DEV] • [CONST]	The axis data specified by the operand axis (Axis) is referred when [AXIS] is specified for the data classification. This FB continues the control even if the target axis causes an error or is disconnected. When [AXIS] is specified and the current position change or home position return is executed for the target axis while this FB is being executed, an unintentional result may occur in the on/off operation of the output signal. Perform the operations mentioned above in the forced-off status, forced-on status, or while this FB is stopped. The axis data specified by the operand axis (Axis) is ignored when [VAR], [DEV], or [CONST] is specified for the data classification.
RingCount_LowerValue (Data source ring counter lower limit value)*2	LREAL	-10000000000.0 to 10000000000.0	The data source ring counter range is set to fit within the range specified by the following formula. RingCount_LowerValue ≤ Data source < RingCount_UpperValue Set the range by referring to the data source range before (LREAL) is specified for the data type. ■Setting example where INT type data is used as the data source RingCount_LowerValue: -32768.0 RingCount_UpperValue: 32768.0 ■Setting example where user-defined data that can be an integer value from 0 to 99 is used RingCount_LowerValue: 0.0 RingCount_UpperValue: 100.0 If RingCount_LowerValue (Data source ring counter lower limit value) is equal to or greater than RingCount_UpperValue (Data source ring counter upper limit value), an MC_DigitalCamSwitch instruction error (error code: 1BCAH) occurs. If the absolute value obtained by subtracting RingCount_LowerValue (Data source ring counter lower limit value) from RingCount_UpperValue (Data source ring counter upper limit value) is less than 2, an MC_DigitalCamSwitch instruction error (error code: 1BCAH) occurs. Ensure that the data source movement amount per refresh cycle is less than half of the data source ring counter range. If the movement amount of the data source per refresh cycle is equal to or more than half of the data source ring counter range, the direction cannot be determined properly and output signals cannot be output at the correct timing.
RingCount_UpperValue (Data source ring counter upper limit value)*2			

*1 ON section determination is performed as the LREAL type. When specifying data that is not the LREAL type, explicitly specify (LREAL) for the data type. When (LREAL) is not specified for the data type, External signal selection string incorrect (error code: 1BC5H) occurs.

*2 When [Axis] is specified for Source (Data source), the setting value is ignored because the axis data specified by the operand axis (Axis) is referred.

Multiple FB instances can be executed simultaneously on a single data source.

The output signal is refreshed in the following timing, regardless of the program execution cycle used.

Data classification	Refresh timing
[VAR]	Network label: Communication cycle of the target station Other than the network label: First operation cycle
[DEV]	First operation cycle
[CONST]	
[AXIS]	Communication cycle of the target station

■Forced off (ForcedOff)

The output signal is forcibly turned off during execution.

This control has priority over Hysteresis (Switching disabled distance) of Track options (TrackOptions).

Forced off (ForcedOff) status	Operation
FALSE	The output signal is turned on/off based on the switch setting. 📖 User's manual for the controller used
TRUE	The output signal is forcibly turned off.

■Forced on (ForcedOn)

The output signal is forcibly turned on during execution.

The output signal is forcibly turned on by setting Forced on (ForcedOn) to TRUE while Forced off (ForcedOff) is TRUE.

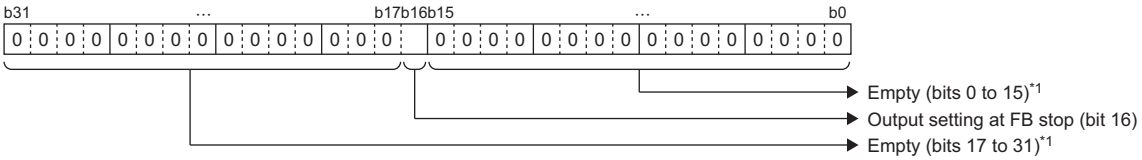
This control has priority over Hysteresis (Switching disabled distance) of Track options (TrackOptions).

Forced on (ForcedOn) status	Operation
FALSE	The output signal is turned on/off based on the switch setting. 📖 User's manual for the controller used
TRUE	The output signal is forcibly turned on.

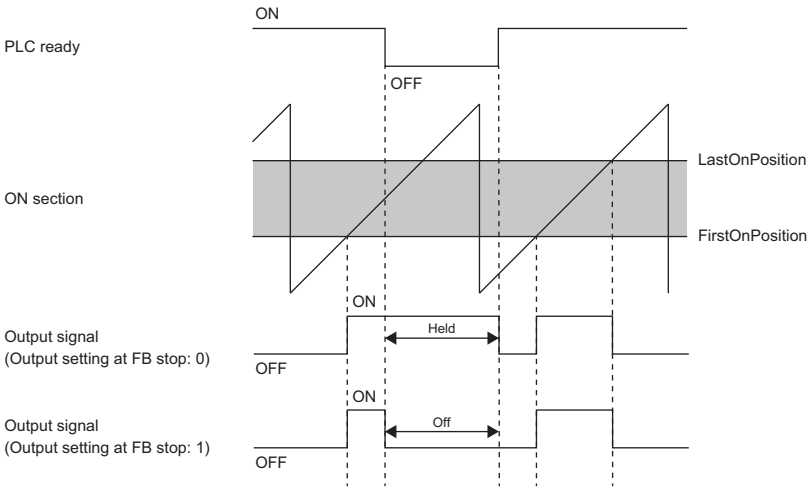
■Options (Options)

The function options for this FB are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to "0". If a value other than "0" is set, an MC_DigitalCamSwitch instruction error (error code: 1BCAH) occurs and the function does not start.

Bit	Name	Description
16	Output setting at FB stop	When Enable (Enable) is changed from TRUE to FALSE, "PLC Ready (MotionSystem.Cd.SequenceReady)" is turned off, or an error occurs, set whether to hold or turn off the output signal when the FB stops. • 0: Hold • 1: Turn off The following figure shows the operation when "PLC Ready (MotionSystem.Cd.SequenceReady)" is switched during execution.  ■Outputs: When [LINK] is specified For details on the output at controller STOP/error, refer to the following. 📖 User's manual for the controller used Since the output to the link device RWw is held, the output hold/clear setting function of the remote I/O has priority. For details, refer to the following. 📖 User's manual for the controller used

Precautions

If the signal set for Output signal (Outputs) differs from Output value (OutputData), check whether the signal set for Output signal (Outputs) is manipulated by another function.

Program example

The following is a program example. Y10 at station address 192.168.4.1 is used as the output signal (Outputs) of Digital Cam Switch Output (MC_DigitalCamSwitch).

■NZ2GN2S1-32DTE setting

Set function setting switches 6 and 7 of the remote I/O module as shown below to configure Synchronous timing control mode.

Function setting switch 6	Function setting switch 7	Operation mode of CC-Link IE TSN network synchronous communication	CC-Link IE TSN Class setting
On	Off	Synchronous timing control mode	CC-Link IE TSN Class B

■Motion system setting

Create labels to be used as the output signals (Outputs) of Digital Cam Switch Output (MC_DigitalCamSwitch).
Using the external signal selection type [LINK], set Y10 of the NZ2GN2S1-32DTE as the Target.

Data name	MC_OUTPUT_REF type data setting			
	Signal.Source.Target	Signal.Detection	Signal.CompensationTime	Signal.FilterTime
Outputs0001	[LINK] Y10@192.168.4.1	— (Setting not required)	0.0	0.0

Create labels to be used as the Data source selection (ValueSource) of Digital Cam Switch Output (MC_DigitalCamSwitch).
Create new data of the MC_CAMSWITCH_SOURCE_REF type and set the data source selection. The following shows the case of setting axis data.

Data name	MC_CAMSWITCH_SOURCE_REF type data setting			
	Source.Target	RingCount_LowerValue	RingCount_UpperValue	SpeedAverageNum
ValueSource0001	[AXIS] mcSetValue@Position	-10000000000.0	10000000000.0	0

Set the compensation time according to the data that was set to Data source selection (ValueSource).
Since the delay time varies depending on the actual system structure, refer to the following for the compensation time.
📖 User's manual for the controller used
The following is an example of setting the compensation time.

Setting item	Setting value
Communication cycle interval setting	2000.00 [μs]
System.PrConst.OperationCycle [1]	0
ValueSource0001.Source.Target	[AXIS] mcSetValue@Position

When the above settings are used

Setting item	Setting value
Outputs.Signal.CompensationTime	-0.002 [s] (-(0.002 [s] (Operation cycle) × 1))

Create operation profile data to be used as the switches (Switches) of Digital Cam Switch Output (MC_DigitalCamSwitch).
For details on how to create operation profile data, refer to the following.
📖 User's manual for the controller used
Create new operation profile data of the MC_CAMSWITCH_REF type and set the switches.

Data name	MC_CAMSWITCH_REF type data setting
	CamSwitchData.ProfileID.Number
CamSwitch0001	1

■Labels used

Label name	Data type	Comment
MC_DigitalCamSwitch_1	MC_DigitalCamSwitch	Digital Cam Switch Output FB
stSwitches	MC_CAMSWITCH_DATA_REF	Switch
stTrackOptions	MC_TRACK_REF	Track options
bEnable	Bit	Enable
bForcedOff	Bit	Forced off
bForcedOn	Bit	Forced on
udOptions	Double word [unsigned]/bit string [32 bits]	Options
bInOperation	Bit	In operation
bBusy	Bit	Executing
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■ST program

```
stSwitches.ProfileID:= CamSwitch0001.ProfileData.ID;

MC_DigitalCamSwitch_1(
    Axis:= Axis0001.AxisRef ,
    Switches:= stSwitches ,
    Outputs:= Outputs0001 ,
    TrackOptions:= stTrackOptions ,
    Enable:= bEnable ,
    ValueSource:= ValueSource0001 ,
    ForcedOff:= bForcedOff ,
    ForcedOn:= bForcedOn ,
    Options:= udOptions ,
    InOperation=> bInOperation ,
    Busy=> bBusy ,
    Error=> bError ,
    ErrorID=> uwErrorID
);
```

41.22 Axis Status Read of Single Axis

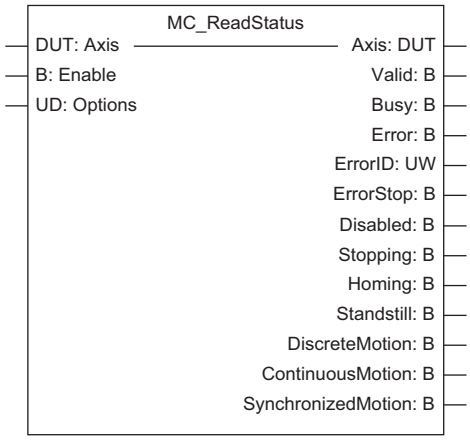
MC_ReadStatus

This FB acquires the state transition status of the specified axis.

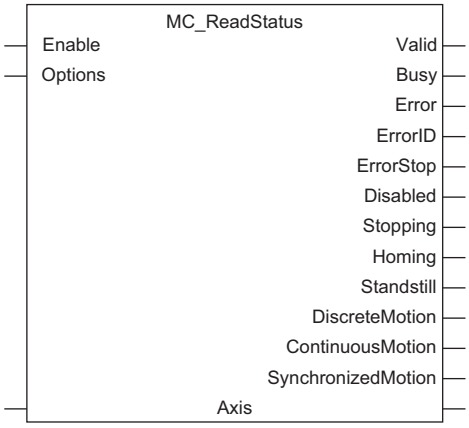
Restriction

When using these instructions, check the versions of the controller and engineering tool used. (MELSEC MX Controller MX-R Model User's Manual)

Ladder



FBD/LD



ST

```
MC_ReadStatus(
   Axis:= ?AXIS_REF? ,
   Enable:= ?BOOL? ,
   Options:= ?DWORD? ,
   Valid=> ?BOOL? ,
   Busy=> ?BOOL? ,
   Error=> ?BOOL? ,
   ErrorID=> ?WORD? ,
   ErrorStop=> ?BOOL? ,
   Disabled=> ?BOOL? ,
   Stopping=> ?BOOL? ,
   Homing=> ?BOOL? ,
   Standstill=> ?BOOL? ,
   DiscreteMotion=> ?BOOL? ,
   ContinuousMotion=> ?BOOL? ,
   SynchronizedMotion=> ?BOOL?
);
```

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Output update	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information) Set AxisName.AxisRef. of the axis variable/axes group variable created using the engineering tool. If an axis variable other than the above is set, Out of axis No. range (error code: 1B82H) occurs.

■Input variable

Input variable	Name	Data type	Input	Setting range	Default value	Description
Enable	Enable	BOOL	Always	TRUE, FALSE	FALSE	This FB is executed while the value is TRUE.
Options	Options	DWORD(HEX)	At start	—	00000000H	The function options are set. For details, refer to the following. ☞ Page 1600 Options (Options)

■Output variables

Output variable	Name	Data type	Default value	Description
Valid	Output value valid	BOOL	FALSE	This variable becomes TRUE when the output value is valid.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when this FB is being executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used
ErrorStop	ErrorStop status	BOOL	FALSE	This variable becomes TRUE when Axis status is ErrorStop.
Disabled	Disabled status	BOOL	FALSE	This variable becomes TRUE when Axis status is Disabled.
Stopping	Stopping status	BOOL	FALSE	This variable becomes TRUE when Axis status is Stopping.
Homing	Homing status	BOOL	FALSE	This variable becomes TRUE when Axis status is Homing.
Standstill	Standstill status	BOOL	FALSE	This variable becomes TRUE when Axis status is Standstill.
DiscreteMotion	DiscreteMotion status	BOOL	FALSE	This variable becomes TRUE when Axis status is DiscreteMotion.
ContinuousMotion	ContinuousMotion status	BOOL	FALSE	This variable becomes TRUE when Axis status is ContinuousMotion.
SynchronizedMotion	SynchronizedMotion status	BOOL	FALSE	This variable becomes TRUE when Axis status is SynchronizedMotion.

Processing details

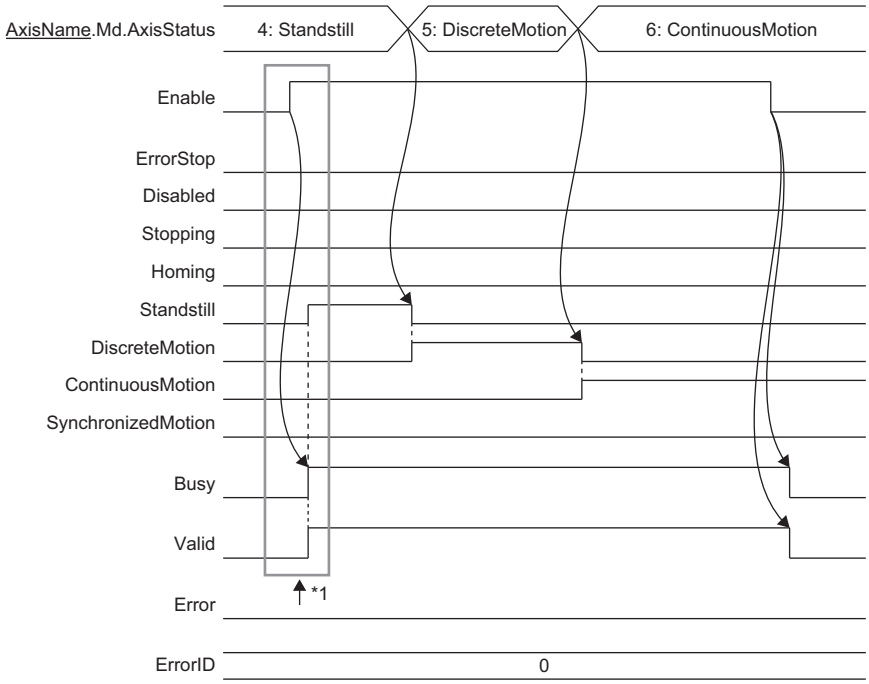
When Enable is set to TRUE, MC_ReadStatus (Axis Status Read of Single Axis) reads the Axis status (AxisName.Md.AxisStatus).

The output variable corresponding to the axis status that was read becomes TRUE. For each of the axis statuses, refer to the following.

📖 User's manual for the controller used

■Timing chart

• When the processing is completed successfully



*1 The axis status is acquired in the cycle where Enable is set to TRUE.

• When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1390 Basic operation of Enable (Enable) type motion control FBs

■Options (Options)

"0" must be set. If a value other than "0" is set, Out of option range (error code: 1BC3H) occurs.

Precautions

The update timing of the data acquired by the motion control FB and of the axis monitor variables differs.
Therefore, when referencing the data acquired by the motion control FB in combination with the axis monitor variables, using a fixed cycle program synchronized with Control Cycle (AxisName.PrConst.OperationCycle) of the target axis is recommended.

Ex.

```
MC_ReadStatus_1(Axis := Axis0001.AxisRef, Enable := TRUE);
IF MC_ReadStatus_1.DiscreteMotion = TRUE THEN
   lePosition := Axis0001.Md.ActualPosition;
END_IF;
```

Program example

The following is a program example. The axis status is monitored with MC_ReadStatus (Axis Status Read of Single Axis), and when it changes to the Standstill state, an arbitrary process is executed.

■Labels used

Label name	Data type	Comment
MC_ReadStatus_1	MC_ReadStatus	Axis Status Read of Single Axis
bEnable	BOOL	Enable

■ST program

```
//----- Axis Status Read of Single Axis -----
MC_ReadStatus_1(Axis := AxisRef, Enable := bEnable);
//Start an arbitrary process when Axis Status becomes Standstill
IF MC_ReadStatus_1.Valid AND MC_ReadStatus_1.StandStill THEN
//Arbitrary process
END_IF;
```

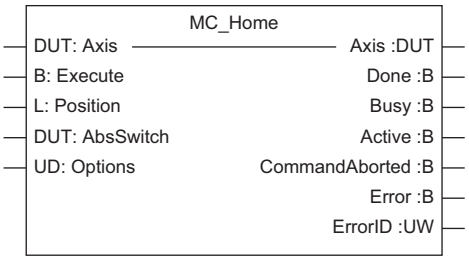
42 MOTION FUNCTION BLOCK

42.1 Homing

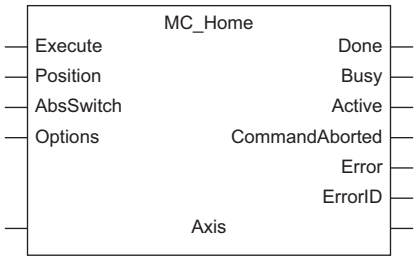
MC_Home

This FB executes homing for the specified axis.

Ladder



FBD/LD



ST

```
MC_Home(
  Axis:= ?AXIS_REF? ,
  Execute:= ?BOOL? ,
  Position:= ?LREAL? ,
  AbsSwitch:= ?MC_INPUT_REF? ,
  Options:= ?DWORD? ,
  Done=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

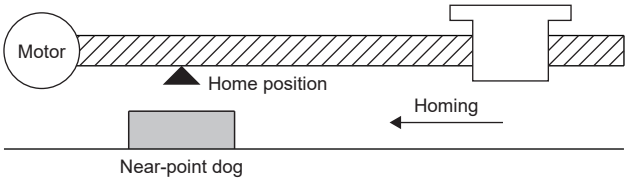
Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_Home (OPR) is executed.
Position	Target position	LREAL	At start	-10000000000.0 to 10000000000.0	0.0	The home position address is set. Set the address within the following range. • -10000000000.0 ≤ Set value < 10000000000.0 *: When the ring counter is enabled, the ring counter range is used.
AbsSwitch	Home position switch	MC_INPUT_REF	At start	—	—	The near-point dog signal that is transmitted to the device is set in the driver homing method. For details, refer to the following. ☞ Page 1606 Home position switch (AbsSwitch)
Options	Options	DWORD(HEX)	At start	00000000H to 00010000H	00000000H	The function options for MC_Home (OPR) are set in the specified bit position. For details, refer to the following. ☞ Page 1607 Options (Options)

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the homing is completed.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_Home (OPR) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MC_Home (OPR) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MC_Home (OPR) is aborted because of an error or multiple start.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

- In homing, a machine home position is established. The address information stored in the motion system and drive unit is not used during this process. The position mechanically established after the homing becomes the start point, or "home position", for positioning control.



- For the homing method, "Driver homing method" and "Data set homing method" are available.
  The homing method used at homing start is determined based on the following conditions.

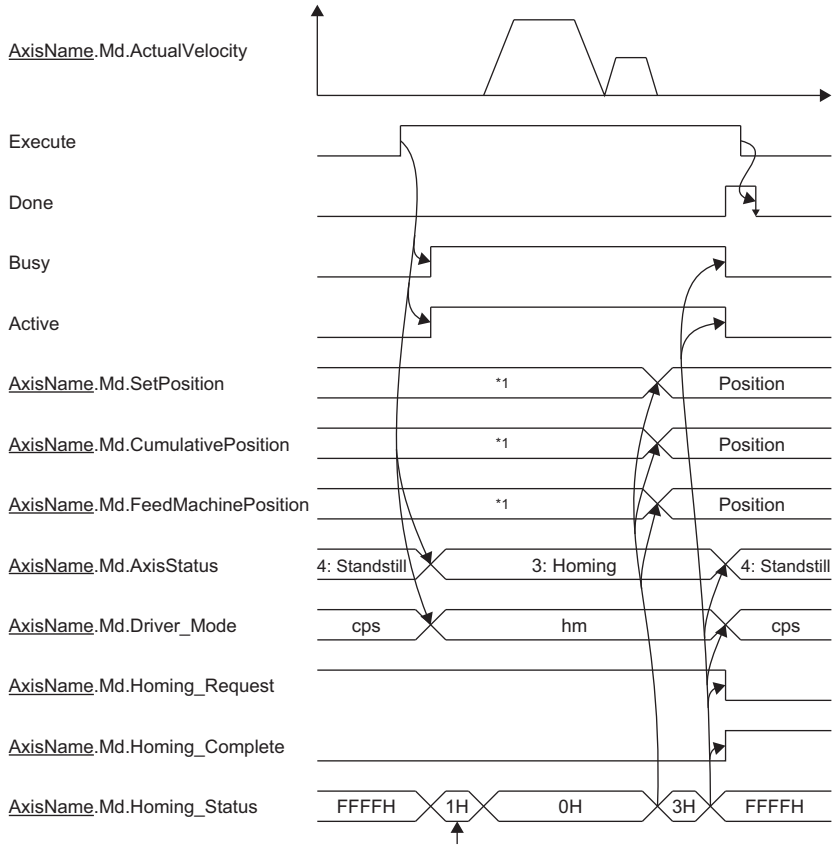
Homing method	Condition for homing method
Driver homing method	Driver homing method is used when all of the following conditions are satisfied. • The axis type is real drive axis. • The drive unit supports Homing mode. • "Home offset (607CH)" is set to object data.
Data set homing method	Data set homing method is used when the conditions listed above are not satisfied.

■Driver homing method

The drive unit is switched to Homing mode, and homing is executed based on the positioning pattern set on the drive unit side.

Change the homing data of the drive unit with MC_WriteParameter (Parameter Write) to change the homing method or each type of parameter. Homing operation and settable parameters vary depending on the specification of the drive unit. For details on the drive unit, refer to the drive unit manual.

- Timing chart
  - Normal operation



The value also becomes "1H" when the home position has already been established.

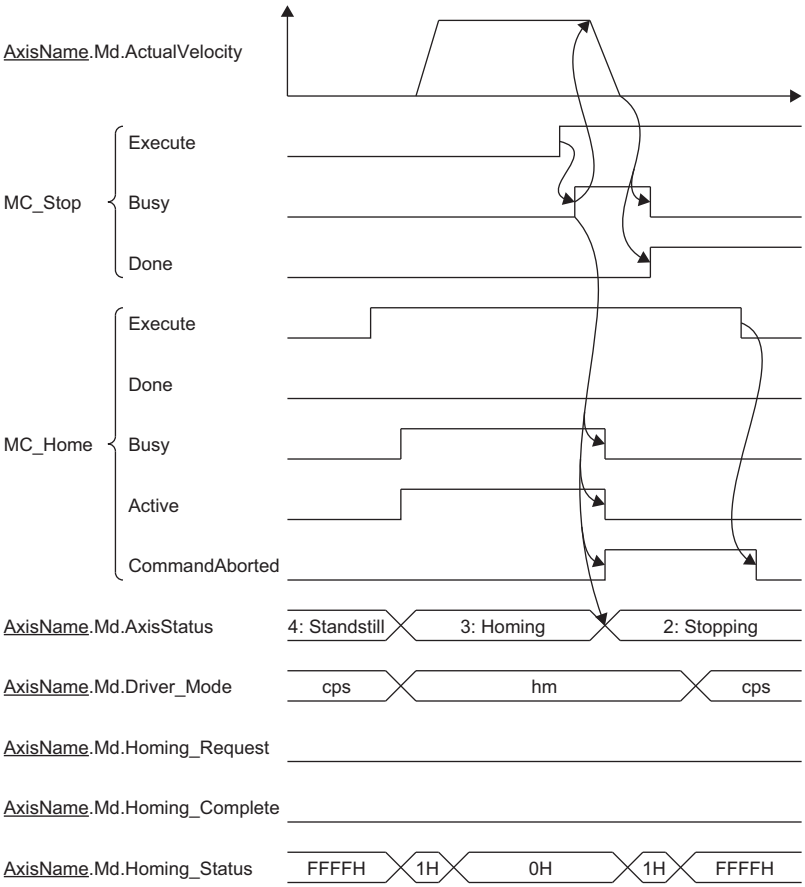
*1: When Follow-up enabled/disabled selection (Options (Options): Bit 16) is set to "0: Follow-up disabled", it will not update during homing. When set to "1: Follow-up enabled", it will update with the current position changes during homing.

- When the operation is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

• When MC_Stop (Forced Stop) is executed

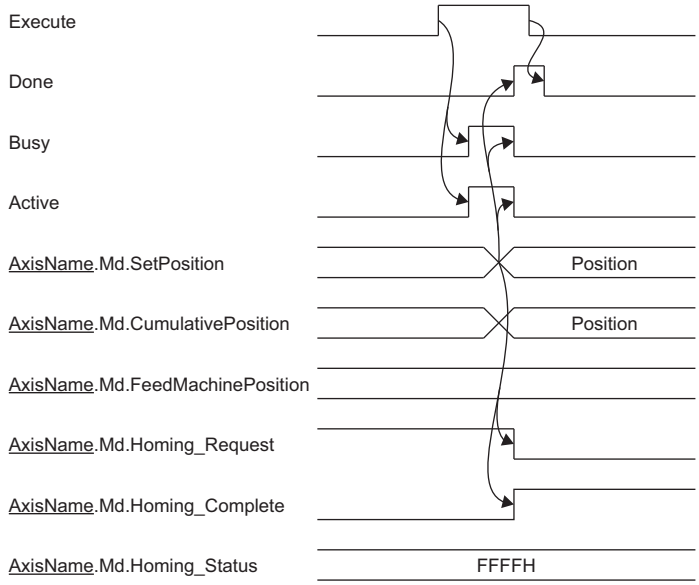


- When MC_Stop (Forced Stop) is set to TRUE when homing, the "HALT" signal is sent to the drive unit. If using a drive unit that does not support HALT, MC_Stop (Forced Stop) will not stop the drive unit, so use the forced stop instead.
- Stop processing at homing depends on the drive unit specifications. Therefore, Deceleration (Deceleration) and Jerk (Jerk) of MC_Stop (Forced Stop) are ignored.

■Data set homing method

Homing is executed for a virtual axis or a real axis that does not have the home position information on the device side. The homing is completed in the motion system, and external signals or others are not used. The Target position (Position) at homing is registered in the motion system as the home position, and Set Position (AxisName.Md.SetPosition) and Cumulative Current Position (AxisName.Md.CumulativePosition) are rewritten to Target position (Position).

- Timing chart
  - Normal operation



  - When the operation is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Homing request

- When it is necessary to execute homing, the motion system changes Home position return request (AxisName.Md.Homing_Request) to TRUE. Home position return request (AxisName.Md.Homing_Request) becomes TRUE in the following cases.
  - When the system's power supply was turned on, or when a reset is performed.*1
  - When the axis type is changed.
  - When homing is started. (The homing request flag will not become FALSE unless homing is completed successfully.)
  - When axis driver unit conversion (numerator/denominator) is changed.
  - When current position restoration is executed.
  - When the absolute position data in the motion system is erased because of a memory error or other cause.
  - When the drive unit's power supply is turned on.*1*2
  - When "Absolute position erased" is detected on the drive unit side.*2
  - When "Polarity (607EH) (b7: position polarity)" of object data is changed*2
  - When the electronic gear is changed on the drive unit side.*2
  - When a drive unit or motor encoder change is detected.
  - When it was detected that the absolute position in the scale measurement encoder is erased.*3
  - When the encoder resolution of the scale measurement encoder is changed.*3
  - When the upper limit value/lower limit value of the encoder ring counter is changed.*4

*1 For an absolute position system, does not become TRUE if homing is completed at reset or system/drive unit power off.
*2 Real drive axis only.
*3 Real encoder axis (via drive module) only.
*4 Virtual encoder axis only.

Point

When Home Position Return Request (AxisName.Md.Homing_Request) becomes TRUE, it is recorded in the event history.
The details can be checked in the event history.

- Execute a homing request clear in systems that do not require homing. Changing Home position return request clear (AxisName.Cd.Homing_ClearRequest) to TRUE causes Home position return request (AxisName.Md.Homing_Request) to become FALSE.

Point

Directly rewriting Home position return request (AxisName.Md.Homing_Request) to FALSE with a program or others will not clear the homing request in the motion system. Be sure to use Home position return request clear (AxisName.Cd.Homing_ClearRequest) to set the homing request to FALSE.

■Home position switch (AbsSwitch)

The near-point dog signal that is transmitted to the device is set in the driver homing method.

Home position switch (AbsSwitch) is set with an MC_INPUT_REF structure. For details on MC_INPUT_REF structures, refer to the following.

☞ Page 1442 MC_INPUT_REF

The setting range for Input signal (AbsSwitch.Signal) set in Home position switch (AbsSwitch) is shown below.

Trigger signal (TriggerInput.Signal) is SIGNAL_SELECT structure type.

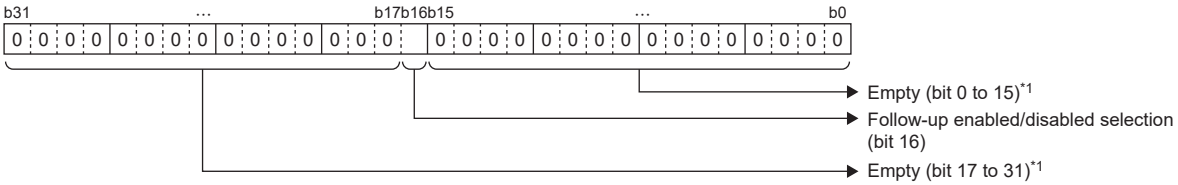
Structure	Variable name	Type	Setting range
SIGNAL_SELECT (Signal select)	Source (Signal)	TARGET_REF ☞ Page 1444 TARGET_REF (Input signal)	■Type • BOOL ■Data category • [OBJ] • [VAR] • [DEV] • [CONST]
	Detection (Signal detection method)	INT (MC_SIGNAL_LOGIC) ☞ Page 1478 MC_SIGNAL_LOGIC	• 0: Detection at TRUE (HighLevel) • 1: Detection at FALSE (LowLevel)
	CompensationTime (Compensation time)	LREAL	0.0 [s]
	FilterTime (Filter time)	LREAL	0.0 to 5.0 [s]

- When [OBJ] is specified, if the target modification is omitted, the object set in the target axis specified by the operand axis (Axis) is referred. However, if the axis set to Axis (Axis) does not have a station address or a station address has not been set, External signal selection incorrect (error code: 1B31H) occurs.
- When the station to be referred when [OBJ] is specified has the Axis Emulate function enabled and also has a set station address, the emulating object will be referred.
- When a signal that cannot be used for Signal name (AbsSwitch.Signal.Source.Target) is used, External signal selection string incorrect (error code: 1B05H), External signal selection incorrect (error code: 1B31H or 1B32H), or Add-on Signal IO internal error (error code: 1B33H) is output.
- If a drive unit other than MELSERVO is specified as the axis, the signal specified by the home position switch cannot be transmitted as the dog signal to the drive unit.

■Options (Options)

The function options for MC_Home (OPR) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to "0". An error occurs if a value other than "0" is set.

Bit	Name	Description
16	Follow-up enabled/disabled selection	Whether the set position follow-up is enabled or disabled during driver homing is set. • 0: Follow-up disabled • 1: Follow-up enabled

Precautions

- FBs other than MC_Stop (Forced Stop) cannot start during homing.
- The software stroke limit check for the Target position (Position) at homing start is shown below.

Homing method	Description
Driver homing method	The software stroke limit check is not performed.
Data set homing method	The software stroke limit check is performed.

- The software stroke limit check of the start position is not performed at homing start.
- The software stroke limit is not checked during homing.
- The movement direction at homing completion is in the positive direction.
- Homing does not support the speed bias function. Set the minimum motor speed at driver-based homing in the drive unit.

■Precautions for the driver homing method

- The homing cannot be started during servo OFF. Thus, the drive unit homing methods (Homing method35, 37 (Data set method)), cannot be executed during servo OFF.
- When the external signals (the hardware stroke limit specified with axis variables and the proximity dog specified with Home position switch (AbsSwitch)) are assigned, the external signals are transmitted to the drive unit. However, when the position accuracy is required in the proximity dog based (not Z-phase based) method, using the drive unit built-in DI is recommended.
- Immediately before the completion of homing, monitor data such as Actual position (AxisName.Md.ActualPosition) may abruptly change, regardless of the actual motor operation. Even if such a phenomenon occurs, the monitor data after the homing will not be affected.
- When Follow-up enabled/disabled selection (Options (Options): Bit 16) is set to "0: Follow-up disabled", the set position is not followed up during homing with the driver homing method. In this case, do not execute synchronous control by setting the axis that is in homing as the master axis. Doing this may cause a Velocity Range Over during Controlling (error code: 1B1AH) to occur in the slave axis.
- When Follow-up enabled/disabled selection (Options (Options): Bit 16) is set to "1: Follow-up enabled", and synchronous control is executed by setting the axis that is in homing as the master axis, the slave axis operations depending on the master axis data source selection are shown below.

Master axis data source selection (MasterValueSource)	Slave axis operation during synchronous control
1: Set value (mcSetValue) 101: Latest set value (mcLatestSetValue)	Synchronous control continues even after homing completes.
2: Actual value (mcActualValue) 102: Latest actual value (mcLatestActualValue)	Although the slave axis synchronizes with the master axis until homing completion, an error or alarm may occur on the drive unit side by reasons of sudden motor acceleration or an excessive command input because synchronous control is not achieved at the end. For details, refer to each drive unit manual. Example: For the MR-J5, the servo alarm ([AL.031.1 (Servo motor speed error)] or [AL. 035.1 (Command frequency error)]) may occur.

- When Home position return operation status (AxisName.Md.Homing_Status) is "0002H: Homing attained (target not reached)", and a stop cause occurs, the set position that has been followed up changes significantly. Therefore, an error or servo alarm may occur on the drive unit side even if "1: Set value (mcSetValue)" or "101: Latest set value (mcLatestSetValue)" is selected in Master axis data source selection (MasterValueSource). For details, refer to each drive unit manual.
  Example: For the MR-J5, the servo alarm ([AL.031.1 (Servo motor speed error)] or [AL. 035.1 (Command frequency error)]) may occur.
- If the following object data items are not mapped, backup is not guaranteed.

Home cycle counter (2D3DH)
Home ABS counter (2D3EH)

- The DOG signal that has been input in the servo amplifier at MR-J5(W)-G connection can be specified as the home position switch signal. For details on the setting method, refer to the following.

📖 User's manual for the controller used

- If the hardware stroke limit signal is detected during homing, the "HALT" signal is sent to the drive unit. To perform a stop from the drive unit side with the limit switch signal, set Hardware stroke limit override (AxisName.Cd.HwStrokeLimit_Override) to "DISABLE (Check disabled)", then temporarily disable the hardware stroke limit check.
- If Hardware stroke limit override (AxisName.Cd.HwStrokeLimit_Override) is set to "ONLY_INSIDE (Check disabled only to inside the range direction)", Start Not Possible (error code: 1AFCH) occurs and homing is not performed.
- When a drive unit other than MELSERVO is used for homing, set the drive unit parameters so that "Home offset (607CH)" becomes 0. If a value other than 0 is set, an unintended operation may occur at homing completion.

Program example

The following is a program example. The homing command (bHomingCMD) is set to TRUE, and homing is performed for Axis 1 (Axis0001) with Home position switch "[DEV](BOOL)D0".

■Axis

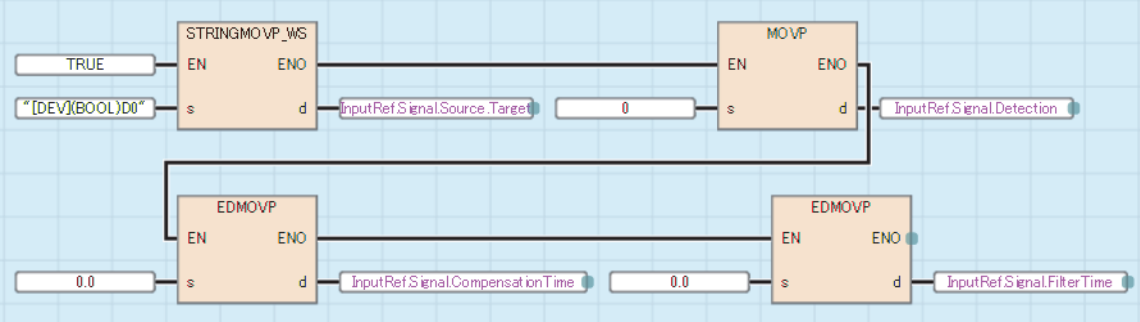
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

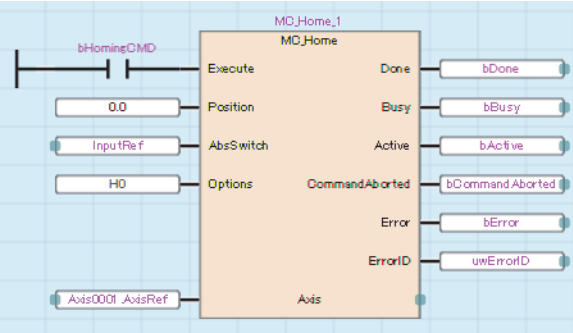
Label name	Data type	Comment
MC_Home_1	MC_Home	OPR FB
bHomingCMD	Bit	Homing command
InputRef	MC_INPUT_REF	Home position switch
bDone	Bit	Execution completion
bBusy	Bit	Executing
bActive	Bit	Controlling
bCommandAborted	Bit	Abortion of execution
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data setting for Home position switch



• Homing



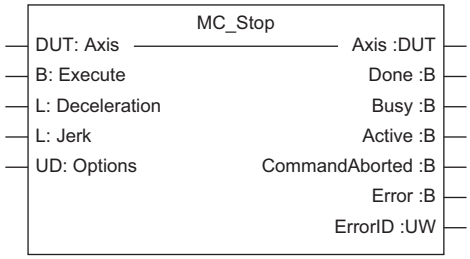
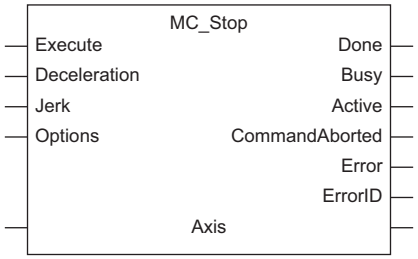
■ST program

```
//-----Data setting for Home position switch-----
InputRef.Signal.Source.Target:= "[DEV](BOOL)D0";
InputRef.Signal.Detection:= MC_SIGNAL_LOGIC__HighLevel;
InputRef.Signal.CompensationTime:= 0.0;
InputRef.Signal.FilterTime:= 0.0;

//-----Homing-----
MC_Home_1(
   Axis:= Axis0001.AxisRef ,
   Execute:= bHomingCMD ,
   Position:= 0.0 ,
   AbsSwitch:= InputRef ,
   Options:= H00000000 ,
   Done=> bDone ,
   Busy=> bBusy ,
   Active=> bActive ,
   CommandAborted=> bCommandAborted ,
   Error=> bError ,
   ErrorID=> uwErrorID
);
```

42.2 Forced Stop

MC_Stop

This FB decelerates the specified axis to a stop.

Ladder	FBD/LD	ST
		MC_Stop(Axis:= ?AXIS_REF? , Execute:= ?BOOL? , Deceleration:= ?LREAL? , Jerk:= ?LREAL? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_Stop (Forced Stop) is executed.
Deceleration	Deceleration	LREAL	At start/ Retrigger possible	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration is set. The setting value differs depending on the acceleration/ deceleration method. For details, refer to "Acceleration/deceleration processing function" in the following. 📖 User's manual for the controller used *: The acceleration/deceleration method takes over the method specified in the on-going control.
Jerk	Jerk	LREAL	At start/ Retrigger possible	0.0	0.0	Set "0.0". *: If a value other than "0.0" is set, Out of Jerk Range (error code: 1A8FH) occurs.
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	Set "00000000H". *: If a value other than "00000000H" is set, Out of Options Range (error code: 1ABBH) occurs.

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the velocity reaches 0.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_Stop (Forced Stop) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MC_Stop (Forced Stop) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MC_Stop (Forced Stop) is aborted.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

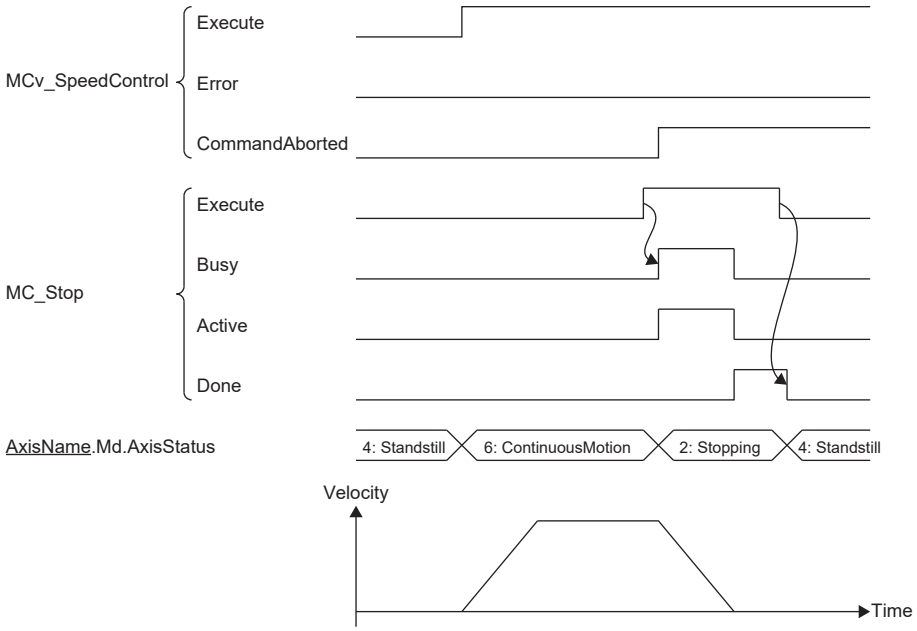
Processing details

- This FB sets Deceleration (Deceleration) and decelerates the under-control FB to a stop.
- When MC_Stop (Forced Stop) is executed, Abortion of execution (CommandAborted) becomes TRUE for the FB being executed and Axis status (AxisName.Md.AxisStatus) transits to "2: Decelerating to Stop (Stopping)". While Execution command (Execute) is TRUE or if the velocity has not reached "0.0", "2: Decelerating to Stop (Stopping)" is maintained. When Execution completion (Done) becomes TRUE and Execution command (Execute) becomes FALSE at stop completion, the axis status changes to "4: Standby (Standstill)".

■Timing chart

- When the processing is completed successfully

When MC_Stop (Forced Stop) is executed during MCv_SpeedControl (Speed Control (Including Position Loop))



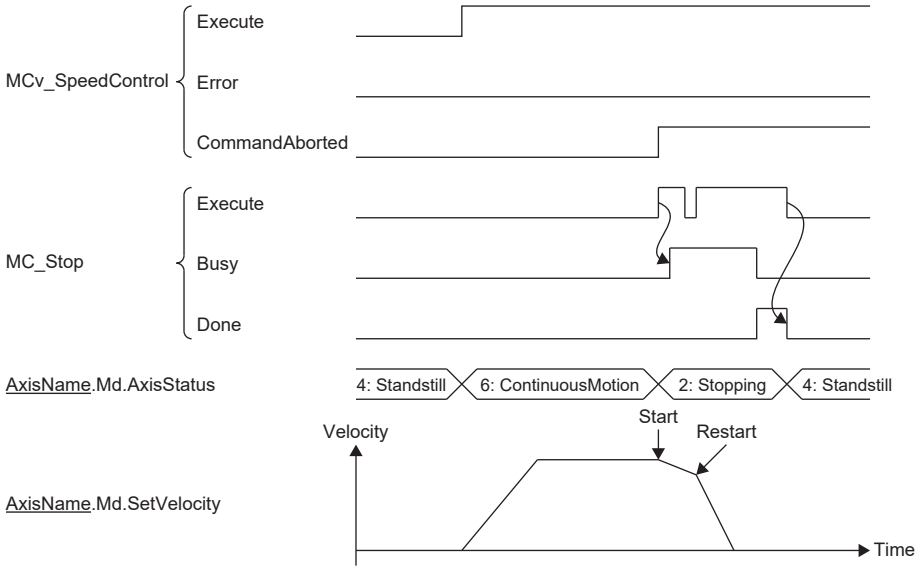
- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

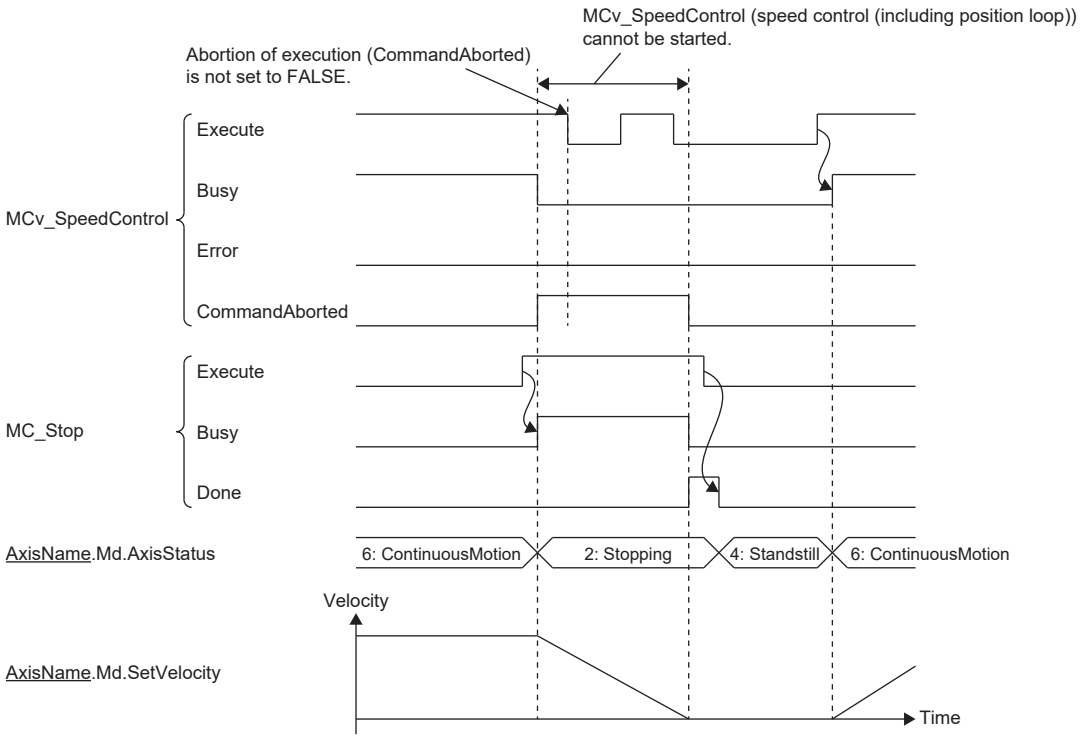
■Operation overview

- When input variable Execution command (Execute) is TRUE during deceleration stop with MC_Stop (Forced Stop), operation commands other than MC_Stop (Forced Stop) are not accepted.
- If Deceleration (Deceleration) is set to "0.0" or omitted, immediate stop will be executed.
- The acceleration/deceleration method and the jerk take over the method specified in the on-going control and decelerate.
- If MC_GroupStop (Group Forced Stop) is executed during single axis synchronous control, synchronization to the master axis will be canceled.
- When the setting value of Deceleration (Deceleration) is changed and MC_Stop (Forced Stop) is restarted, the deceleration stop is executed from the restart point based on the value of Deceleration (Deceleration) in MC_Stop (Forced Stop).



- When an FB that operates axes other than MC_Stop (Forced Stop) is executed while Axis status (AxisName.Md.AxisStatus) is set to "2: Decelerating to Stop (Stopping)", Start Not Possible (error code: 1AFCH) occurs and Error (Error) becomes TRUE for MC_Stop (Forced Stop) and Axis status (AxisName.Md.AxisStatus) changes to "1: Stopping on Error (ErrorStop)".
- After the axis stop, the axis status is transited to "4: Standby (Standstill)" by executing Axis error reset (AxisName.Cd.ErrorReset).

- Even if MC_Stop (Forced Stop) is used to set Execution command (Execute) to FALSE for the FB whose execution was aborted, once the stop operation starts, Abortion of execution (CommandAborted) remains TRUE until the stop is completed. After the stop completes, Abortion of execution (CommandAborted) becomes FALSE if Execution command (Execute) is FALSE for the FB whose execution was aborted.



Program example

The following is a program example. The axis stop request (bStop) is set to TRUE, and Axis 1 (Axis0001) is decelerated to a stop at a deceleration of "100000.0".

■Axis

Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

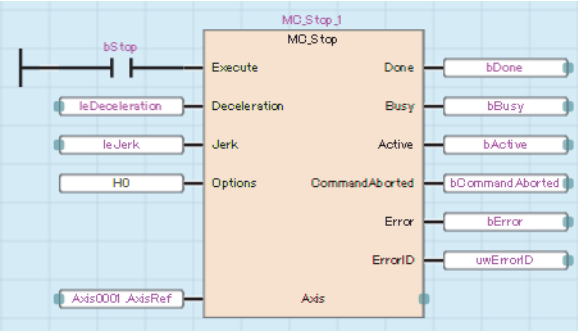
Label name	Data type	Comment
MC_Stop_1	MC_Stop	Forced Stop FB
bStop	Bit	Axis stop request
leDeceleration	Double-precision real number	Deceleration
leJerk	Double-precision real number	Jerk
bDone	Bit	Completion
bBusy	Bit	Executing
bActive	Bit	Controlling
bCommandAborted	Bit	Abortion of execution
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data setting for axis stop



• Axis stop



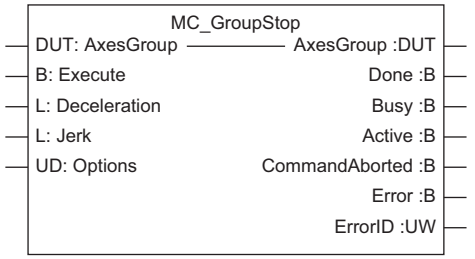
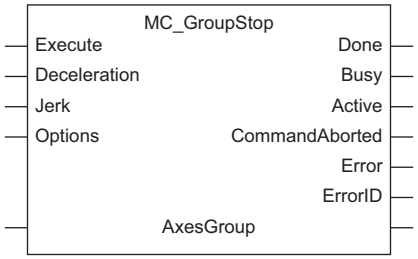
■ST program

```
//-----Data setting for axis stop-----
leDeceleration:= 100000.0;
leJerk:= 0.0;

//-----Axis stop-----
MC_Stop_1(
   Axis:= Axis0001.AxisRef ,
   Execute:= bStop ,
   Deceleration:= leDeceleration ,
   Jerk:= leJerk ,
   Options:= H00000000 ,
   Done=> bDone ,
   Busy=> bBusy ,
   Active=> bActive ,
   CommandAborted=> bCommandAborted ,
   Error=> bError ,
   ErrorID=> uwErrorID
);
```

42.3 Group Forced Stop

MC_GroupStop

Decelerates the specified axes group to a stop.

Ladder	FBD/LD	ST
		MC_GroupStop(AxesGroup:= ?AXES_GROUP_REF? , Execute:= ?BOOL? , Deceleration:= ?LREAL? , Jerk:= ?LREAL? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
AxesGroup	Axes group information	AXES_GROUP_REF	At start	—	Omission not possible	An axes group is set. For the variable (AxesGroupName.AxesGroupRef.) used, refer to the following. ☞ Page 1423 AxesGroupName.AxesGroupRef. (Axes group information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_GroupStop (Group Forced Stop) is executed.
Deceleration	Deceleration	LREAL	At start/ Retrigger possible	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration is set. The setting value differs depending on the acceleration/ deceleration method. For details, refer to "Acceleration/deceleration processing function" in the following. 📖 User's manual for the controller used *: The acceleration/deceleration method takes over the method specified in the on-going control.
Jerk	Jerk	LREAL	At start	0.0	0.0	Set "0.0". *: If a value other than "0.0" is set, Out of Jerk Range (error code: 1A8FH) occurs.
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	Set "00000000H". *: If a value other than "00000000H" is set, Out of Options Range (error code: 1ABBH) occurs.

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the velocity reaches 0.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when operating the velocity toward 0.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MC_GroupStop (Group Forced Stop) is controlling the axes.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MC_GroupStop (Group Forced Stop) is aborted.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

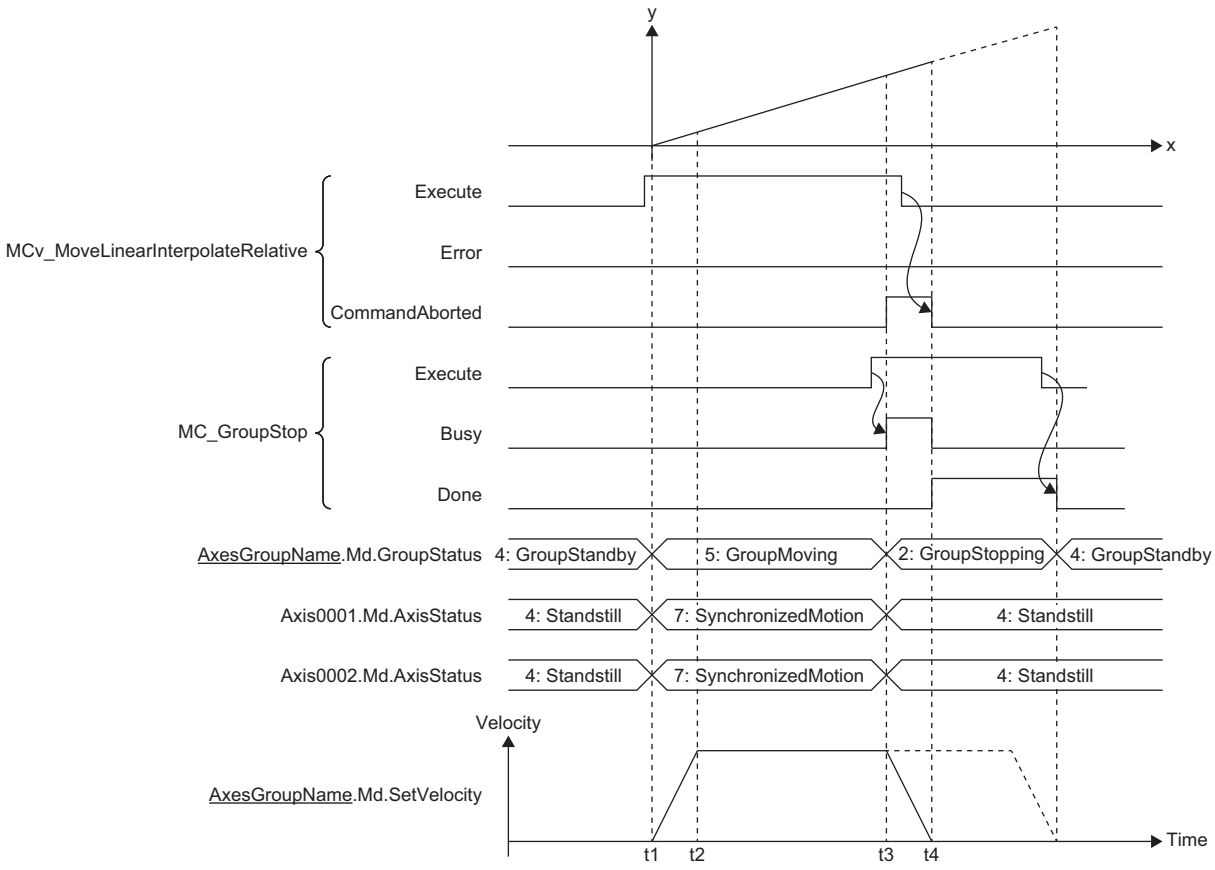
Processing details

- This FB sets Deceleration (Deceleration) and decelerates the under-control FB to a stop. The axes group passes through the previous operation path and decelerates to a stop.
- When MC_GroupStop (Group Forced Stop) is executed, Abortion of execution (CommandAborted) of the FB being executed becomes TRUE and Axes group status (AxesGroupName.Md.GroupStatus) transits to "2: Decelerating to Stop (GroupStopping)". While Execution command (Execute) is TRUE or if the velocity has not reached "0.0", "2: Decelerating to Stop (GroupStopping)" is maintained. When Execution completion (Done) becomes TRUE and Execution command (Execute) becomes FALSE at stop completion, the axis group status changes to "4: Standby (GroupStandby)".

■Timing chart

- When the processing is completed successfully

When MC_GroupStop (Group Forced Stop) is executed during MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control)



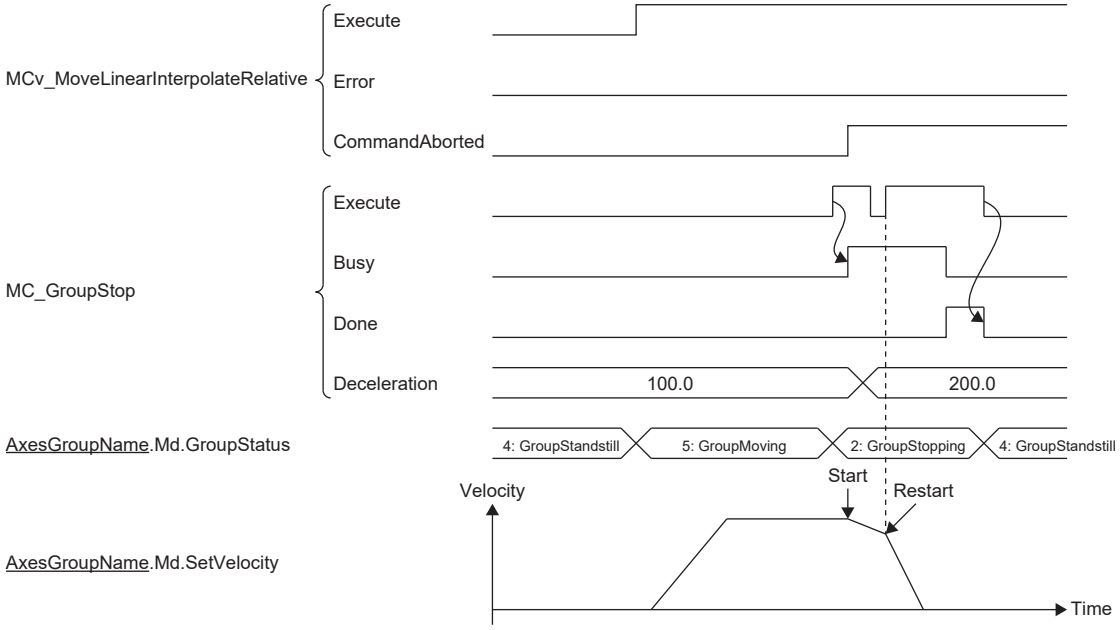
- When the processing is completed with an error

For the timing chart, refer to the following.

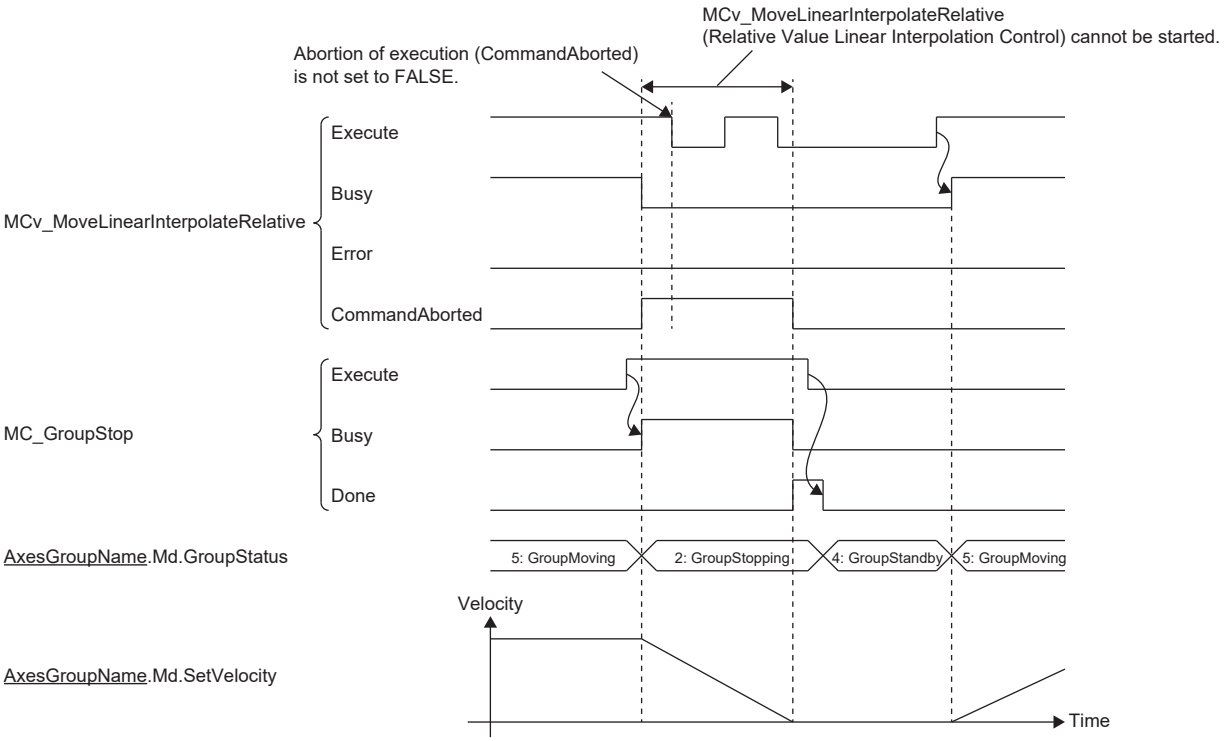
☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Operation overview

- When input variable Execution command (Execute) is TRUE during deceleration stop with MC_GroupStop (Group Forced Stop), operation commands other than MC_GroupStop (Group Forced Stop) are not accepted.
- If Deceleration (Deceleration) is set to "0.0" or omitted, immediate stop will be executed.
- The acceleration/deceleration method and the jerk take over the method specified in the on-going control and decelerate.
- When the setting value of Deceleration (Deceleration) is changed and MC_GroupStop (Group Forced Stop) is restarted, the deceleration stop is executed from the restart point based on the value of Deceleration (Deceleration) in MC_GroupStop (Group Forced Stop) and Jerk (Jerk) of the under-control FB.



- When an FB that operates axes other than MC_GroupStop (Group Forced Stop) is executed while Axes group status (AxesGroupName.Md.GroupStatus) is set to "2: Decelerating to Stop (GroupStopping)", Start Not Possible (error code: 1AFCH) occurs and Error (Error) becomes TRUE for MC_GroupStop (Group Forced Stop) and Axes group status (AxesGroupName.Md.GroupStatus) changes to "1: Stopping on Error (GroupErrorStop)".
- Even if MC_GroupStop (Group Forced Stop) is used to set Execution command (Execute) to FALSE for the FB whose execution was aborted, once the stop operation starts, Abortion of execution (CommandAborted) remains TRUE until the stop is completed. After the stop completes, Abortion of execution (CommandAborted) becomes FALSE if Execution command (Execute) is FALSE for the FB whose execution was aborted.



Program example

The following is a program example. The axes group stop request (bGroupStop) is set to TRUE, and Axes group 1 (AxesGroup001) is decelerated to a stop at a deceleration of "100000.0".

■Axes group

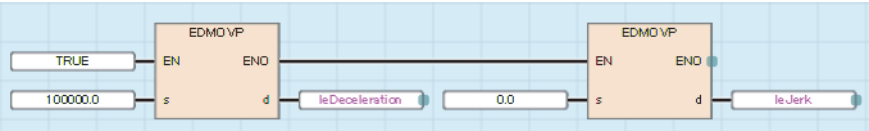
Axes group No.	Label name	Data type	Comment
1	AxesGroup001	AXES_GROUP_REF	Axes group 1

■Labels used

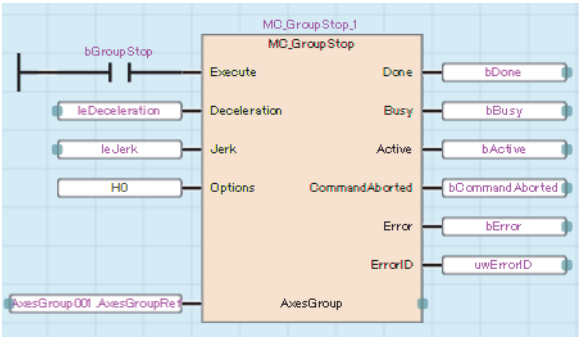
Label name	Data type	Comment
MC_GroupStop_1	MC_GroupStop	Axes group stop FB
bGroupStop	Bit	Axes group stop request
leDeceleration	Double-precision real number	Deceleration
leJerk	Double-precision real number	Jerk
bDone	Bit	Completion
bBusy	Bit	Executing
bActive	Bit	Controlling
bCommandAborted	Bit	Abortion of execution
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data setting for axes group stop



• Axes group stop



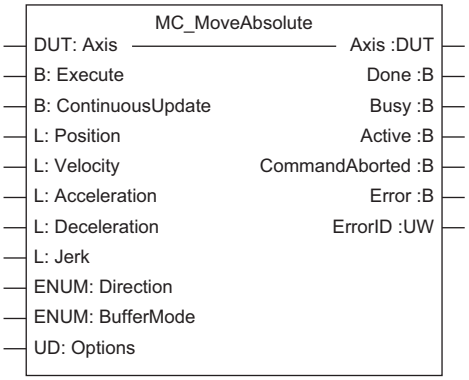
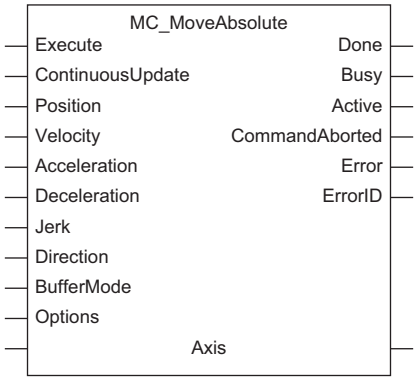
■ST program

```
//-----Data setting for axes group stop-----
leDeceleration:= 100000.0;
leJerk:= 0.0;

//-----Axes group stop-----
MC_GroupStop_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bGroupStop ,
   Deceleration:= leDeceleration ,
   Jerk:= leJerk ,
   Options:= H00000000 ,
   Done=> bDone ,
   Busy=> bBusy ,
   Active=> bActive ,
   CommandAborted=> bCommandAborted ,
   Error=> bError ,
   ErrorID=> uwErrorID
);
```

42.4 Absolute Value Positioning

MC_MoveAbsolute

This FB sets the target position of the absolute position and performs positioning.

Ladder	FBD/LD	ST
		MC_MoveAbsolute(Axis:= ?AXIS_REF? , Execute:= ?BOOL? , ContinuousUpdate:= ?BOOL? , Position:= ?LREAL? , Velocity:= ?LREAL? , Acceleration:= ?LREAL? , Deceleration:= ?LREAL? , Jerk:= ?LREAL? , Direction:= ?INT? , BufferMode:= ?INT? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_MoveAbsolute (Absolute Value Positioning) is executed.
Continuous Update	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	Whether to enable or disable continuous change of Target position (Position), Velocity (Velocity), Acceleration (Acceleration), and Deceleration (Deceleration) is set. • FALSE: Disabled • TRUE: Enabled
Position	Target position	LREAL	At start/Retrigger possible/ Continuous update possible	-10000000000.0 to 10000000000.0	0.0	The target position of the absolute position is set. The settable range varies for each setting. For details, refer to the following. ☞ Page 1625 Target position (Position)
Velocity	Velocity	LREAL	At start/Retrigger possible/ Continuous update possible	0.0, 0.0001 to 2500000000.0	0.0	The velocity is set. For details, refer to the following. ☞ Page 1626 Velocity (Velocity)
Acceleration	Acceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The acceleration is set. For details, refer to the following. ☞ Page 1626 Acceleration (Acceleration)

Input variable	Name	Data type	Input	Valid range	Default value	Description
Deceleration	Deceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration is set. For details, refer to the following. ☞ Page 1626 Deceleration (Deceleration)
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	The jerk is set. For details, refer to the following. ☞ Page 1626 Jerk (Jerk)
Direction	Direction selection	INT (MC_DIRECTION)	At start	1 to 3	0	When software stroke limit is disabled, the direction used to move from the current position to the target position is set. • 1: Positive direction (mcPositiveDirection) • 2: Negative direction (mcNegativeDirection) • 3: Shortest path (mcShortestWay) *: If this setting is omitted, Out of Direction Selection Range (error code: 1AA5H) occurs. For details, refer to the following. ☞ Page 1627 Direction selection (Direction)
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0 to 5	0	The buffer mode is set. • 0: Aborting(mcAborting) • 1: Buffered(mcBuffered) • 2: BlendingLow(mcBlendingLow) • 3: BlendingPrevious(mcBlendingPrevious) • 4: BlendingNext(mcBlendingNext) • 5: BlendingHigh(mcBlendingHigh) For details, refer to the following. ☞ Page 1630 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	00000000H to 00010021H	00000000H	The function options for MC_MoveAbsolute (Absolute Value Positioning) are set in the specified bit position. For details, refer to the following. ☞ Page 1631 Options (Options)

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the target position is reached.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_MoveAbsolute (Absolute Value Positioning) is being executed. This variable becomes FALSE after the target position is reached.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MC_MoveAbsolute (Absolute Value Positioning) is controlling the axis. This variable becomes FALSE after the target position is reached.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MC_MoveAbsolute (Absolute Value Positioning) is aborted. The variable becomes FALSE when Execution command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 🕮 User's manual for the controller used

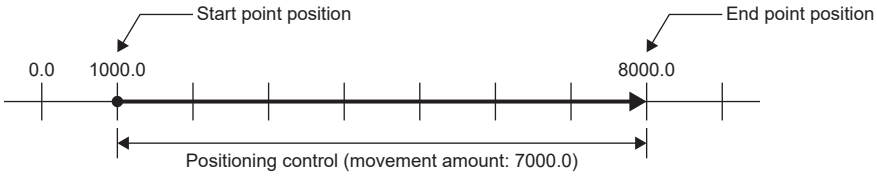
Processing details

- This FB sets Target position (Position), Velocity (Velocity), Acceleration (Acceleration), Deceleration (Deceleration), Jerk (Jerk), Direction selection (Direction), Buffer mode (BufferMode), and Options (Options), and executes positioning from the current position at start (start point position) to the specified position (end point position) set in Target position (Position).

Ex.

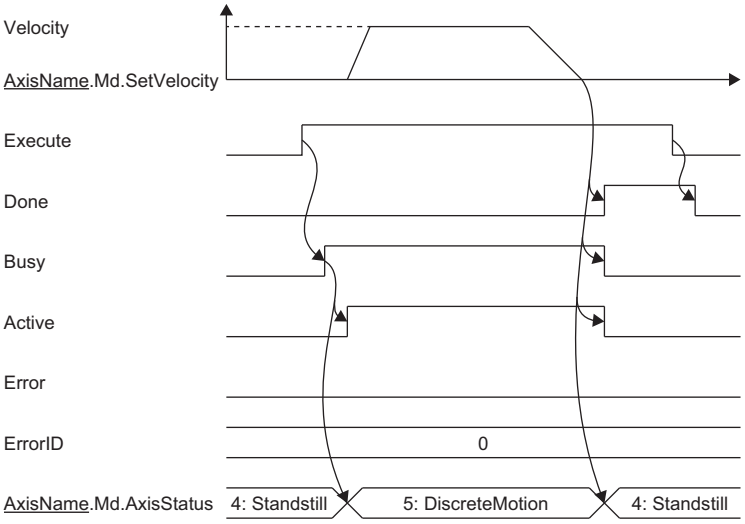
When the start point position (current stop position) is "1000.0" and Target position (Position) is set to "8000.0"

· Positioning is performed in the positive direction for a movement amount of "7000.0 (8000.0 - 1000.0)"



■Timing chart

- When the processing is completed successfully



- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Target position (Position)

The target position of the absolute position is set. The valid range differs depending on whether the software stroke limit is valid/invalid, Direction selection (Direction), and the setting for Target position specification exceeding the ring counter (Options (Options): Bit 16).

- When the software stroke limit is valid (when Software stroke limit target (AxisName.Md.SwStrokeLimit_Target) is set to a value other than "-1: Invalid (Invalid)")

The valid range is as follows regardless of Direction selection (Direction) and the setting for Target position specification exceeding the ring counter (Options (Options): Bit 16).

Valid range
Ring counter lower limit value ≤ Target position < Ring counter upper limit value

If a value outside the range is set, Out of Target Position Range (error code: 1A85H) occurs and the axis will not start. In addition, even when the value set is within the valid range, Software Stroke Limit Over (Target Position) (error code: 1A80H) occurs and the axis will not start if a position which exceeds the software stroke limit is specified as the target position.

- When the software stroke limit is invalid (when Software stroke limit target (AxisName.Md.SwStrokeLimit_Target) is set to "-1: Invalid (Invalid)")

Setting "Target position specification exceeding the ring counter (Options (Options): Bit 16)" to "1: Allow" allows execution of positioning that exceeds the ring counter range by one revolution or more.

If a value outside the range is set, Out of Target Position Range (error code: 1A85H) occurs and the axis will not start.

Direction selection (Direction)	Target position specification exceeding the ring counter (Options (Options): Bit 16)	Valid range
1: Positive direction (mcPositiveDirection)	0: Disabled	Ring counter lower limit value ≤ Target position < Ring counter upper limit value
	1: Enabled	Ring counter lower limit value ≤ Target position < Positioning range upper limit value
2: Negative direction (mcNegativeDirection)	0: Disabled	Ring counter lower limit value ≤ Target position < Ring counter upper limit value
	1: Enabled	Positioning range lower limit value ≤ Target position < Ring counter upper limit value
3: Shortest path (mcShortestWay)	0: Disabled	Ring counter lower limit value ≤ Target position < Ring counter upper limit value
	1: Enabled	Ring counter lower limit value ≤ Target position < Ring counter upper limit value

- For the direction used to move to the target position, refer to the following.

☞ Page 1627 Direction selection (Direction)

■Velocity (Velocity)

The command velocity used in MC_MoveAbsolute (Absolute Value Positioning) is set.

Setting range
0.0, 0.0001 to 2500000000.0[*1*2]

*1 To perform floating-point operation, the lower limit value of the command velocity is limited by the following restrictions.
If the velocity converted from the command velocity to operation cycle is less than 0.00001, Out of Operation Cycle Converted Velocity Range (error code: 1B16H) (when changing the velocity, Out of Operation Cycle Converted Velocity Range Warning (event code: 00D2FH)) occurs. To improve the precision of floating-point operation, change Position command unit (AxisName.Pr.Unit_Position) and Speed command unit (AxisName.Pr.Unit_Velocity) and perform settings so that the velocity converted into the operation cycle is not less than "0.00001".

*2 When a motion control FB with a commanded velocity of "0.0" performs a multiple start, the commanded velocity of the previous motion control FB is used.

■Acceleration (Acceleration)

The acceleration used in MC_MoveAbsolute (Absolute Value Positioning) is set.

The setting range differs depending on the setting for Acceleration/deceleration method setting (Options (Options): Bit 0 to 2).

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	0.000000[*1], positive numbers from 0.000001 to 8400.0 [s]

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Deceleration (Deceleration)

The deceleration used in MC_MoveAbsolute (Absolute Value Positioning) is set.

Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Deceleration (Deceleration) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Jerk (Jerk)

The jerk used in MC_MoveAbsolute (Absolute Value Positioning) is set.

Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Jerk (Jerk) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000, positive numbers from 0.0001 to 2147483647.0 [U/s^3]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

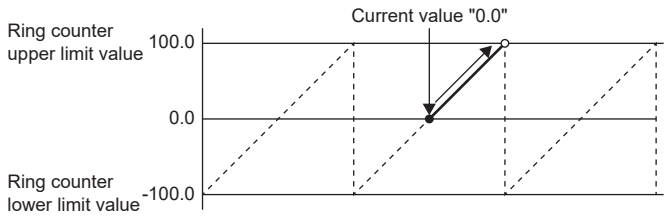
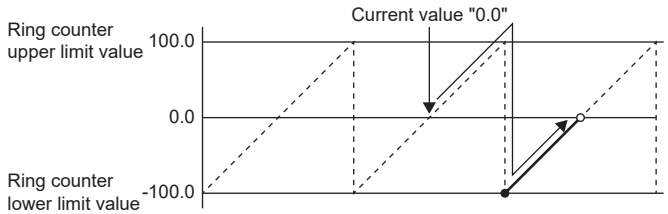
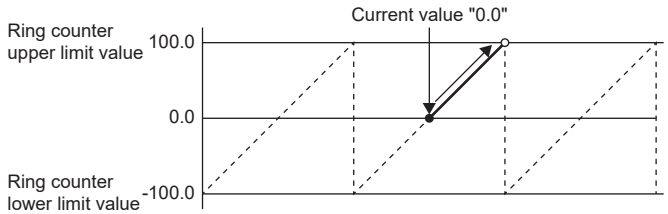
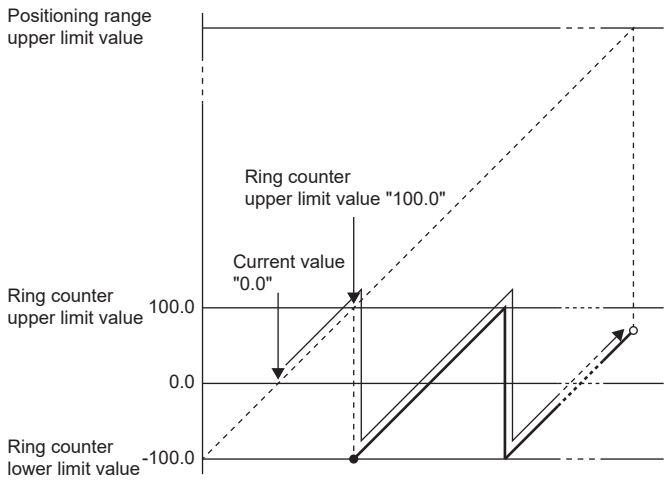
■Direction selection (Direction)

The setting for Direction selection (Direction) is ignored when the software stroke limit is valid. Positioning control is performed in a direction which does not cross over into the area outside the software stroke limit range. However, when the value does not cross over into the area outside the software stroke limit range both in the positive direction and the negative direction, positioning control is performed in the direction closest to the target position (the direction in which the absolute value of the movement amount is shortest) based on the current position. If the distance is the same in both the positive direction and the negative direction, control is performed using the current direction.

When the software stroke limit is invalid, the direction used to move from the current position to the target position can be set by selecting from "1: Positive direction (mcPositiveDirection)", "2: Negative direction (mcNegativeDirection)", or "3: Shortest path (mcShortestWay)" in Direction selection (Direction).

• 1: Positive direction (mcPositiveDirection)

Positioning is performed from the current position to the target position in the positive direction (address increasing).

Target position specification exceeding the ring counter (Options (Options): Bit 16)	Setting range	Operation
■0: Disabled The setting range of the target position is "Ring counter lower limit value ≤ Target position < Ring counter upper limit value".	Current value ≤ Target position < Ring counter upper limit value	
	Ring counter lower limit value ≤ Target position < Current value	
■1: Enabled Allows positioning operation that overflows the ring counter upper limit value to be performed. For this setting, the ring counter upper limit value is used as a reference to perform positioning using the amount of movement exceeded as the relative movement amount. If the target position is set to a value smaller than the ring counter lower limit value, positioning does not start and Out of Target Position Range (error code: 1A85H) occurs.	Current value ≤ Target position < Ring counter upper limit value Ring counter lower limit value ≤ Target position < Current value	
	Ring counter upper limit value ≤ Target position < Positioning range upper limit value	

Ex.

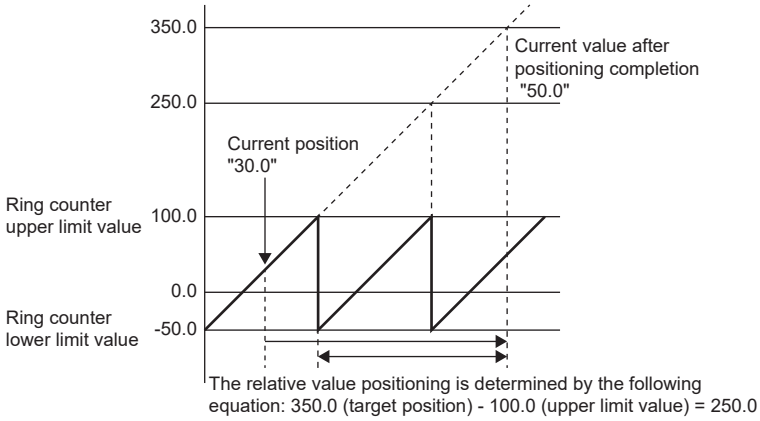
Operation when set to the values shown below (performs relative value positioning after exceeding the ring counter upper limit value).

· Ring counter upper limit value: 100.0

· Ring counter lower limit value: -50.0

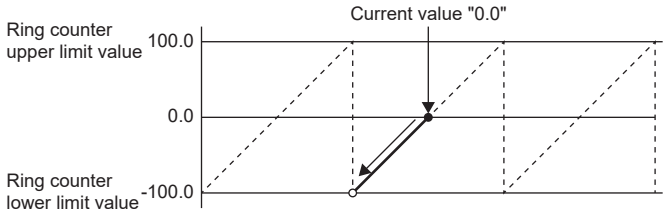
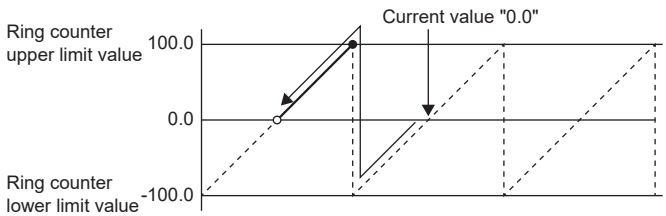
· Current position: 30.0

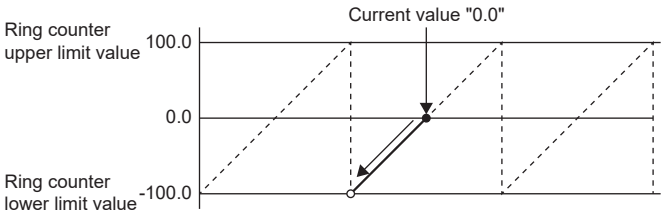
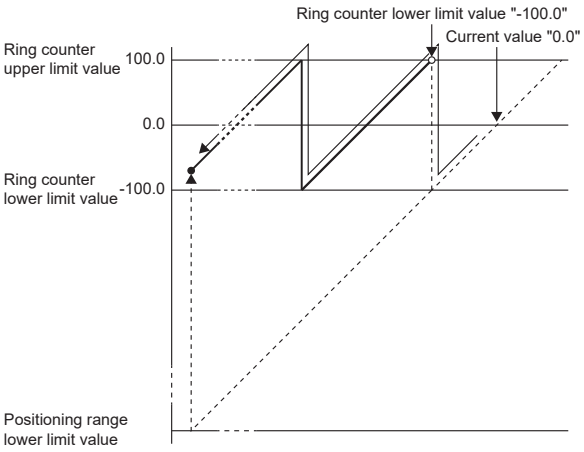
· Target position: 350.0



• 2: Negative direction (mcNegativeDirection)

Positioning is performed from the current position to the target position in the negative direction (address decreasing).

Target position specification exceeding the ring counter (Options (Options): Bit 16)	Setting range	Operation
■0: Disabled The setting range of the target position is "Ring counter lower limit value ≤ Target position < Ring counter upper limit value".	Ring counter lower limit value ≤ Target position ≤ Current value	
	Current value < Target position < Ring counter upper limit value	

Target position specification exceeding the ring counter (Options (Options): Bit 16)	Setting range	Operation
■1: Enabled Allows positioning operation that underflows the ring counter lower limit value to be performed. For this setting, the ring counter lower limit value is used as a reference to perform positioning using the amount of movement exceeded as the relative movement amount. If the target position is set to a value equal to or larger than the ring counter upper limit value, positioning does not start and Out of Target Position Range (error code: 1A85H) occurs.	• Ring counter lower limit value ≤ Target position ≤ Current value • Current value < Target position < Ring counter upper limit value	
	Positioning range lower limit value ≤ Target position < Ring counter lower limit value	

Ex.

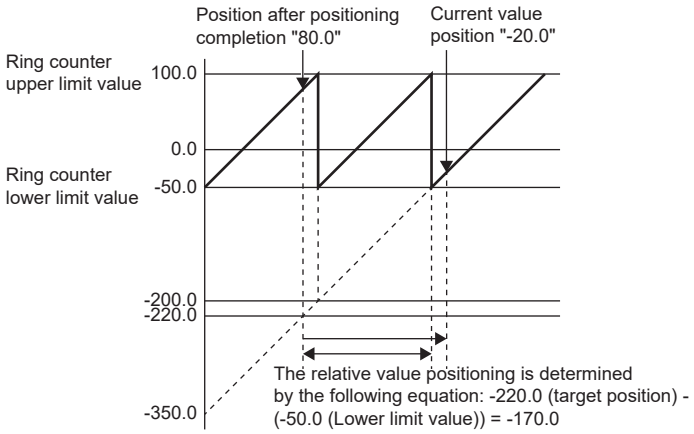
Operation when set to the values shown below

· Ring counter upper limit value: 100.0

· Ring counter lower limit value: -50.0

· Current position: -20.0

· Target position: -220.0



• 3: Shortest path (mcShortestWay)

Based on the current position, positioning control is performed in the direction closest to the target position (the direction in which the absolute value of the movement amount is shortest). If the distance is the same in both the positive direction and the negative direction, control is performed using the current direction. Regardless of the setting for Target position specification exceeding the ring counter (Options (Options): Bit 16), the setting range for the target position is "Ring counter lower limit value ≤ Target position < Ring counter upper limit value". If a value outside the range is set, Out of Target Position Range (error code: 1A85H) occurs and the axis will not start.

Ex.

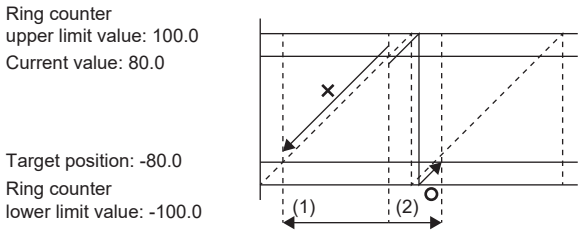
Operation when set to the values shown below

· Ring counter upper limit value: 100.0

· Ring counter lower limit value: -100.0

· Current position: 80.0

· Target position: -80.0



*1 Since the movement amount of (1) is "160.0" and the movement amount of (2) is "40.0", positioning moves in the positive direction from (2).

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following buffer modes are settable in MC_MoveAbsolute (Absolute Value Positioning).

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.
1: Buffered(mcBuffered)	Buffers the next FB on the under-control FB. If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after completion of the under-control FB.
2: BlendingLow(mcBlendingLow)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the lower target velocity between the under-control FB and the buffering FB is used as the switching speed.
3: BlendingPrevious(mcBlendingPrevious)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the under-control FB is used as the switching speed.
4: BlendingNext(mcBlendingNext)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the buffering FB is used as the switching speed.
5: BlendingHigh(mcBlendingHigh)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the higher target velocity between the under-control FB and the buffering FB is used as the switching speed.

*1 Stops are not performed between the under-control FB and the buffering FB.

Point

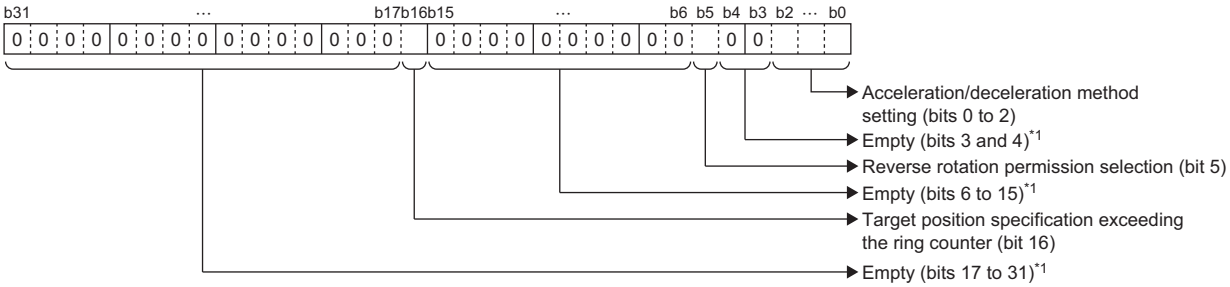
For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

■Options (Options)

The function options for MC_MoveAbsolute (Absolute Value Positioning) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
0 to 2	Acceleration/deceleration method setting	The acceleration/deceleration method used to perform control is set. • 0: Acceleration/deceleration specification method (mcAccDec) • 1: Acceleration/deceleration time-fixed method (mcFixedTime)
5	Reverse rotation permission selection	When the operation directions of the FB being executed and the buffering FB are different, whether to allow reverse rotation is set. • 0: Enabled • 1: Disabled *: This bit is valid when switching to the subsequent buffering FB during axis operation or when changing the target position.
16	Target position specification exceeding the ring counter	When the software stroke limit is invalid, whether to allow a target position that exceeds the ring counter upper limit value or lower limit value is set. • 0: Disabled • 1: Enabled

• Acceleration/deceleration method setting (bit 0 to 2)

Setting value	Description
0: Acceleration/deceleration specification method (mcAccDec)	An acceleration/deceleration method using Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk) set in MC_MoveAbsolute (Absolute Value Positioning).
1: Acceleration/deceleration time-fixed method (mcFixedTime)	An acceleration/deceleration method using the acceleration/deceleration time specified in MC_MoveAbsolute (Absolute Value Positioning) regardless of the velocity. The acceleration/deceleration time does not use Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk).

• Reverse rotation permission selection (Bit 5)

Setting value	Description
0: Enabled	Allows reverse rotation. A deceleration stop is performed once, then operation starts to the changed direction once the deceleration stop is completed.
1: Disabled	Does not allow reverse rotation. When switched, Overrun Error (error code: 1AE3H) occurs and a deceleration stop is performed.

■Necessary object data

When using MC_MoveAbsolute (Absolute Value Positioning), set the following object data for the axis.

• Target position (607AH)

If the object data is not set, Necessary Object Data Unset (error code: 1AF7H) occurs and the axis will not start.

For details on object data settings, refer to the following.

📖 User's manual for the controller used

Program example

The following is a program example. The single axis absolute value positioning start (bMoveAbsolute) is set to TRUE, and absolute positioning control is performed for Axis 1 (Axis0001).

- Operation



- Settings

Item	Setting value
Target position	80000.0
Velocity	50000.0
Acceleration	100000.0
Deceleration	100000.0
Jerk	0.0

■Axis

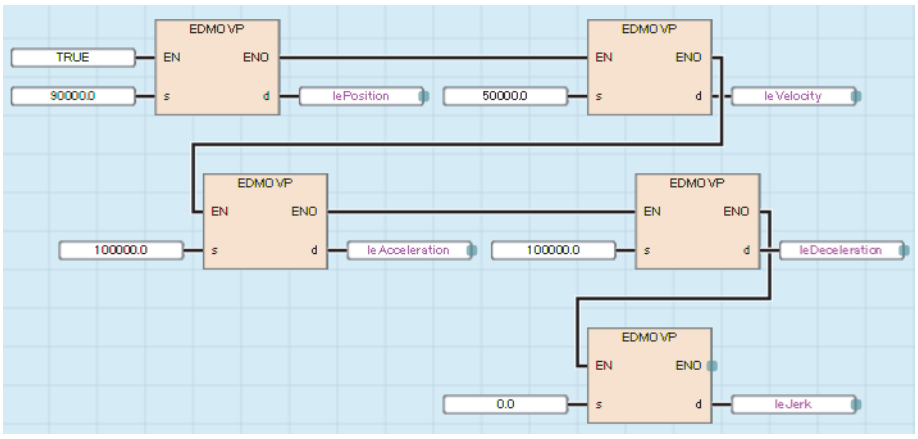
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

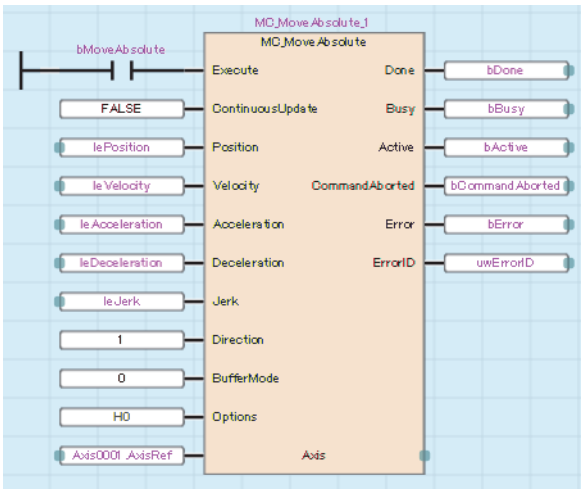
Label name	Data type	Comment
MC_MoveAbsolute_1	MC_MoveAbsolute	Absolute Value Positioning FB
bMoveAbsolute	Bit	Single axis absolute value positioning start
lePosition	Double-precision real number	Target position
leVelocity	Double-precision real number	Velocity
leAcceleration	Double-precision real number	Acceleration
leDeceleration	Double-precision real number	Deceleration
leJerk	Double-precision real number	Jerk
bDone	Bit	Execution completion
bBusy	Bit	Executing
bActive	Bit	Controlling
bCommandAborted	Bit	Abortion of execution
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data setting for single axis positioning control



• Absolute Value Positioning



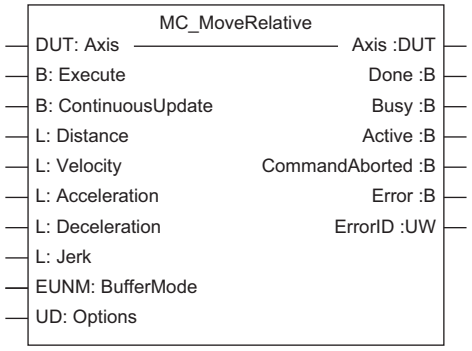
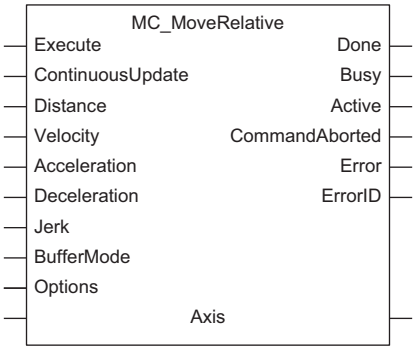
■ST program

```
//-----Data setting for single axis positioning control-----
lePosition:= 80000.0;
leVelocity:= 50000.0;
leAcceleration:= 100000.0;
leDeceleration:= 100000.0;
leJerk:= 0.0;

//-----Absolute Value Positioning-----
MC_MoveAbsolute_1(
   Axis:= Axis0001.AxisRef ,
   Execute:= bMoveAbsolute ,
   ContinuousUpdate:= FALSE ,
   Position:= lePosition ,
   Velocity:= leVelocity ,
   Acceleration:= leAcceleration ,
   Deceleration:= leDeceleration ,
   Jerk:= leJerk ,
   Direction:= MC_DIRECTION__mcPositiveDirection ,
   BufferMode:= MC_BUFFER_MODE__mcAborting ,
   Options:= H00000000 ,
   Done=> bDone ,
   Busy=> bBusy ,
   Active=> bActive ,
   CommandAborted=> bCommandAborted ,
   Error=> bError ,
   ErrorID=> uwErrorID
);
```

42.5 Relative Value Positioning

MC_MoveRelative

This FB sets the movement amount of the relative position and performs positioning.

Ladder	FBD/LD	ST
		MC_MoveRelative(Axis:= ?AXIS_REF? , Execute:= ?BOOL? , ContinuousUpdate:= ?BOOL? , Distance:= ?LREAL? , Velocity:= ?LREAL? , Acceleration:= ?LREAL? , Deceleration:= ?LREAL? , Jerk:= ?LREAL? , BufferMode:= ?INT? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_MoveRelative (Relative Value Positioning) is executed.
Continuous Update	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	Whether to enable or disable continuous change of Movement amount (Distance), Velocity (Velocity), Acceleration (Acceleration), and Deceleration (Deceleration) is set. • FALSE: Disabled • TRUE: Enabled
Distance	Movement amount	LREAL	At start/Retrigger possible/ Continuous update possible	-10000000000.0 to 10000000000.0	0.0	The relative position from the current position at start to the end point is set. For details, refer to the following. ☞ Page 1638 Movement amount (Distance)
Velocity	Velocity	LREAL	At start/Retrigger possible/ Continuous update possible	0.0, 0.0001 to 2500000000.0	0.0	The velocity is set. For details, refer to the following. ☞ Page 1638 Velocity (Velocity)
Acceleration	Acceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The acceleration is set. For details, refer to the following. ☞ Page 1638 Acceleration (Acceleration)
Deceleration	Deceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration is set. For details, refer to the following. ☞ Page 1639 Deceleration (Deceleration)
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	The jerk is set. For details, refer to the following. ☞ Page 1639 Jerk (Jerk)
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0 to 5	0	The buffer mode is set. • 0: Aborting(mcAborting) • 1: Buffered(mcBuffered) • 2: BlendingLow(mcBlendingLow) • 3: BlendingPrevious(mcBlendingPrevious) • 4: BlendingNext(mcBlendingNext) • 5: BlendingHigh(mcBlendingHigh) For details, refer to the following. ☞ Page 1639 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	00000000H to 00000021H	00000000H	The function options for MC_MoveRelative (Relative Value Positioning) are set in the specified bit position. For details, refer to the following. ☞ Page 1640 Options (Options)

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the relative position is reached.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_MoveRelative (Relative Value Positioning) is executed. This variable becomes FALSE after the relative position is reached.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MC_MoveRelative (Relative Value Positioning) is controlling the axis. This variable becomes FALSE after the relative position is reached.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MC_MoveRelative (Relative Value Positioning) is aborted. The variable becomes FALSE when Execution command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

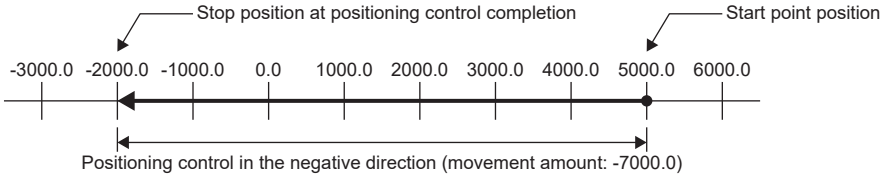
Processing details

- This FB sets Movement amount (Distance), Velocity (Velocity), Acceleration (Acceleration), Deceleration (Deceleration), Jerk (Jerk), and Buffer mode (BufferMode), and executes positioning from the current position at start (start point position) using the movement amount set in Movement amount (Distance). The movement direction is determined by the sign of the movement amount. Axis status (AxisName.Md.AxisStatus) becomes "5: During positioning operation (DiscreteMotion)".

Ex.

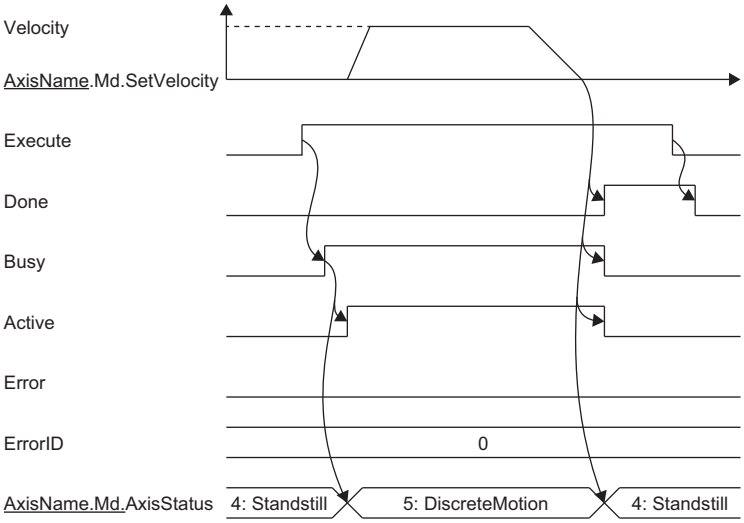
When the start point position (current stop position) is "5000.0" and the movement amount is set to "-7000.0"

· Positioning is performed to "-2000.0".



■Timing chart

- When the processing is completed successfully



- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Movement amount (Distance)

The movement amount from the current position at start to the end point is set.

The movement direction is determined by the sign (+/-) of the movement amount.

Movement amount	Description
If positive (+)	Moves in the positive direction (address increasing direction).
If negative (-)	Moves in the negative direction (address decreasing direction).

When the movement amount is "0", the axis is not operated but Execution completion (Done) becomes TRUE.

If the set movement amount is outside the range, Out of Target Position Range (error code: 1A85H) occurs and the operation will not start.

If "current value + movement amount" overflows the software stroke limit upper limit value or software stroke limit lower limit value when the software stroke limit is valid, Software Stroke Limit Over (target position) (error code: 1A80H) will occur and the operation will not start.

■Velocity (Velocity)

The command velocity used in MC_MoveRelative (Relative Value Positioning) is set.

Setting range
0.0, 0.0001 to 2500000000.0[*1*2]

*1 To perform floating-point operation, the lower limit value of the command velocity is limited by the following restrictions.
If the velocity converted from the command velocity to operation cycle is less than 0.00001, Out of Operation Cycle Converted Velocity Range (error code: 1B16H) (when changing the velocity, Out of Operation Cycle Converted Velocity Range Warning (event code: 00D2FH)) occurs. To improve the precision of floating-point operation, change Position command unit (AxisName.Pr.Unit_Position) and Speed command unit (AxisName.Pr.Unit_Velocity) and perform settings so that the velocity converted into the operation cycle is not less than "0.00001".

*2 When a motion control FB with a commanded velocity of "0.0" performs a multiple start, the commanded velocity of the previous motion control FB is used.

■Acceleration (Acceleration)

The acceleration used in MC_MoveRelative (Relative Value Positioning) is set.

The setting range differs depending on the setting for Acceleration/deceleration method setting (Options (Options): Bit 0 to 2).

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	0.000000[*1], positive numbers from 0.000001 to 8400.0 [s]

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Deceleration (Deceleration)

The deceleration used in MC_MoveRelative (Relative Value Positioning) is set.

Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Deceleration (Deceleration) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000*1, positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Jerk (Jerk)

The jerk used in MC_MoveRelative (Relative Value Positioning) is set.

Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Jerk (Jerk) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000, positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following buffer modes are settable in MC_MoveRelative (Relative Value Positioning).

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.
1: Buffered(mcBuffered)	Buffers the next FB on the under-control FB. If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after completion of the under-control FB.
2: BlendingLow(mcBlendingLow)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the lower target velocity between the under-control FB and the buffering FB is used as the switching speed.
3: BlendingPrevious(mcBlendingPrevious)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the under-control FB is used as the switching speed.
4: BlendingNext(mcBlendingNext)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the buffering FB is used as the switching speed.
5: BlendingHigh(mcBlendingHigh)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the higher target velocity between the under-control FB and the buffering FB is used as the switching speed.

*1 Stops are not performed between the under-control FB and the buffering FB.

Point

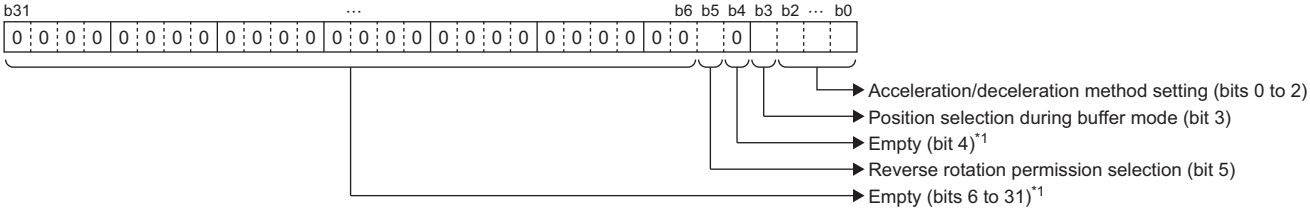
For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

■Options (Options)

The function options for MC_MoveRelative (Relative Value Positioning) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



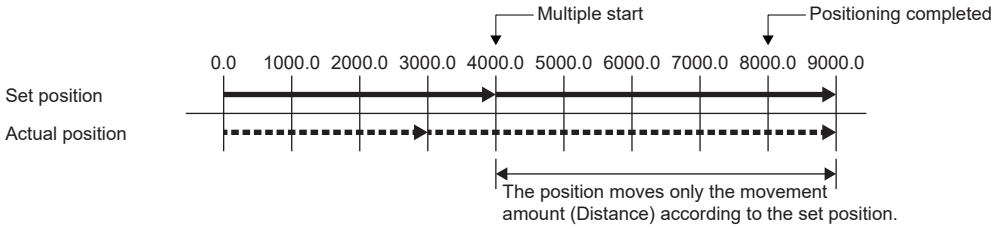
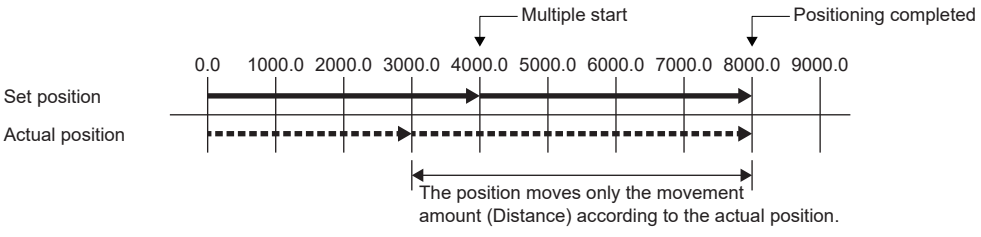
*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
0 to 2	Acceleration/deceleration method setting	The acceleration/deceleration method used to perform control is set. • 0: Acceleration/deceleration specification method (mcAccDec) • 1: Acceleration/deceleration time-fixed method (mcFixedTime)
3	Position selection during buffer mode	The position when executing relative positioning control by multiple start is set. • 0: Set position • 1: Actual position *: This bit is valid when setting "0: Aborting (mcAborting)" in Buffer mode (BufferMode).
5	Reverse rotation permission selection	When the operation directions of the FB being executed and the buffering FB are different, whether to allow reverse rotation is set. • 0: Enabled • 1: Disabled *: This bit is valid when switching to the subsequent buffering FB during axis operation or when changing the target position.

• Acceleration/deceleration method setting (bit 0 to 2)

Setting value	Description
0: Acceleration/deceleration specification method (mcAccDec)	An acceleration/deceleration method using Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk) set in MC_MoveRelative (Relative Value Positioning).
1: Acceleration/deceleration time-fixed method (mcFixedTime)	An acceleration/deceleration method using the acceleration/deceleration time specified in MC_MoveRelative (Relative Value Positioning) regardless of the velocity. The acceleration/deceleration time does not use Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk).

• Position selection during buffer mode (Bit 3)

Setting value	Description
0: Set position	Relative positioning control is performed from the set position. Example: When multiple start is performed with Movement amount (Distance) set to "5000.0" and Options (Options) set to "00000000H (Bit 3 is 0: Set position)" 
1: Actual position	Relative positioning control is performed from the actual position. Example: When multiple start is performed with Movement amount (Distance) set to "5000.0" and Options (Options) set to "00000008H (Bit 3 is 1: Actual position)". 

• Reverse rotation permission selection (Bit 5)

Setting value	Description
0: Enabled	Allows reverse rotation. A deceleration stop is performed once, then operation starts to the changed direction once the deceleration stop is completed.
1: Disabled	Does not allow reverse rotation. When switched, Overrun Error (error code: 1AE3H) occurs and a deceleration stop is performed.

■Necessary object data

When using MC_MoveRelative (Relative Value Positioning), set the following object data for the axis.

• Target position (607AH)

If the object data is not set, Necessary Object Data Unset (error code: 1AF7H) occurs and the axis will not start.

For details on object data settings, refer to the following.

📖 User's manual for the controller used

Precautions

• The current position where MC_MoveRelative (Relative Value Positioning) was started for the first time is used as the start address even if a restart/continuous update is performed.

• Since floating point type processing is used, if relative positioning control is repeatedly executed, the set movement amount may not be reached due to a calculation error.

Program example

The following is a program example. The single axis relative value positioning start (bMoveRelative) is set to TRUE, and relative positioning control is performed for Axis 1 (Axis0001).

• Operation



• Settings

Item	Setting value
Target position	80000.0
Velocity	50000.0
Acceleration	100000.0
Deceleration	100000.0
Jerk	0.0

■Axis

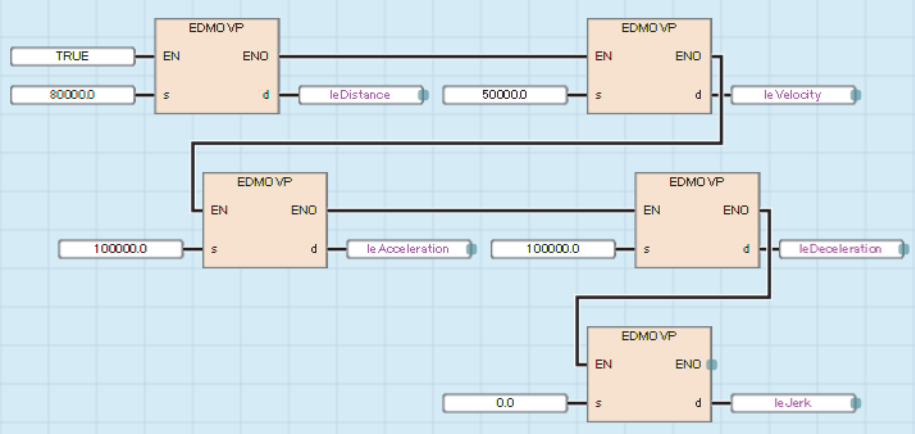
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

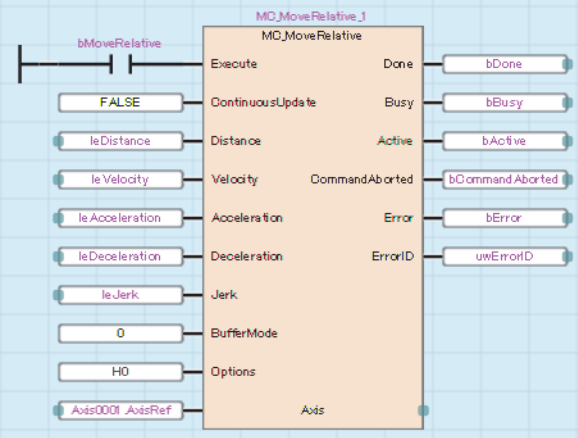
Label name	Data type	Comment
MC_MoveRelative_1	MC_MoveRelative	Relative Value Positioning FB
bMoveRelative	Bit	Single axis relative value positioning start
leDistance	Double-precision real number	Movement amount
leVelocity	Double-precision real number	Velocity
leAcceleration	Double-precision real number	Acceleration
leDeceleration	Double-precision real number	Deceleration
leJerk	Double-precision real number	Jerk
bDone	Bit	Execution completion
bBusy	Bit	Executing
bActive	Bit	Controlling
bCommandAborted	Bit	Abortion of execution
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data setting for single axis positioning control



• Relative Value Positioning



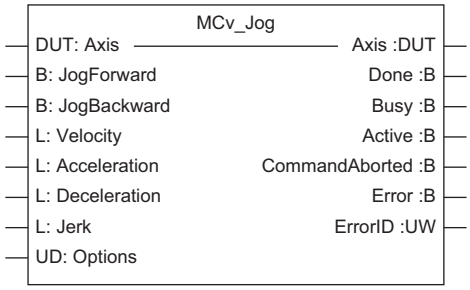
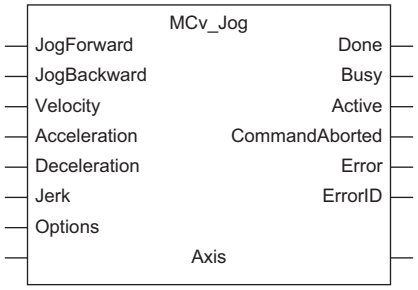
■ST program

```
//-----Data setting for single axis positioning control-----
leDistance:= 80000.0;
leVelocity:= 50000.0;
leAcceleration:= 100000.0;
leDeceleration:= 100000.0;
leJerk:= 0.0;

//-----Relative Value Positioning-----
MC_MoveRelative_1(
    Axis:= Axis0001.AxisRef ,
    Execute:= bMoveRelative ,
    ContinuousUpdate:= FALSE ,
    Distance:= leDistance ,
    Velocity:= leVelocity ,
    Acceleration:= leAcceleration ,
    Deceleration:= leDeceleration ,
    Jerk:= leJerk ,
    BufferMode:= MC_BUFFER_MODE__mcAborting ,
    Options:= H00000000 ,
    Done=> bDone ,
    Busy=> bBusy ,
    Active=> bActive ,
    CommandAborted=> bCommandAborted ,
    Error=> bError ,
    ErrorID=> uwErrorID
);
```

42.6 JOG Operation

MCv_Jog

This FB executes JOG operation according to the command velocity.

Ladder	FBD/LD	ST
		MCv_Jog(Axis:= ?AXIS_REF? , JogForward:= ?BOOL? , JogBackward:= ?BOOL? , Velocity:= ?LREAL? , Acceleration:= ?LREAL? , Deceleration:= ?LREAL? , Jerk:= ?LREAL? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
JogForward	Positive rotation JOG command	BOOL	Always	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_Jog (JOG) is executed in the positive direction. For details, refer to the following. ☞ Page 1646 Positive rotation JOG command (JogForward)/ Reverse rotation JOG command (JogBackward)
JogBackward	Reverse rotation JOG command	BOOL	Always	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_Jog (JOG) is executed in the reverse direction. For details, refer to the following. ☞ Page 1646 Positive rotation JOG command (JogForward)/ Reverse rotation JOG command (JogBackward)
Velocity	Velocity	LREAL	At start	0.0, 0.0001 to 2500000000.0	0.0	The command velocity is set. For details, refer to the following. ☞ Page 1647 Velocity (Velocity)
Acceleration	Acceleration	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	The acceleration is set. For details, refer to the following. ☞ Page 1648 Acceleration (Acceleration)
Deceleration	Deceleration	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration is set. For details, refer to the following. ☞ Page 1648 Deceleration (Deceleration)
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	The jerk is set. For details, refer to the following. ☞ Page 1648 Jerk (Jerk)
Options	Options	DWORD(HEX)	At start	00000000H to 00000001H	00000000H	The function options for MCv_Jog (JOG) are set in the specified bit position. For details, refer to the following. ☞ Page 1649 Options (Options)

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE only for one scan when a deceleration stop is completed by turning off the JOG command.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_Jog (JOG) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MCv_Jog (JOG) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MCv_Jog (JOG) is aborted.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

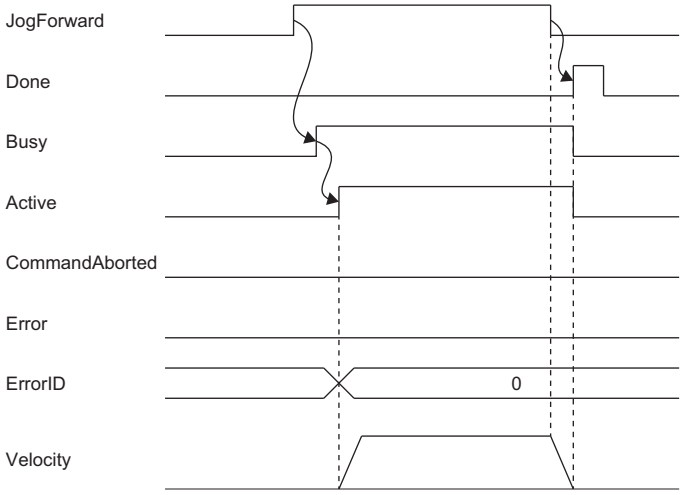
Processing details

- This FB moves the target axis to the specified direction when Positive rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) is set to TRUE.
- Axis status (AxisName.Md.AxisStatus) is "6: During continuous operation (ContinuousMotion)" during JOG operation.
- A deceleration stop is performed when Positive rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) is set to FALSE.
- Axis status (AxisName.Md.AxisStatus) changes to "4: Standby (Standstill)" at deceleration stop completion.
- If Error (Error) becomes TRUE during deceleration by Positive rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) becoming FALSE, Error (Error) remains TRUE until Positive rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) is set to TRUE.
- When another operation FB is started during JOG operation, the operation is performed based on the setting of Buffer mode (BufferMode) of the started operation FB.
- When JOG operation is started during another operation FB, the start request is ignored and Start during Operation Warning (event code: 00D01H) occurs. Start JOG operation when Axis status (AxisName.Md.AxisStatus) is "4: Standby (Standstill)".
- To change the velocity during JOG operation, use the override function to perform the velocity change. For details on the override function, refer to the following.

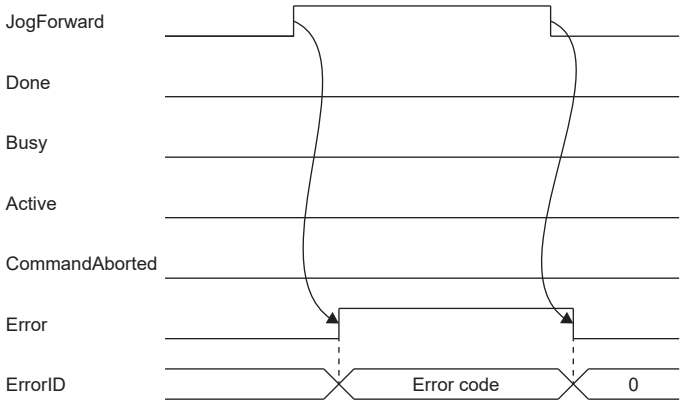
📖 User's manual for the controller used

■Timing chart

- When the processing is completed successfully

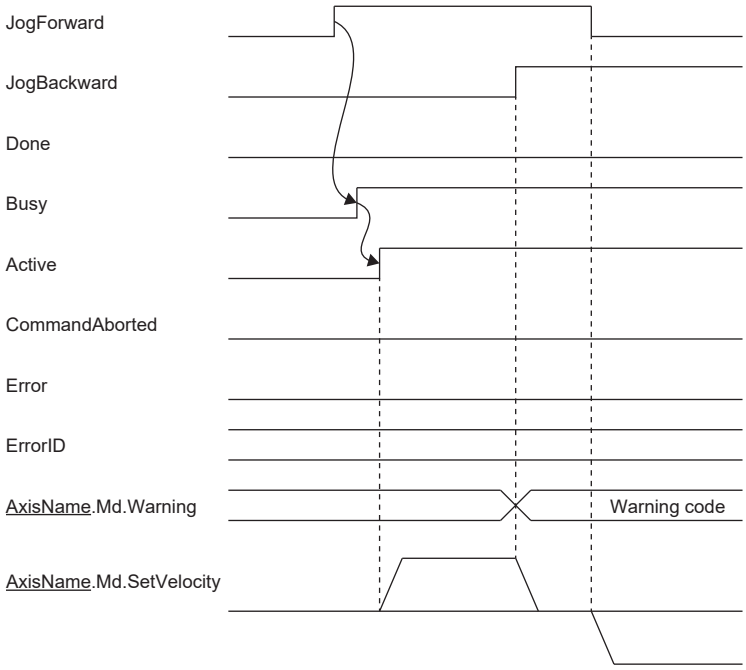


• When the processing is completed with an error

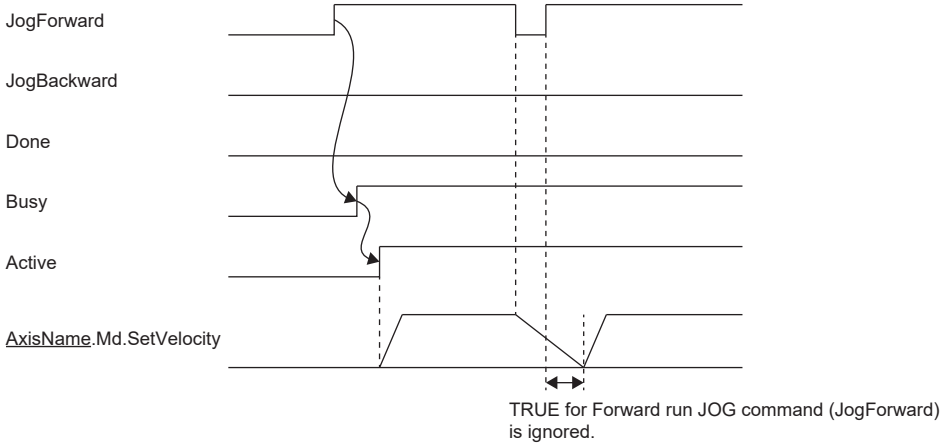


■Positive rotation JOG command (JogForward)/Reverse rotation JOG command (JogBackward)

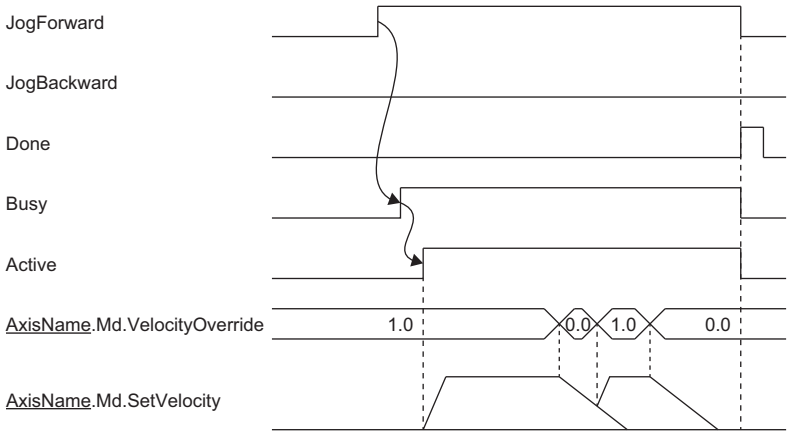
- Setting Positive rotation JOG command (JogForward) to TRUE starts positive rotation JOG operation, and setting Reverse rotation JOG command (JogBackward) to TRUE starts reverse rotation JOG operation, following which the target axis moves in the specified direction.
- If Positive rotation JOG command (JogForward) and Reverse rotation JOG command (JogBackward) are set to TRUE simultaneously, JOG Command Input at Both Directions Warning (event code: 00D02H) occurs, Abortion of execution (CommandAborted) becomes TRUE, and JOG operation will not start. To perform JOG operation, first set the JOG commands for both directions to FALSE and wait for Abortion of execution (CommandAborted) to become FALSE, then set either Positive rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) to TRUE.
- During JOG operation, if the other JOG command (Positive rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward)) of MCv_JOG (JOG) becomes TRUE, JOG Command Input at Both Directions Warning (event code: 00D02H) occurs and a deceleration stop is performed from the point where the said TRUE was detected. When performing JOG operation, only set either Positive rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) to TRUE. To perform JOG operation, only one of Positive rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) must be set to TRUE.



- During the deceleration stop by Positive rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) becoming FALSE, re-acceleration will not be performed even if the JOG command is set to TRUE. The acceleration will be performed after the deceleration stop is completed.



- To repeatedly perform deceleration stop and acceleration while moving by JOG operation, use the override function to perform the velocity change.



■Velocity (Velocity)

The command velocity used in MCv_Jog (JOG) is set.

The settings are imported when JOG operation is started by Positive rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) becoming TRUE. Changes made during JOG operation are not accepted.

Setting range
0.0, 0.0001 to 2500000000.0[*1*2]

*1 To perform floating-point operation, the lower limit value of the command velocity is limited by the following restrictions.
If the velocity converted from the command velocity to operation cycle is less than 0.00001, Out of Operation Cycle Converted Velocity Range (error code: 1B16H) (when changing the velocity, Out of Operation Cycle Converted Velocity Range Warning (event code: 00D2FH)) occurs. To improve the precision of floating-point operation, change Position command unit (AxisName.Pr.Unit_Position) and Speed command unit (AxisName.Pr.Unit_Velocity) and perform settings so that the velocity converted into the operation cycle is not less than "0.00001".

*2 When a motion control FB with a commanded velocity of "0.0" performs a multiple start, the commanded velocity of the previous motion control FB is used.

■Acceleration (Acceleration)

The acceleration used in MCv_Jog (JOG) is set.

The settings are imported when JOG operation is started by Positive rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) becoming TRUE. Changes made during JOG operation are not accepted.

The setting range differs depending on the setting for Acceleration/deceleration method setting (Options (Options): Bit 0 to 2).

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	0.000000[*1], positive numbers from 0.000001 to 8400.0 [s]

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Deceleration (Deceleration)

The deceleration used in MCv_Jog (JOG) is set.

The settings are imported when JOG operation is started by Positive rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) becoming TRUE. Changes made during JOG operation are not accepted.

Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Deceleration (Deceleration) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Jerk (Jerk)

The jerk used in MCv_Jog (JOG) is set.

The settings are imported when JOG operation is started by Positive rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) becoming TRUE. Changes made during JOG operation are not accepted.

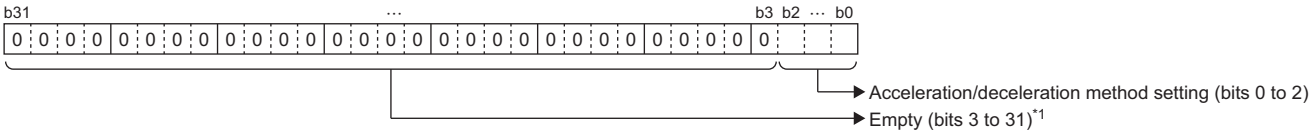
Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Jerk (Jerk) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000, positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

■Options (Options)

The function options for MCv_Jog (JOG) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
0 to 2	Acceleration/deceleration method setting	The acceleration/deceleration method used to perform control is set. • 0: Acceleration/deceleration specification method (mcAccDec) • 1: Acceleration/deceleration time-fixed method (mcFixedTime)

• Acceleration/deceleration method setting (bit 0 to 2)

Setting value	Description
0: Acceleration/deceleration specification method (mcAccDec)	An acceleration/deceleration method using Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk) set in MCv_Jog (JOG).
1: Acceleration/deceleration time-fixed method (mcFixedTime)	An acceleration/deceleration method using the acceleration/deceleration time specified in MCv_Jog (JOG) regardless of the velocity. The acceleration/deceleration time does not use Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk).

■Necessary object data

When using MCv_Jog (JOG), set the following object data for the axis.

• Target position (607AH)

If the object data is not set, Necessary Object Data Unset (error code: 1AF7H) occurs and the axis will not start.

For details on object data settings, refer to the following.

📖 User's manual for the controller used

Precautions

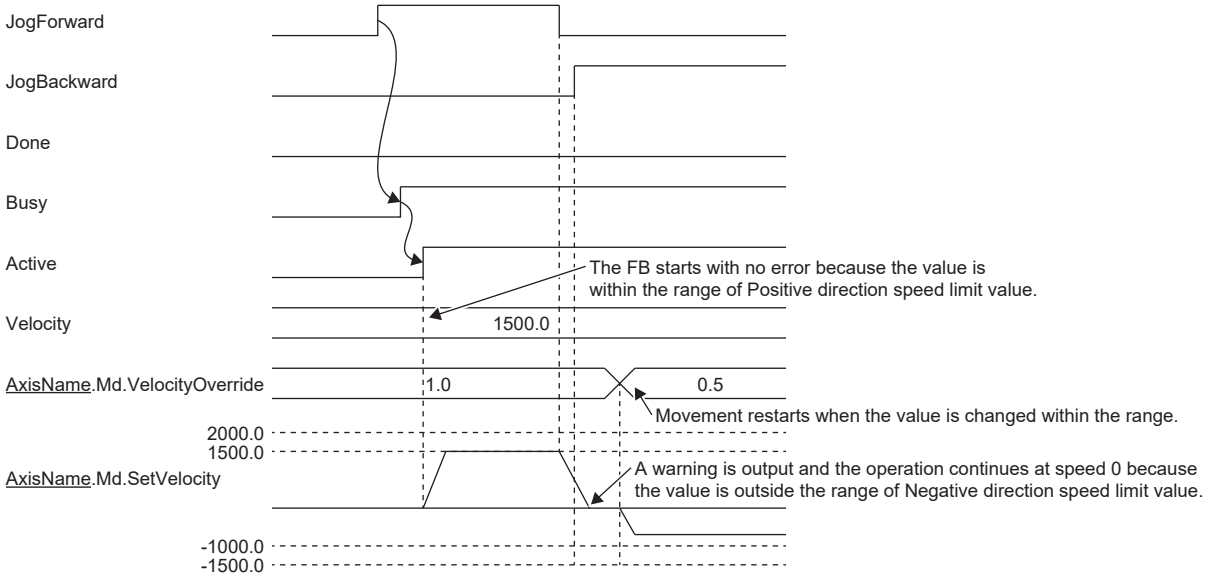
- For safety, first set a small value in Velocity (Velocity) and check the movement, then gradually increase the value.
- Use the hardware stroke limit function when performing JOG operation near the upper limit or lower limit. If the hardware stroke limit function is not used, the workpiece may exceed the moving range and cause an accident.
- When the software stroke limit function is enabled and Software Stroke Limit Override (AxisName.Cd.SwStrokeLimit_Override) is other than DISABLE (Check disabled) or ONLY_INSIDE (Check disabled only to inside the range direction), Software Stroke Limit Override (AxisName.Cd.SwStrokeLimit_Override) is rewritten into ONLY_INSIDE (Check disabled only to inside the range direction) during MCv_Jog (JOG) execution, and is rewritten into a blank area at JOG operation completion.
- When the hardware stroke limit function is enabled and Hardware Stroke Limit Override (AxisName.Cd.HwStrokeLimit_Override) is other than DISABLE (Check disabled) or ONLY_INSIDE (Check disabled only to inside the range direction), Hardware Stroke Limit Override (AxisName.Cd.HwStrokeLimit_Override) is rewritten into ONLY_INSIDE (Check disabled only to inside the range direction) during MCv_Jog (JOG) execution, and is rewritten into a blank area during JOG operation completion.
- Do not change Software Stroke Limit Override (AxisName.Cd.SwStrokeLimit_Override) and Hardware Stroke Limit Override (AxisName.Cd.HwStrokeLimit_Override) during MCv_Jog (JOG) execution.
- When multiple start is executed during JOG operation, the values of Software Stroke Limit Override (AxisName.Cd.SwStrokeLimit_Override) and Hardware Stroke Limit Override (AxisName.Cd.HwStrokeLimit_Override) at the multiple start are retained to the next FB to analyze the next FB.
- When the movement is executed in the opposite direction from that at start after a deceleration stop is performed during JOG operation, and if the command velocity exceeds the velocity limit value, Velocity Limit Value Over Warning on Direction Change (event code: 00D20H) occurs. In addition, when the acceleration time exceeds 8400 seconds, Acceleration Time Over Warning on Direction Change (event code: 00D32H) occurs. (The operation continues at velocity 0.) Movement starts after changing control to remove the cause.

Ex.

When the following values are set

· Positive Direction Speed Limit Value (AxisName.Pr.VelocityLimit_Positive): 2000.0

· Negative Direction Speed Limit Value (AxisName.Pr.VelocityLimit_Negative): 1000.0



Program example

The following is a program example. The JOG positive rotation command (bJogForward) is set to TRUE for positive rotation JOG operation and the JOG reverse rotation command (bJogBackward) is set to TRUE for reverse rotation JOG operation. Then, JOG operation is performed for Axis 1 (Axis0001).

• Setting

Item	Setting value
Velocity	1500.0
Acceleration	2000.0
Deceleration	2000.0
Jerk	0.0

■Axis

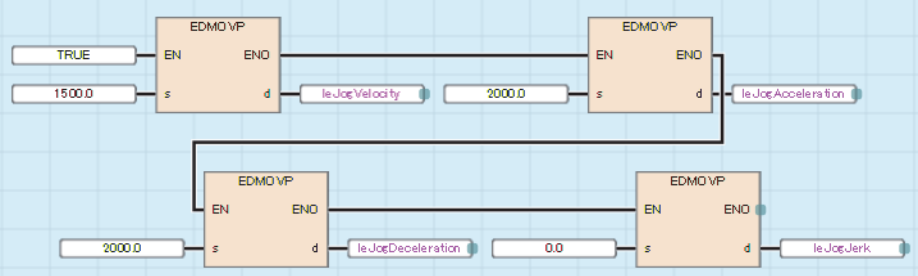
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

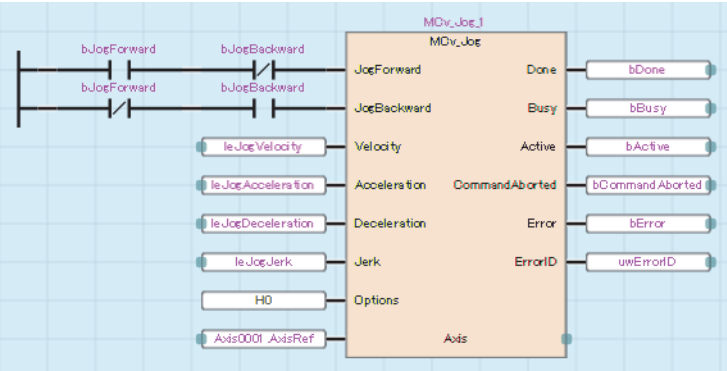
Label name	Data type	Comment
MCv_Jog_1	MCv_Jog	JOG FB
bJogForward	Bit	JOG positive rotation command
bJogBackward	Bit	JOG reverse rotation command
leJogVelocity	Double-precision real number	JOG velocity
leJogAcceleration	Double-precision real number	JOG acceleration
leJogDeceleration	Double-precision real number	JOG deceleration
leJogJerk	Double-precision real number	JOG jerk
bJogF	Bit	JOG positive rotation
bJogB	Bit	JOG reverse rotation
bDone	Bit	Execution completion
bBusy	Bit	Executing
bActive	Bit	Controlling
bCommandAborted	Bit	Abortion of execution
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data settings for JOG operation



• Axis0001 JOG operation



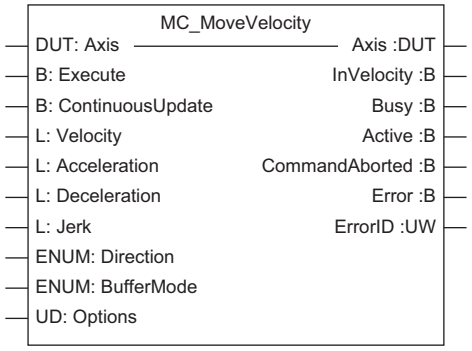
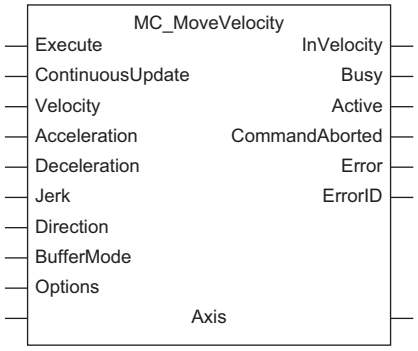
■ST program

```
//-----Data settings for JOG operation-----
leJogVelocity:= 1500.0;
leJogAcceleration:= 2000.0;
leJogDeceleration:= 2000.0;
leJogJerk:= 0.0;

//-----Axis0001 JOG operation-----
bJogForward:= (bJogForward=TRUE) & (bJogBackward=FALSE);
bJogBackward:= (bJogForward=FALSE) & (bJogBackward=TRUE);

MCv_Jog_1(
   Axis:= Axis0001.AxisRef ,
   JogForward:= bJogForward ,
   JogBackward:= bJogBackward ,
   Velocity:= leJogVelocity ,
   Acceleration:= leJogAcceleration ,
   Deceleration:= leJogDeceleration ,
   Jerk:= leJogJerk ,
   Options:= H00000000 ,
   Done=> bDone ,
   Busy=> bBusy ,
   Active=> bActive ,
   CommandAborted=> bCommandAborted ,
   Error=> bError ,
   ErrorID=> uwErrorID
);
```

42.7 Speed Control

MC_MoveVelocity

This FB switches the drive unit to csv and performs velocity control according to the specified velocity.

Ladder	FBD/LD	ST
		MC_MoveVelocity(Axis:= ?AXIS_REF? , Execute:= ?BOOL? , ContinuousUpdate:= ?BOOL? , Velocity:= ?LREAL? , Acceleration:= ?LREAL? , Deceleration:= ?LREAL? , Jerk:= ?LREAL? , Direction:= ?INT? , BufferMode:= ?INT? , Options:= ?DWORD? , InVelocity=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_MoveVelocity (Speed Control) is executed.
Continuous Update	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	Whether to enable or disable continuous change of Velocity (Velocity), Acceleration (Acceleration), and Deceleration (Deceleration) is set. • FALSE: Disabled • TRUE: Enabled
Velocity	Velocity	LREAL	At start/Retrigger possible/ Continuous update possible	0.0, ±0.0001 to ±2500000000.0	0.0	The command velocity is set. When the velocity is negative, the axis moves in the negative direction. When "0.0" is set, the axis does not operate, but Axis status (AxisName.Md.AxisStatus) changes to "6: During continuous operation (ContinuousMotion)". For details, refer to the following. ☞ Page 1656 Velocity (Velocity)
Acceleration	Acceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The acceleration is set. For details, refer to the following. ☞ Page 1656 Acceleration (Acceleration)
Deceleration	Deceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration is set. For details, refer to the following. ☞ Page 1656 Deceleration (Deceleration)
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	The jerk is set. For details, refer to the following. ☞ Page 1656 Jerk (Jerk)
Direction	Direction selection	INT (MC_DIRECTION)	At start	1, 2	0	The direction is set. • 1: Positive direction (mcPositiveDirection) • 2: Negative direction (mcNegativeDirection) *: When "2: Negative direction (mcNegativeDirection)" is set and Velocity (Velocity) is negative, the movement direction of the motor will be positive. *: If this setting is omitted, Out of Direction Selection Range (error code: 1AA5H) occurs.
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0, 1	0	The buffer mode is set. • 0: Aborting(mcAborting) • 1: Buffered(mcBuffered) For details, refer to the following. ☞ Page 1657 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	00000000H to 0002001H	00000000H	The function options for MC_MoveVelocity (Speed Control) are set in the specified bit position. For details, refer to the following. ☞ Page 1657 Options (Options)

■Output variables

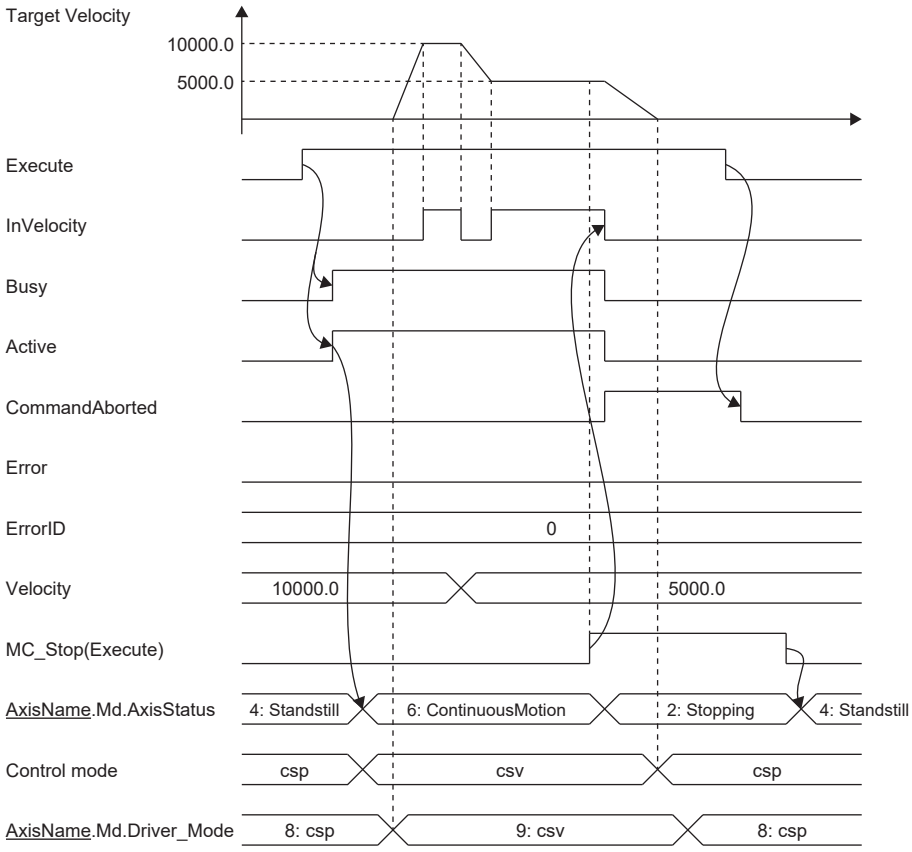
Output variable	Name	Data type	Default value	Description
InVelocity	Target velocity reached	BOOL	FALSE	This variable becomes TRUE when the command velocity calculated by the motion system reaches the target velocity. When the target velocity has been changed due to change at Continuous update (ContinuousUpdate) TRUE, this variable remains FALSE until the velocity reaches the post-change target velocity.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_MoveVelocity (Speed Control) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MC_MoveVelocity (Speed Control) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MC_MoveVelocity (Speed Control) is aborted. The variable becomes FALSE when Execution command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

This FB switches the drive unit control mode to csv and performs control. This function controls the command velocity based on the set Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk). To finish MC_MoveVelocity (Speed Control), start MC_Stop (Forced Stop).

■Timing chart

• When the processing is completed successfully



• When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Velocity (Velocity)

The command velocity used in MC_MoveVelocity (Speed Control) is set.

Setting range
0.0, ±0.0001 to ±2500000000.0[*1*2]

*1 To perform floating-point operation, the lower limit value of the command velocity is limited by the following restrictions.
If the velocity converted from the command velocity to operation cycle is less than 0.00001, Out of Operation Cycle Converted Velocity Range (error code: 1B16H) (when changing the velocity, Out of Operation Cycle Converted Velocity Range Warning (event code: 00D2FH)) occurs. To improve the precision of floating-point operation, change Position command unit (AxisName.Pr.Unit_Position) and Speed command unit (AxisName.Pr.Unit_Velocity) and perform settings so that the velocity converted into the operation cycle is not less than "0.00001".

*2 When a motion control FB with a commanded velocity of "0.0" performs a multiple start, the commanded velocity of the previous motion control FB is used.

■Acceleration (Acceleration)

The acceleration used in MC_MoveVelocity (Speed Control) is set.

The setting range differs depending on the setting for Acceleration/deceleration method setting (Options (Options): Bit 0 to 2).

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	0.000000[*1], positive numbers from 0.000001 to 8400.0 [s]

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Deceleration (Deceleration)

The deceleration used in MC_MoveVelocity (Speed Control) is set.

Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Deceleration (Deceleration) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Jerk (Jerk)

The jerk used in MC_MoveVelocity (Speed Control) is set.

Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Jerk (Jerk) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000, positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following buffer modes are settable in MC_MoveVelocity (Speed Control).

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.
1: Buffered(mcBuffered)	Buffers the next FB on the under-control FB. If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after completion of the under-control FB.

Point

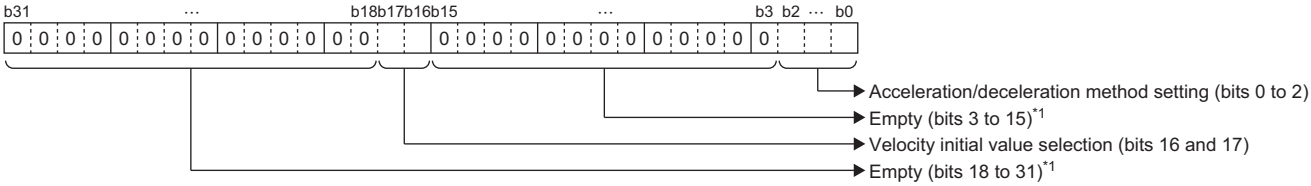
For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

■Options (Options)

The function options for MC_MoveVelocity (Speed Control) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
0 to 2	Acceleration/deceleration method setting	The acceleration/deceleration method used to perform control is set. • 0: Acceleration/deceleration specification method (mcAccDec) • 1: Acceleration/deceleration time-fixed method (mcFixedTime)
16, 17	Velocity initial value selection	The velocity initial value used when changing the control mode from cyclic synchronous position mode (csp) to cyclic synchronous velocity mode (csv) is set. • 0: Command velocity • 1: Actual velocity • 2: Automatic selection

• Acceleration/deceleration method setting (bit 0 to 2)

Setting value	Description
0: Acceleration/deceleration specification method (mcAccDec)	An acceleration/deceleration method using Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk) set in MC_MoveAbsolute (Absolute Value Positioning).
1: Acceleration/deceleration time-fixed method (mcFixedTime)	An acceleration/deceleration method using the acceleration/deceleration time specified in MC_MoveAbsolute (Absolute Value Positioning) regardless of the velocity. The acceleration/deceleration time does not use Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk).

• Velocity initial value selection (Bit 16, 17)

Setting value	Description
0: Command velocity	The velocity during command is used as the velocity commanded to the drive unit immediately after switching.
1: Actual velocity	The motor speed received from the drive unit at switching is used.*1
2: Automatic selection	The lower velocity between "0: Command velocity" and "1: Actual velocity" is used as the velocity commanded to the drive unit immediately after switching.*2

*1 When object data "Velocity actual value (606CH)" is not mapped, the motor speed received from the drive unit at switching is not used. (Speed initial value selection becomes "0".)

*2 When object data "Velocity actual value (606CH)" is not mapped, the velocity commanded to the drive unit immediately after switching becomes "0: Command velocity".

■Related object data

The following functions result in errors or specification restrictions if object data is not set, PDO mapping is not performed, or the device station has no object.

For details on object data settings, refer to the following.

📖 User's manual for the controller used

◎: Required, ○: Recommended

Instruction/function	Required object data	PDO mapping	Operation when object is not set
MC_MoveVelocity (Speed Control)	Target velocity	◎	Necessary Object Data Unset (error code: 1AF7H) occurs.
	Velocity actual value	○	Options (Options): Bits 16 and 17 ☞ Page 1657 Options (Options)

Precautions

- Velocity override factor (AxisName.Cd.VelocityOverride) and Acceleration override factor (AxisName.Cd.AccelerationOverride) are valid.
- The set position and the feed machine position are updated by follow up.
- The switching time of the control mode depends on the specifications of the drive unit.
- When the stop cause occurs during the control mode switching, the axis will stop immediately.
- Do not start a positioning control FB while switching the control mode. Check that Driver control mode (AxisName.Md.Driver_Mode) has switched to "9: Velocity control (csv)" before starting a positioning control FB.
- When switching the control mode from cyclic synchronous position mode (csp) to cyclic synchronous velocity mode (csv), if a stop cause (axis error detection) occurs before the control mode is switched, follow-up may be performed in cyclic synchronous position mode (csp) until zero speed status is reached.
- Note the following precautions when using MR-J5(W)-G and switching from csp to csv without waiting for the motor to stop, or when switching from csv to csp.
  - Set servo parameter (extension setting) "ZSP disabled selection at control switching (PC76.1)" to "1: Disabled (control switching is performed regardless of the range of ZSP)" and disable monitoring of zero speed status. Note that doing so may cause vibration or impact to occur at control mode switching.
  - For the setting values of servo parameters (basic setting) "Electronic gear numerator (PA06)" and "Electronic gear denominator (PA07)", refer to the following.

  📖 User's manual for the controller used

Program example

The following is a program example. The speed control command (bMoveVelocity) is set to TRUE, the mode is switched to velocity control mode, and speed control is performed for Axis 1 (Axis0001).

• Setting

Item	Setting value
Velocity	100000.0
Acceleration	50000.0
Deceleration	50000.0
Jerk	0.0

■Axis

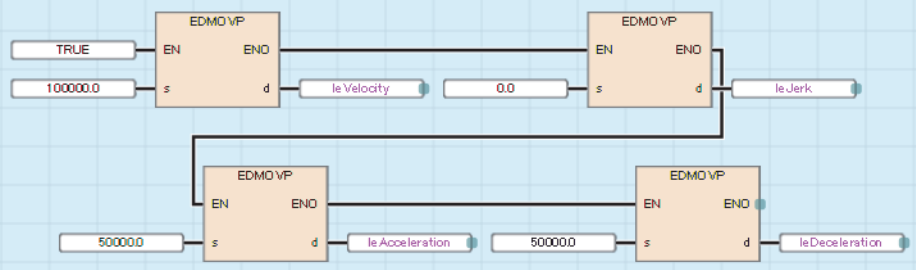
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

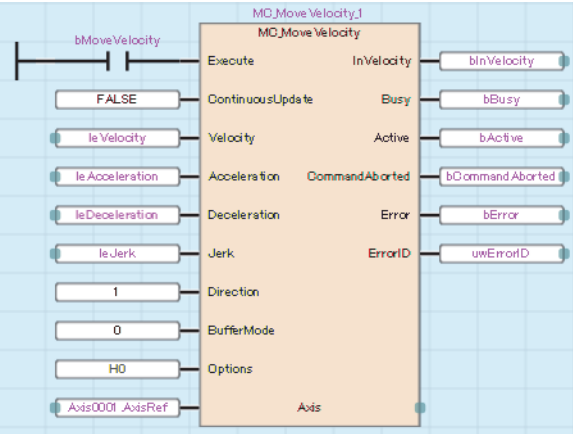
Label name	Data type	Comment
MC_MoveVelocity_1	MC_MoveVelocity	Speed Control FB
bMoveVelocity	Bit	Speed control command
leVelocity	Double-precision real number	Command velocity
leAcceleration	Double-precision real number	Acceleration
leDeceleration	Double-precision real number	Deceleration
leJerk	Double-precision real number	Jerk
bInVelocity	Bit	Target velocity reached
bBusy	Bit	Executing
bActive	Bit	Controlling
bCommandAborted	Bit	Abortion of execution
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data settings for speed control



• Speed Control



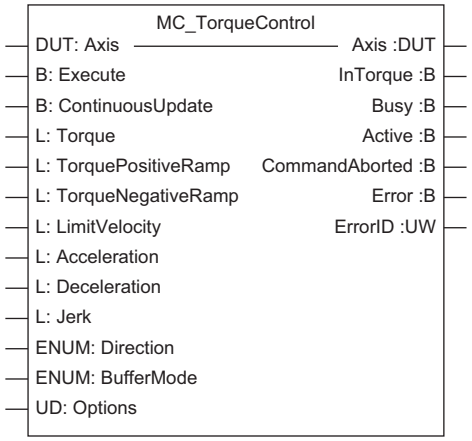
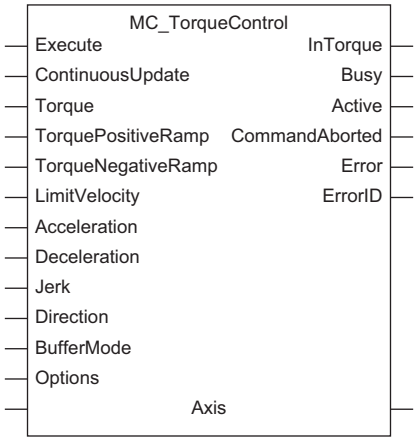
■ST program

```
//-----Data settings for speed control-----
leVelocity:= 100000.0;
leAcceleration:= 50000.0;
leDeceleration:= 50000.0;
leJerk:= 0.0;

//-----Speed Control-----
MC_MoveVelocity_1(
   Axis:= Axis0001.AxisRef ,
   Execute:= bMoveVelocity ,
   ContinuousUpdate:= FALSE ,
   Velocity:= leVelocity ,
   Acceleration:= leAcceleration ,
   Deceleration:= leDeceleration ,
   Jerk:= leJerk ,
   Direction:= MC_DIRECTION__mcPositiveDirection ,
   BufferMode:= MC_BUFFER_MODE__mcAborting ,
   Options:= H00000000 ,
   InVelocity=> bInVelocity ,
   Busy=> bBusy ,
   Active=> bActive ,
   CommandAborted=> bCommandAborted ,
   Error=> bError ,
   ErrorID=> uwErrorID
);
```

42.8 Torque Control

MC_TorqueControl

This FB switches the drive unit to cst and performs torque control according to the specified target torque.

Ladder	FBD/LD	ST
		MC_TorqueControl(Axis:= ?AXIS_REF? , Execute:= ?BOOL? , ContinuousUpdate:= ?BOOL? , Torque:= ?LREAL? , TorquePositiveRamp:= ?LREAL? , TorqueNegativeRamp:= ?LREAL? , LimitVelocity:= ?LREAL? , Acceleration:= ?LREAL? , Deceleration:= ?LREAL? , Jerk:= ?LREAL? , Direction:= ?INT? , BufferMode:= ?INT? , Options:= ?DWORD? , InTorque=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_TorqueControl (Torque Control) is executed.
ContinuousUpdate	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	Whether to enable or disable continuous change of Target torque (Torque), Torque positive direction ramp (TorquePositiveRamp), Torque negative direction ramp (TorqueNegativeRamp), Limit velocity (LimitVelocity), Acceleration (Acceleration), and Deceleration (Deceleration) is set. • FALSE: Disabled • TRUE: Enabled

Input variable	Name	Data type	Input	Valid range	Default value	Description
Torque	Target torque	LREAL	At start/Retrigger possible/ Continuous update possible	-1000.0 to 1000.0 [%]	0.0	The target torque is set. The ratio of the servo motor used to the rated torque is set in percentage. The number of significant digits after the decimal point differs depending on the ID assigned to the object data map. Values less than the number of significant digits are rounded down. To output the target torque in the positive direction (address increasing direction), set a positive value. To output the target torque in the negative direction (address decreasing direction), set a negative value.
TorquePositiveRamp	Torque positive direction ramp	LREAL	At start/Retrigger possible/ Continuous update possible	■When "0: Ramp method" is set 0.0 to 1000.0 [%/s] ■When "1: Time constant method" or "2: Fixed time method" is set 0.0 to 8400.0 [s]	0.0	The positive direction target torque is set. For details, refer to the following. ☞ Page 1664 Torque positive direction ramp (TorquePositiveRamp)/Torque negative direction ramp (TorqueNegativeRamp)
TorqueNegativeRamp	Torque negative direction ramp	LREAL	At start/Retrigger possible/ Continuous update possible		0.0	The negative direction target torque is set. ☞ Page 1664 Torque positive direction ramp (TorquePositiveRamp)/Torque negative direction ramp (TorqueNegativeRamp)
LimitVelocity	Limit velocity	LREAL	At start/Retrigger possible/ Continuous update possible	0.0001 to 2500000000.0	0.0	The velocity limit value for the cyclic torque mode is set. *: For the velocity limit value, refer to the drive unit manual. ■When the MR-J5(W)-G is used • Use the object data "Velocity limit value (2D20H)" as the velocity limit value. • When the object data "Velocity limit value (2D20H)" is not mapped, velocity commands from the controller will not be enabled. (The setting value of the servo parameter "Speed limit (PT67)" is enabled.) *: For details on object data, refer to the following. 📖 User's manual for the controller used
Acceleration	Acceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The acceleration and acceleration/deceleration time used until the limit velocity is reached is set. For details, refer to the following. ☞ Page 1665 Acceleration (Acceleration)
Deceleration	Deceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration used until the limit velocity is reached is set. For details, refer to the following. ☞ Page 1665 Deceleration (Deceleration)
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	The jerk used until the limit velocity is reached is set. For details, refer to the following. ☞ Page 1665 Jerk (Jerk) *: When "0.0" is set or this setting is omitted, the jerk is not applied.
Direction	Direction selection	INT (MC_DIRECTION)	At start	0	0	Set "0". *: If a value other than "0" is set, Out of Direction Selection Range (error code: 1AA5H) occurs.
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0, 1	0	The buffer mode is set. • 0: Aborting(mcAborting) • 1: Buffered(mcBuffered) For details, refer to the following. ☞ Page 1665 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	00000000H to 00060001H	00000000H	The function options for MC_TorqueControl (Torque Control) are set in the specified bit position. For details, refer to the following. ☞ Page 1666 Options (Options)

■Output variables

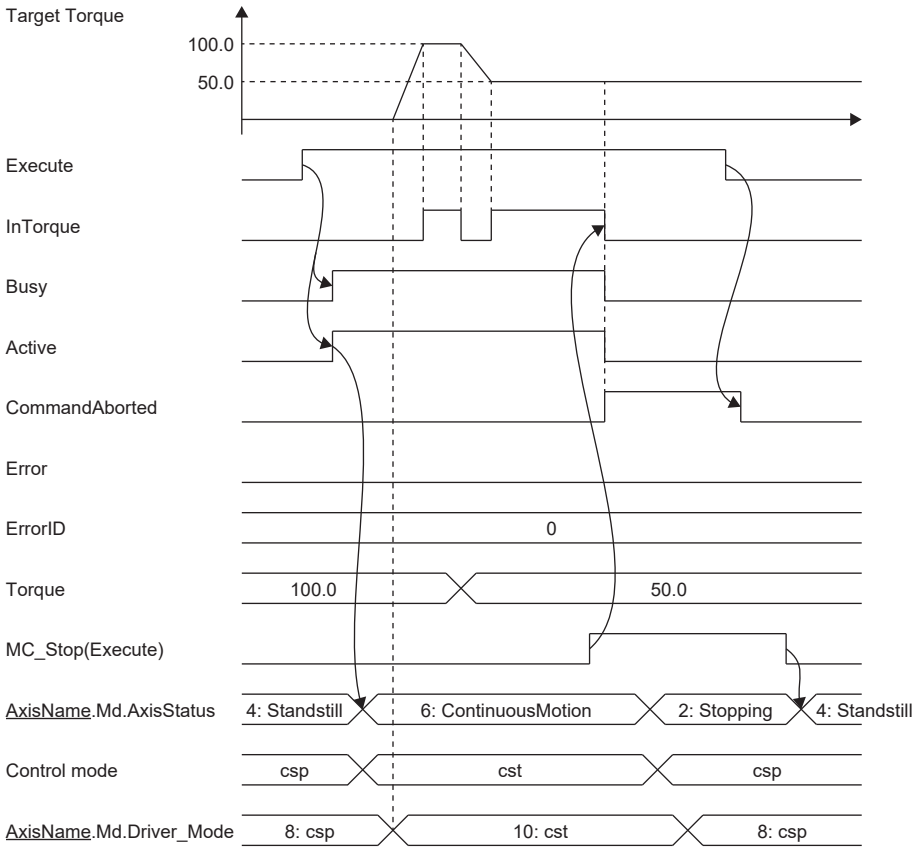
Output variable	Name	Data type	Default value	Description
InTorque	Target torque reached	BOOL	FALSE	This variable becomes TRUE when the target torque is reached. When the target torque has been changed due to change at Continuous update (ContinuousUpdate) TRUE, this variable remains FALSE until the set torque reaches the post-change target torque.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_TorqueControl (Torque Control) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MC_TorqueControl (Torque Control) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MC_TorqueControl (Torque Control) is aborted. The variable becomes FALSE when Execution command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

This FB switches the drive unit control mode to cst (cyclic synchronous torque mode) and performs control. This function controls the set torque based on the specified Torque positive direction ramp (TorquePositiveRamp) and Torque negative direction ramp (TorqueNegativeRamp). To finish MC_TorqueControl (Torque Control), start MC_Stop (Forced Stop).

■Timing chart

• When the processing is completed successfully



• When the processing is completed with an error

For the timing chart, refer to the following.

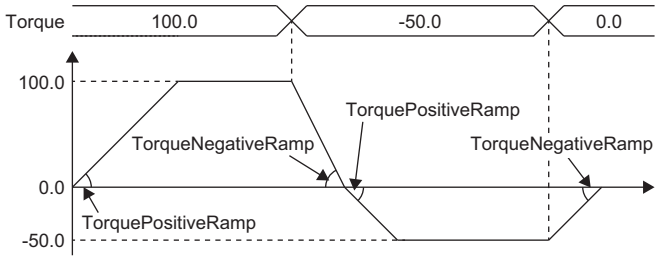
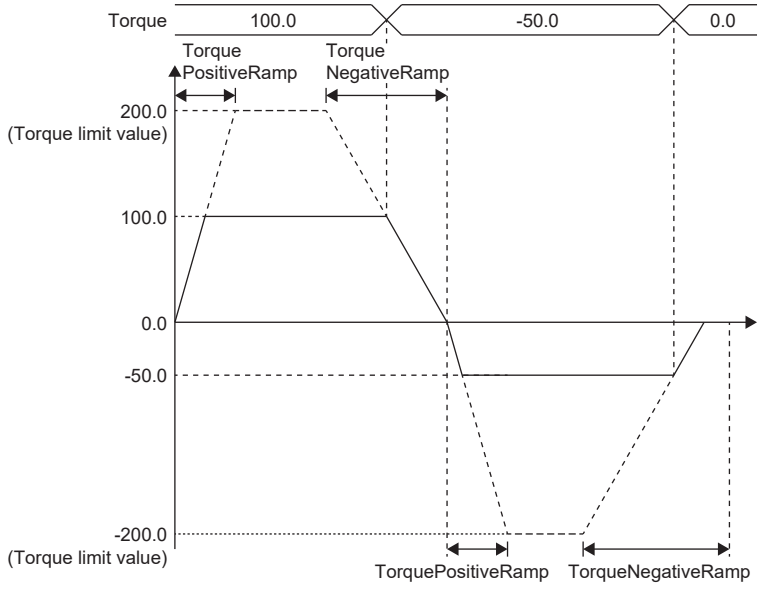
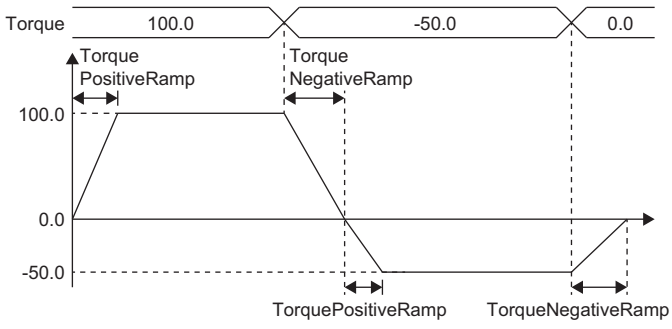
☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Torque positive direction ramp (TorquePositiveRamp)/Torque negative direction ramp (TorqueNegativeRamp)

The specified set torque is set.

Set the setting value used until the target torque is reached from the current set torque.

The setting value used until the target torque is reached differs depending on the setting of Torque ramp function selection (Options (Options): Bit 16, 17).

Torque ramp function selection (Options (Options): Bit 16, 17)	Description	Setting range
0: Ramp method	Specify the ramp from the current set torque until the target torque is reached. 	0.0 to 1000.0 [%/s]
1: Time constant method	Set the time until the set torque reaches the positive/negative torque limit value from "0". 	0.0 to 8400.0 [s]
2: Fixed time method	Set the time from the current set torque until the target torque is reached. 	0.0 to 8400.0 [s]

When the output torque direction is changed due to the target torque change, the set torque becomes "0.0" based on the setting value of Torque negative direction ramp (TorqueNegativeRamp). After that, the set torque becomes the target torque based on the setting value of Torque positive direction ramp (TorquePositiveRamp).

When Torque positive direction ramp (TorquePositiveRamp) and Torque negative direction ramp (TorqueNegativeRamp) are set to "0.0", the value reaches the target torque in one operation cycle.

■Acceleration (Acceleration)

The acceleration used in MC_TorqueControl (Torque Control) is set.

The setting range differs depending on the setting for Acceleration/deceleration method setting (Options (Options): Bit 0 to 2).

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	0.000000[*1], positive numbers from 0.000001 to 8400.0 [s]

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Deceleration (Deceleration)

The deceleration used in MC_TorqueControl (Torque Control) is set.

Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Deceleration (Deceleration) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Jerk (Jerk)

The jerk used in MC_TorqueControl (Torque Control) is set.

Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Jerk (Jerk) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000, positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following buffer modes are settable in MC_TorqueControl (TorqueControl).

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately. Switches to torque control immediately.
1: Buffered (mcBuffered)	Buffers the next FB on the under-control FB. If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after completion of the under-control FB. Switches to torque control after the processing of the previous FB is completed. When the previous FB is MC_TorqueControl (Torque Control), switches to torque control when Target torque reached (InTorque) is TRUE.

Point

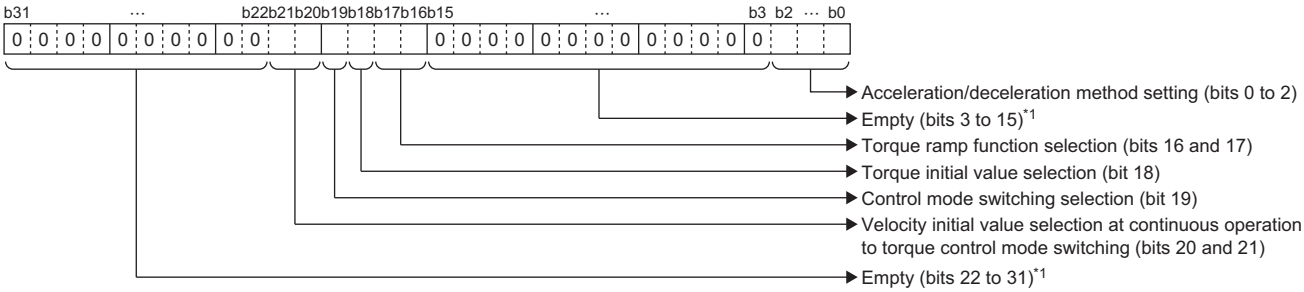
For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

■Options (Options)

The function options for MC_TorqueControl (Torque Control) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
0 to 2	Acceleration/deceleration method setting	The acceleration/deceleration method used to perform control is set. • 0: Acceleration/deceleration specification method (mcAccDec) • 1: Acceleration/deceleration time-fixed method (mcFixedTime)
16, 17	Torque ramp function selection	The method used until reaching the target torque from the set torque is set. • 0: Ramp method • 1: Time constant method • 2: Fixed time method *: For details, refer to the following. ☞ Page 1664 Torque positive direction ramp (TorquePositiveRamp)/Torque negative direction ramp (TorqueNegativeRamp)
18	Torque initial value selection	The torque initial value used when switching to cyclic synchronous torque mode (cst) is set. • 0: Target torque • 1: Actual torque
19	Control mode switching selection*2	The switching control mode is set. • 0: cst (cyclic synchronous torque mode) • 1: ct (continuous operation to torque control mode)
20, 21	Velocity initial value selection at continuous operation to torque control mode switching*3	The velocity initial value used when changing the control mode from cyclic synchronous position mode (csp) to continuous operation to torque control mode (ct) is set. • 0: Command velocity • 1: Actual velocity • 2: Automatic selection

*2 Switching to continuous operation to torque mode is possible when object data "Supported drive modes (6502Hh) (Bit 20)" is TRUE. When this bit is set to "1: ct (Continuous operation to torque control mode)" while "Supported drive modes (6502Hh) (Bit 20)" is FALSE, Driver Control Mode Unsupported (error code: 1B1BH) occurs.
For details on continuous operation to torque control mode, refer to the following.
📖 User's manual for the controller used

*3 Enabled only when Control mode switching selection (Options (Options): Bit 19) is set to "1: ct (Continuous operation to torque control mode)".

• Acceleration/deceleration method setting (bit 0 to 2)

Setting value	Description
0: Acceleration/deceleration specification method (mcAccDec)	An acceleration/deceleration method using Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk) set in MC_MoveAbsolute (Absolute Value Positioning).
1: Acceleration/deceleration time-fixed method (mcFixedTime)	An acceleration/deceleration method using the acceleration/deceleration time specified in MC_MoveAbsolute (Absolute Value Positioning) regardless of the velocity. The acceleration/deceleration time does not use Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk).

• Torque initial value selection (Bit 18)

Setting value	Description
0: Target torque	Target torque (Torque) value at start is set as the set torque regardless of the value of Torque positive direction ramp (TorquePositiveRamp) and Torque negative direction ramp (TorqueNegativeRamp) immediately after the control mode switching.
1: Actual torque	The Torque actual value at switching is set as the set torque.*1

*1 When the object assigned to the object data Torque actual value is not mapped to a PDO, the actual torque received from the drive unit at switching is not used. The torque initial value is 0.

Point

Normally, set this bit to "0: Target torque". Set "1: Actual torque" only when switching the control mode immediately after the command to the motor is completed without waiting for the servo motor to stop.

• Velocity initial value selection at continuous operation to torque control mode switching (Bit 20, 21)

Setting value	Description
0: Command velocity	The velocity during command is used as the velocity commanded to the drive unit immediately after switching.
1: Actual velocity	The motor speed received from the drive unit at switching is used.*1
2: Automatic selection	The lower velocity between "0: Command velocity" and "1: Actual velocity" is used as the velocity commanded to the drive unit immediately after switching.*2

*1 When object data "Velocity actual value (606CH)" is not mapped, the motor speed received from the drive unit at switching is not used. (Speed initial value selection becomes "0".)

*2 When object data "Velocity actual value (606CH)" is not mapped, the velocity commanded to the drive unit immediately after switching becomes "0: Command velocity".

■Related object data

The following functions result in errors or specification restrictions if object data is not set, PDO mapping is not performed, or the device station has no object.

For details on object data settings, refer to the following.

📖 User's manual for the controller used

◎: Required, ○: Recommended

Instruction/function	Required object data	PDO mapping	Operation when object is not set
MC_TorqueControl (Torque Control) (Control mode switching*1: cst)	Target torque	◎	Necessary Object Data Unset (error code: 1AF7H) occurs.
	Velocity limit value	○	Velocity limitation by LimitVelocity cannot be applied during cst.
	Torque actual value	○	The operation set in Torque initial value selection (Options (Options) bit: 18) is performed. ☞ Page 1666 Options (Options)
MC_TorqueControl (Torque Control) (Control mode switching*1: ct)	Target torque	◎	Necessary Object Data Unset (error code: 1AF7H) occurs.
	Velocity limit value	◎	Necessary Object Data Unset (error code: 1AF7H) occurs.
	Torque actual value	○	The operation set in Torque initial value selection (Options (Options) bit: 18) is performed. ☞ Page 1666 Options (Options)
	Velocity actual value	○	The operation set in Velocity initial value selection (Options (Options) bits: 16, 17) is performed. ☞ Page 1657 Options (Options)

*1 This can be set in Control mode switching selection (Options bit 19).

Precautions

- Velocity override factor (AxisName.Cd.VelocityOverride) and Acceleration override factor (AxisName.Cd.AccelerationOverride) are valid.
- The set position and the feed machine position are updated by follow up.
- The switching time of the control mode depends on the specifications of the drive unit.
- When the stop cause occurs during the control mode switching, the axis will stop immediately.
- Do not start a positioning control FB while switching the control mode. Check that Driver control mode (AxisName.Md.Driver_Mode) has switched to "10: Torque control (cst)" before starting the positioning control FB.
- When switching the control mode from cyclic synchronous position mode (csp) to cyclic synchronous velocity mode (csv), if a stop cause (axis error detection) occurs before the control mode is switched, follow-up may be performed in cyclic synchronous position mode (csp) until zero speed status is reached.
- Note the following precautions when using MR-J5(W)-G and switching from csp to cst without waiting for the motor to stop, or when switching from cst to csp.
  - Set servo parameter (extension setting) "ZSP disabled selection at control switching (PC76.1)" to "1: Disabled (control switching is performed regardless of the range of ZSP)" and disable monitoring of zero speed status. Note that doing so may cause vibration or impact to occur at control mode switching.
  - For the setting values of servo parameters (basic setting) "Electronic gear numerator (PA06)" and "Electronic gear denominator (PA07)", refer to the following.

  User's manual for the controller used
- When Velocity limit value is not mapped, the set torque is changed to 0 when a stop cause occurs.

■Torque limit change during cyclic synchronous torque mode (cst)

- At restart or continuous update, if the target torque is changed to a value greater than the torque limit value (Positive direction torque limit value (AxisName.Md.TorqueLimit_Positive), Negative direction torque limit value (AxisName.Md.TorqueLimit_Negative)), Torque Limit Value Over Warning (event code: 00D12H) occurs and the axis operates at the value before change.
- During torque control, if Positive direction torque limit value (AxisName.Cd.TorqueLimit_Positive), Negative direction torque limit value (AxisName.Cd.TorqueLimit_Negative) are changed to a value smaller than the target torque, the target torque will be changed to the torque limit value. In this case, the set torque is changed in one cycle.

Program example

The following is a program example. The torque control command (bTorqueControl) is set to TRUE, the mode is switched to torque control mode, and torque control is performed for Axis 1 (Axis0001).

• Settings

Item	Setting value
Velocity	100000.0
Acceleration	50000.0
Deceleration	50000.0
Jerk	0.0

■Axis

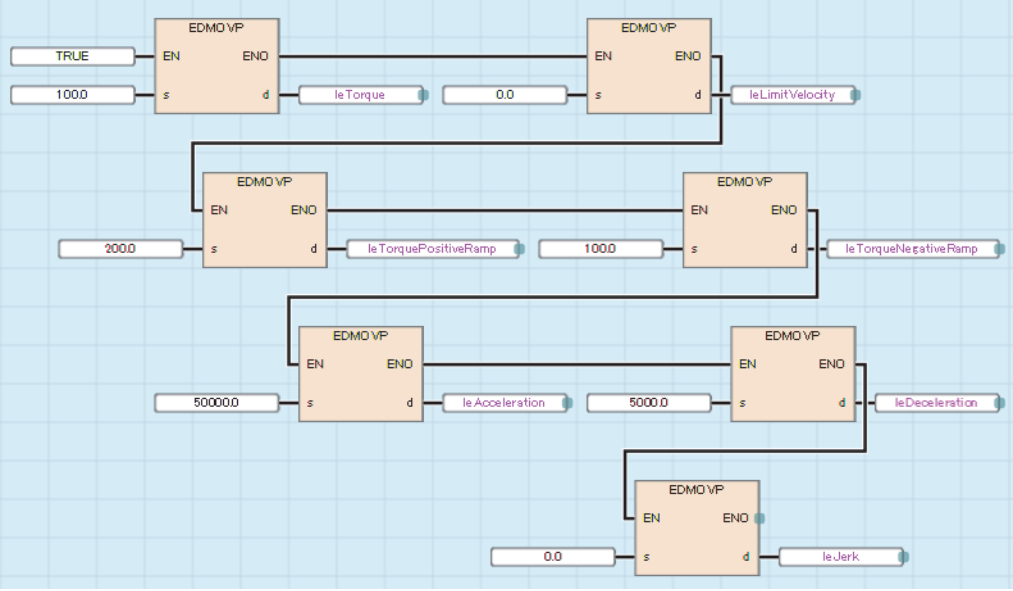
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

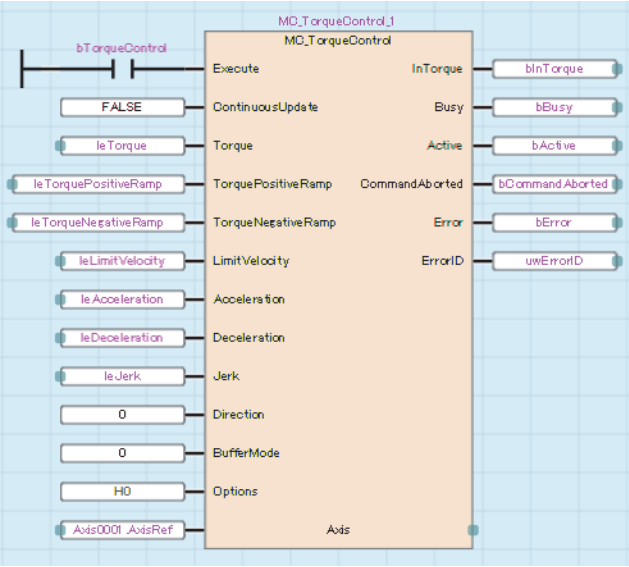
Label name	Data type	Comment
MC_TorqueControl_1	MC_TorqueControl	Torque Control FB
bTorqueControl	Bit	Torque control command
leTorque	Double-precision real number	Target torque
leTorquePositiveRamp	Double-precision real number	Torque positive direction ramp
leTorqueNegativeRamp	Double-precision real number	Torque negative direction ramp
leLimitVelocity	Double-precision real number	Limit velocity
leAcceleration	Double-precision real number	Acceleration
leDeceleration	Double-precision real number	Deceleration
leJerk	Double-precision real number	Jerk
bInTorque	Bit	Target torque reached
bBusy	Bit	Executing
bActive	Bit	Controlling
bCommandAborted	Bit	Abortion of execution
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data setting for Torque Control



• Torque Control



■ST program

```
//-----Data setting for Torque Control-----
leTorque:= 100.0;
leTorquePositiveRamp:= 200.0;
leTorqueNegativeRamp:= 100.0;
leLimitVelocity:= 100000.0;
leAcceleration:= 50000.0;
leDeceleration:= 50000.0;
leJerk:= 0.0;

//-----Torque Control-----
MC_TorqueControl_1(
   Axis:= Axis0001.AxisRef ,
   Execute:= bTorqueControl ,
   ContinuousUpdate:= FALSE ,
   Torque:= leTorque ,
   TorquePositiveRamp:= leTorquePositiveRamp ,
   TorqueNegativeRamp:= leTorqueNegativeRamp ,
   LimitVelocity:= leLimitVelocity ,
   Acceleration:= leAcceleration ,
   Deceleration:= leDeceleration ,
   Jerk:= leJerk ,
   Direction:= 0 ,
   BufferMode:= MC_BUFFER_MODE__mcAborting ,
   Options:= H00000000 ,
   InTorque=> bInTorque ,
   Busy=> bBusy ,
   Active=> bActive ,
   CommandAborted=> bCommandAborted ,
   Error=> bError ,
   ErrorID=> uwErrorID
);
```

42.9 Pressure Control

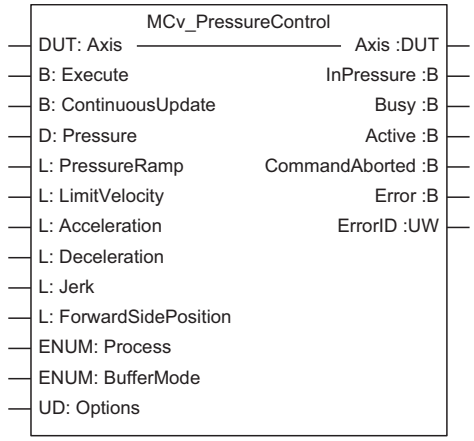
MCv_PressureControl

This FB switches the control mode of the drive unit to cyclic synchronous pressure mode (cspr) and performs pressure control according to the specified pressure.

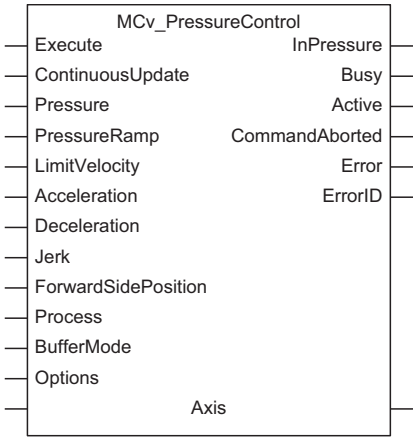
Restriction

When using these instructions, check the versions of the controller and engineering tool used. (MELSEC MX Controller MX-R Model User's Manual)

Ladder



FBD/LD



ST

```
MCv_PressureControl(
  Axis:= ?AXIS_REF? ,
  Execute:= ?BOOL? ,
  ContinuousUpdate:= ?BOOL? ,
  Pressure:= ?DINT? ,
  PressureRamp:= ?LREAL? ,
  LimitVelocity:= ?LREAL? ,
  Acceleration:= ?LREAL? ,
  Deceleration:= ?LREAL? ,
  Jerk:= ?LREAL? ,
  ForwardSidePosition:= ?LREAL? ,
  Process:= ?INT? ,
  BufferMode:= ?INT? ,
  Options:= ?DWORD? ,
  InPressure=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_PressureControl (Pressure Control) is executed.
ContinuousUpdate	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	Whether to enable or disable continuous change of Pressure (Pressure), Pressure ramp (PressureRamp), Limit velocity (LimitVelocity), Acceleration (Acceleration), Deceleration (Deceleration), Forward-side position (ForwardSidePosition), and Process (Process) of pressure control is set. • FALSE: Disabled • TRUE: Enabled
Pressure	Pressure	DINT	At start/Retrigger possible/ Continuous update possible	0 to 30000 [press units]	0	This variable sets the specified pressure. The units [press units] are the same as the units of the object data Target pressure (2E50H). They differ general pressure unit systems. For details on [press units], refer to the manual of the drive unit to be used. If the value is out of range during execution of the FB, an FB input variable error (error code: 1B27H) is output. If the value is changed to outside the range by a restart/ continuous update, the changed value is ignored, and the operation continues with the value before the change. When changed, Out of Pressure Range Warning (event code: 00D4FH) is output. *: The valid range is restricted depending on the settings. For details, refer to "Pressure ramp function" in the following manual. 📖User's manual for the controller used
PressureRamp	Pressure ramp	LREAL	At start/Retrigger possible/ Continuous update possible	0.0, 0.0001 to 2147483647.0 [press units/s]	0.0	This variable sets the ramp from the command pressure to reach the pressure (Pressure). • 0.0: Ramp disabled (The specified pressure is reached in one control cycle.) • 0.0001 to 2147483647.0: Ramp enabled *: When a positive number less than 0.0001 is set, the value is imported as 0.0 [press units/s]. If the value is out of range during execution of the FB, an FB input variable error (error code: 1B27H) is output. If the value is changed to outside the range by a restart/ continuous update, the changed value is ignored, and the operation continues with the value before the change. When changed, Out of Pressure Ramp Range Warning (event code: 00D50H) is output. *: The valid range is restricted depending on the settings. For details, refer to "Pressure ramp function" in the following manual. 📖User's manual for the controller used
LimitVelocity	Limit velocity	LREAL	At start/Retrigger possible/ Continuous update possible	0.0, 0.0001 to 2500000000.0	0.0	The velocity limit value for the cyclic synchronous pressure mode is set. If the value is out of range during execution of the FB, an Out of limit velocity range error (error code: 1A86H) is output. If the value is changed to outside the range by a restart/ continuous update, the changed value is ignored, and the operation continues with the value before the change. When changed, Out of Limit Velocity Range Warning (event code: 00D1BH) is output.
Acceleration	Acceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The acceleration and acceleration/deceleration time used until the limit velocity is reached is set. For details, refer to the following. ☞ Page 1676 Acceleration (Acceleration)
Deceleration	Deceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration used until the limit velocity is reached is set. For details, refer to the following. ☞ Page 1676 Deceleration (Deceleration)

Input variable	Name	Data type	Input	Valid range	Default value	Description
Jerk	Jerk	LREAL	At start	0.0	0.0	Set 0.0. *: If a value other than 0.0 is set, Out of Jerk Range (error code: 1A8FH) occurs.
ForwardSidePosition	Forward-side position	LREAL	At start/Retrigger possible/ Continuous update possible	-10000000000.0 to 10000000000.0	0.0	The limit position or the last position is set. For details, refer to the following. ☞ Page 1676 Forward-side position (ForwardSidePosition)
Process	Pressure control process	INT (MC_PRESSURE_PROCESS)	At start/Retrigger possible/ Continuous update possible	0 to 2	0	The pressure control process to be sent to the drive unit is set. • 0: Feed (Feed) • 1: Dwell (Dwell) • 2: Pressure release (Release) For details, refer to the following. ☞ Page 1676 Pressure control process (Process)
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0, 1	0	The buffer mode is set. • 0: Aborting(mcAborting) • 1: Buffered(mcBuffered) For details, refer to the following. ☞ Page 1677 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	00000000H to 00090001H	00000000H	The function options for MCv_PressureControl (Pressure Control) are set in the specified bit position. For details, refer to the following. ☞ Page 1677 Options (Options)

■Output variables

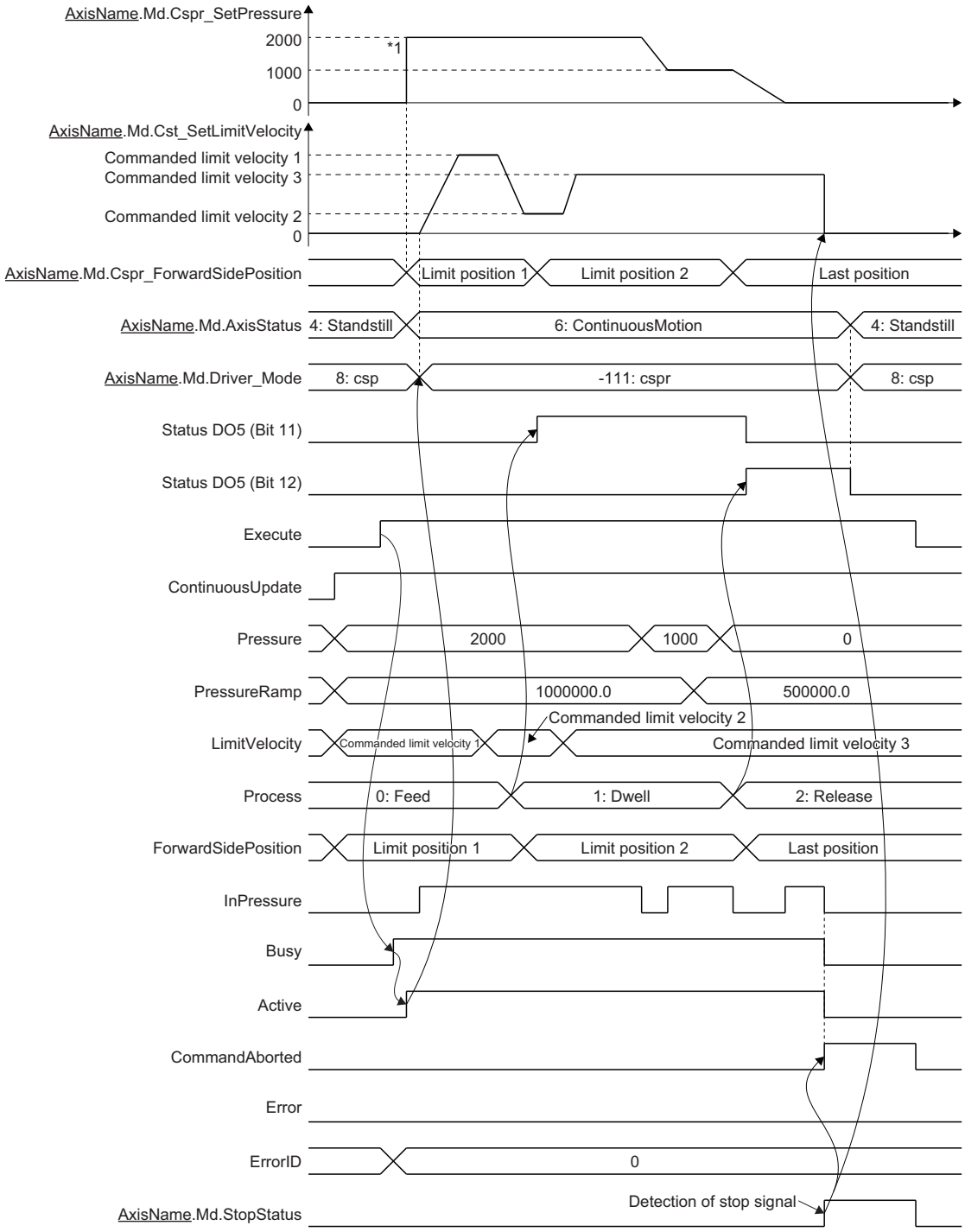
Output variable	Name	Data type	Default value	Description
InPressure	Target pressure reached	BOOL	FALSE	This variable becomes TRUE when the pressure reaches the command pressure. When the pressure has been changed due to change at Continuous update (ContinuousUpdate) TRUE, this variable remains FALSE until the pressure reaches the post-change pressure.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_PressureControl (Pressure Control) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MCv_PressureControl (Pressure Control) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MCv_PressureControl (Pressure Control) is aborted. The variable becomes FALSE when Execution command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

- This FB switches the drive unit control mode to cyclic synchronous pressure mode (cspr) and performs control. The specified pressure is controlled according to the set Pressure (Pressure) and Pressure ramp (PressureRamp). To stop MCv_PressureControl (Pressure Control), perform multiple start with another motion FB to either perform multiple start with MC_Stop (Forced Stop), use the axis stop signal, or switch to cyclic synchronous position mode (csp).

■Timing chart

- When the processing is completed successfully



*1 The command pressure at start depends on the setting of Specified pressure initial value selection (Options (Options): Bit 16). (This applies when "0: Specified pressure" is selected.)

- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Acceleration (Acceleration)

The acceleration/deceleration time for MCv_PressureControl (Pressure Control) is set.

The setting range differs depending on the setting for Acceleration/deceleration method setting (Options (Options): Bit 0 to 2).

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	0.000000[*1], positive numbers from 0.000001 to 8400.0 [s]

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Deceleration (Deceleration)

The deceleration time for MCv_PressureControl (Pressure Control) is set.

Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Deceleration (Deceleration) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Forward-side position (ForwardSidePosition)

The limit position or the last position is set.

Set Forward-side position (ForwardSidePosition) within the following range.

Setting range
-10000000000.0 ≤ Setting value < 10000000000.0[*1]

*1 When the ring counter is enabled: "Ring counter lower limit value ≤ Setting value < Ring counter upper limit value"

If the value is out of range during execution of the FB, an FB input variable error (error code: 1B27H) is output.

If the value is changed to outside the range by a restart/continuous update, the changed value is ignored, and the operation continues with the value before the change. When changed, Forward-Side Position Incorrect Warning (event code: 00D51H) is output.

The valid range is restricted depending on the settings. For details, refer to "limit position/final position (forward end position)" in the following manual.

User's manual for the controller used

■Pressure control process (Process)

The pressure control process to be sent to the drive unit is set.

Setting value	Description
0: Feed (Feed)	The process of moving the tip part of a machine for pushing and applying pressure into contact with a workpiece and gradually moving the tip to the dwell start position
1: Dwell (Dwell)	The process of applying pressure to the workpiece
2: Pressure release (Release)	The process of releasing pressure from the workpiece

If the value is out of range during execution of the FB, an FB input variable error (error code: 1B27H) is output.

If the value is changed to outside the range by a restart/continuous update, the changed value is ignored, and the operation continues with the value before the change. When changed, Process Incorrect Warning (event code: 00D52H) is output.

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following buffer modes are settable in MCv_PressureControl (Pressure Control).

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.
1: Buffered(mcBuffered)	Buffers the next FB on the under-control FB. If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after completion of the under-control FB.

Point

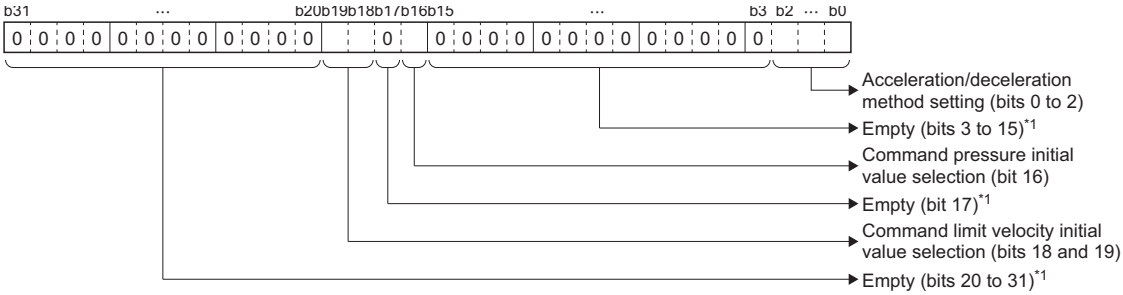
For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

■Options (Options)

The function options used in MCv_PressureControl (Pressure Control) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
0 to 2	Acceleration/deceleration method setting	The acceleration/deceleration method used to perform control is set. • 0: Acceleration/deceleration specification method (mcAccDec) • 1: Acceleration/deceleration time-fixed method (mcFixedTime)
16	Specified pressure initial value selection	The initial value of the specified pressure used when switching to cyclic synchronous pressure mode (cspr) is set. • 0: Specified pressure • 1: Actual pressure
18, 19	Command limit velocity initial value selection*2	The limit velocity initial value used when changing the control mode from cyclic synchronous position mode (csp) to cyclic synchronous pressure mode (cspr) is set. • 0: Command velocity • 1: Specified limit velocity • 2: Automatic selection

*2 If a value outside the range is specified, an Out of velocity initial value range error (error code: 1A91H) occurs when FB is executed.

• Acceleration/deceleration method setting (bit 0 to 2)

Setting value	Description
0: Acceleration/deceleration specification method (mcAccDec)	An acceleration/deceleration method using Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk) set in MCv_PressureControl (Pressure Control)
1: Acceleration/deceleration time-fixed method (mcFixedTime)	An acceleration/deceleration method using the acceleration/deceleration time set in MCv_PressureControl (Pressure Control) regardless of the velocity. The acceleration/deceleration time is set to Acceleration (Acceleration), and Deceleration (Deceleration), and Jerk (Jerk) are not used.

• Specified pressure initial value selection (Bit 16)

Setting value	Description
0: Specified pressure	The initial value of the command pressure is Pressure (Pressure).
1: Actual pressure	The initial value of the command pressure is the actual pressure (object data_PressureActualValue (AxisName.Md.Io_PressureActualValue)). If the actual pressure is a negative value, the initial value will be "0". If the actual pressure exceeds "30000", the initial value will be "30000".*1

*1 When the object assigned to the object data Pressure actual value (2E52H) is not mapped to a PDO, the actual pressure is not used. (The pressure initial value is 0.)

- Command limit velocity initial value selection (Bit 18, 19)

Setting value	Description
0: Command velocity	The initial value of the command limit velocity is the set velocity (AxisName.Md.SetVelocity). After switching, acceleration or deceleration is applied toward the limit velocity (LimitVelocity).
1: Specified limit velocity	The initial value of the command limit velocity is the limit velocity (LimitVelocity).
2: Automatic selection	The initial value of the command limit velocity is the lower of "0: Command velocity" and "1: Specified limit velocity". If "0: Command velocity" is lower, after switching, acceleration or deceleration is applied toward the limit velocity (LimitVelocity).

■Necessary object data

When using MCv_PressureControl (Pressure Control), map the objects assigned to the object data listed below to a PDO.
◎: PDO mapping required, ○: PDO mapping recommended

Required object data	Mapping availability	Operation when object is not set
Control DI 5 (2D050H)	◎	At start or multiple start, if the required object data is not set, Necessary Object Data Unset (error code: 1AF7H) occurs.
Velocity limit value (2D20H)		
Target pressure (2E50H)		
Forward-side stop function (2E56H)	○	The Forward-Side Position (ForwardSidePosition) setting is ignored, and limit position/last position is disabled.
Pressure actual value (2E52H)		The setting value does not become the actual pressure. "0" is used as the pressure initial value.

For details on object data settings, refer to "Axis assignment" in the following manual.
📖User's manual for the controller used

Precautions

- Follow-up is performed during cyclic synchronous pressure mode (cspr). For details on follow-up, refer to the following.
  📖User's manual for the controller used
- The switching time of the control mode depends on the specifications of the drive unit.
- Do not perform multiple start with the positioning control FB until the control mode has been switched from cyclic synchronous position mode (csp) to cyclic synchronous pressure mode (cspr). Check that Driver control mode (AxisName.Md.Driver_Mode) has switched to "-111: Cyclic synchronous pressure mode (cspr)" before performing multiple start with the positioning control FB.
- During cyclic synchronous pressure mode (cspr), the actual motor speed may not reach the limit velocity depending on the machine load conditions.
- When switching the control mode from cyclic synchronous position mode (csp) to cyclic synchronous pressure mode (cspr), or from cyclic synchronous pressure mode (cspr) to cyclic synchronous position mode (csp), stopping the motor (zero speed status) before switching is recommended. Zero speed status can be monitored by assigning a network label to Status DO2 (2D12H).
- When switching the control mode from cyclic synchronous position mode (csp) to cyclic synchronous pressure mode (cspr), if a stop cause (axis error detection) occurs before the control mode is switched, follow-up may be performed in cyclic synchronous position mode (csp) until zero speed status is reached.
- When switching the control mode from cyclic synchronous position mode (csp) to cyclic synchronous pressure mode (cspr), or from cyclic synchronous pressure mode (cspr) to cyclic synchronous position mode (csp) without waiting for the motor to stop, set the servo parameter "ZSP disabled selection at control switching (PC76.1)" to "1: Disabled" and disable zero speed status monitoring. Note that vibration or impact may occur at control mode switching.
- During cyclic synchronous pressure mode (cspr), driver command discard detection is not performed regardless of the setting of Driver command discard detection setting (AxisName.Pr.StopOption_DriverTargetIgnored). When the servo parameter "Limit switch enabled status selection (PD41.2)" is set to "0: Limit switch always enabled" and limit detection is performed on the drive unit side, switch to cyclic synchronous position mode (csp) using MC_Stop (Forced Stop) or the axis stop signal when stopping due to limit detection.
- Wire the load cell amplifier output to the analog input of the drive unit. For details, refer to the manuals of the connected devices.
- For functions of the drive unit that cannot be used during cyclic synchronous pressure mode (cspr), refer to the manuals of the connected devices.

- During pressure control, abnormalities in devices such as the load cell detector or fluctuations in the load cell pressure may cause the axis to keep moving backward. Set the software stroke limit.
- To limit the command pressure, change the Positive pressure limit value (2E5BH) by using one of the following methods. For details on the Positive pressure limit value (2E5BH), refer to the manuals of the connected devices.
  - Change the Positive pressure limit value (2E5BH) by using MC_WriteParameter (Parameter Write). For details on MC_WriteParameter (Parameter Write), refer to the following.

☞ Page 1521 Parameter Write

  - Map the Positive pressure limit value (2E5BH) to an RPDO and assign it a network label. Change the value of that label. For details on network labels and PDO mapping settings, refer to the following.

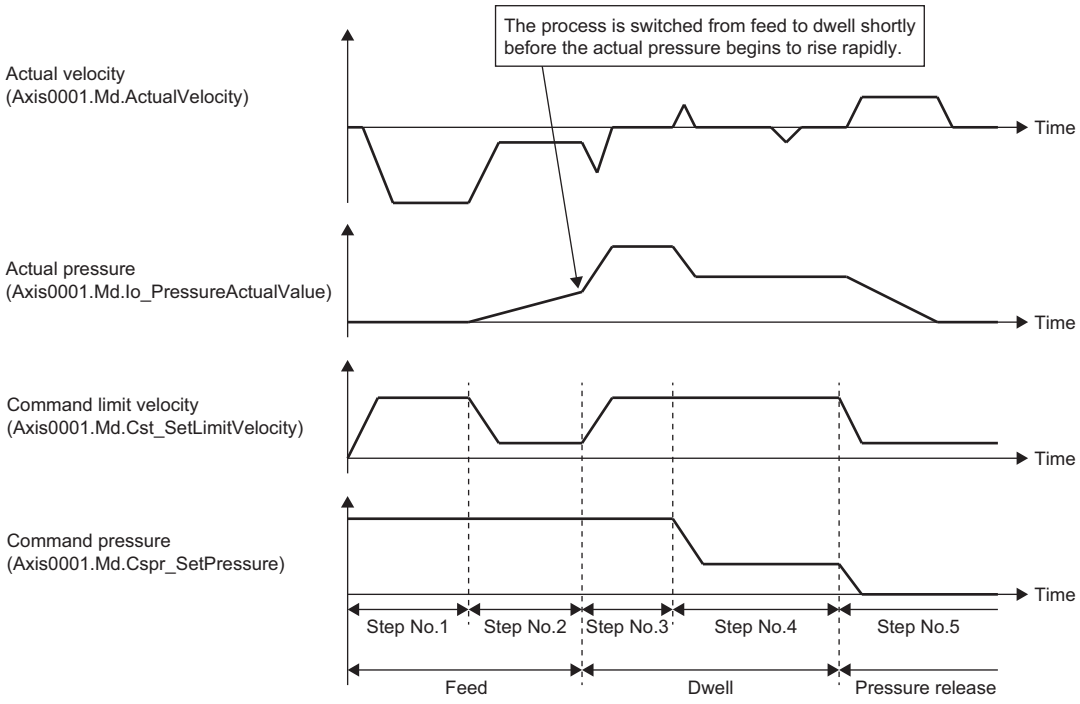
User's manual for the controller used

Program example

The following is a program example. Pressure control is performed by sequentially switching through steps No.1 to No.5 using the MR-J5-G-LL on axis 1.

- Operation

Step	Process	Description	Conditions for switching to next step
No.1	Feed	The operation to bring the tip part of the machine close to a workpiece	Actual position (Axis0001.Md.ActualPosition) ≤ -20000 [μm]
No.2		The operation to decelerate and touch the tip part of the machine against the workpiece at low speed, then move the tip part of the machine to the point just before the actual pressure rapidly rises after contact	Object data_PressureActualValue (Axis0001.Md.Io_PressureActualValue) > 1500 [press units]
No.3	Dwell	The operation to apply pressure to the workpiece until the conditions for switching are satisfied	50 [ms] passed from start of step
No.4		The operation to apply pressure to the workpiece until the conditions for switching are satisfied	100 [ms] passed from start of step
No.5	Pressure release	The operation to reduce the pressure applied to the workpiece until the actual pressure drops sufficiently	—



- Axis parameter settings (Axis parameters)

Item	Setting value
Driver unit conversion numerator (Axis0001.Pr.Drive_UnitConvRatioNum)	67108864
Driver unit conversion denominator (Axis0001.Pr.Drive_UnitConvRatioDen)	20000μm
Software stroke limit lower value (Axis0001.Pr.SwStrokeLimit_Lower)	-24000.0μm
Software stroke limit target (Axis0001.Pr.SwStrokeLimit_Target)	1: Set position
Software stroke limit upper value (Axis0001.Pr.SwStrokeLimit_Upper)	100000.0
Position command unit (Axis0001.Pr.Unit_Position)	μm

• Basic settings (Timer setting)

Item	Setting value
High-speed timer setting	100μs

• Servo parameters (MR-J5-G-LL)

No.	Name	Setting value
PA14	Moving direction selection	0: Rotation in the CCW direction when the positioning address increases
PW12.1	Pressure increase direction selection for positioning address	0: Pressure increase when the positioning address decreases
PW12.2	Forward-side/reverse-side stop function	1 (Forward-side stop: Enabled, Reverse-side stop: Disabled)
PW34.0	Pressure model adaptive control selection	0 (Dwell: Disabled, Pressure release: Disabled)

■Axis

Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

Label name	Data type	Comment
MCv_PressureControl_1	MCv_PressureControl	Pressure control FB
bPressureControl	Bit	Pressure control command
dPressure	Double word [signed]	Specified pressure
leLimitVelocity	Double-precision real number	Specified limit velocity
bInPressure	Bit	Target pressure reached
bBusy	Bit	Executing
bActive	Bit	Controlling
bCommandAborted	Bit	Abortion of execution
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code
wStepNo	Word [signed]	Step No.
bSwitchingConditionSatisfied	Bit	Conditions for switching to next step
fbTimerForDwellStepNo3	TIMER_HIGH_FB_M	High-speed timer for dwell step No.3
fbTimerForDwellStepNo4	TIMER_HIGH_FB_M	High-speed timer for dwell step No.4

■ST program

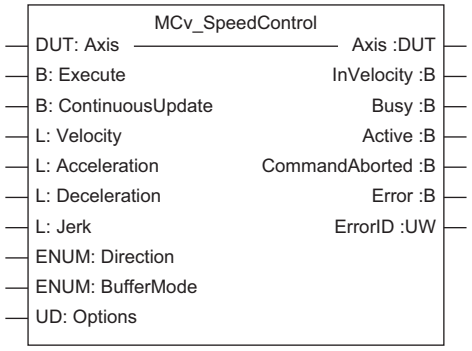
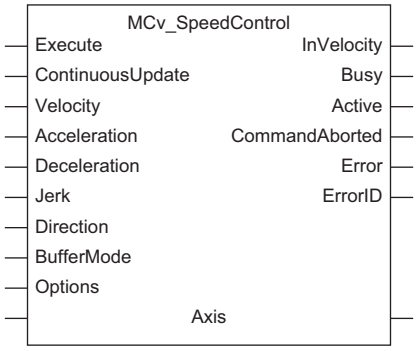
```
//-----Step-by-step data settings-----CASE wStepNo OF
  1:
    dPressure:= 2500;
    leLimitVelocity:= 50000.0;
  2:
    dPressure:= 2500;
    leLimitVelocity:= 10000.0;
  3:
    dPressure:= 2500;
    leLimitVelocity:= 50000.0;
  4:
    dPressure:= 2000;
    leLimitVelocity:= 50000.0;
  5:
    dPressure:= 0;
    leLimitVelocity:= 10000.0;
END_CASE;

//-----Call for pressure control FB-----
MCv_PressureControl_1(
  Axis:= Axis0001.AxisRef ,
```

```
    Execute:= bPressureControl ,
    ContinuousUpdate:= TRUE ,
    Pressure:= dPressure ,
    PressureRamp:= 10000.0 ,
    LimitVelocity:= leLimitVelocity ,
    Acceleration:= 3000000.0 ,
    Deceleration:= 4000000.0 ,
    Jerk:= 0.0 ,
    ForwardSidePosition:= -24000.0 ,
    Process:= MC_PRESSURE_PROCESS__Feed ,
    BufferMode:= MC_BUFFER_MODE__mcAborting ,
    Options:= H00000000 ,
    InPressure=> bInPressure ,
    Busy=> bBusy ,
    Active=> bActive ,
    CommandAborted=> bCommandAborted ,
    Error=> bError ,
    ErrorID=> uwErrorID
);

//------Switch to next step-----
CASE wStepNo OF
    1:
        bSwitchingConditionSatisfied:= (Axis0001.Md.ActualPosition <= -20000.0);
    2:
        bSwitchingConditionSatisfied:= (Axis0001.Md.Io_PressureActualValue > 1500);
    3:
        bSwitchingConditionSatisfied:= fbTimerForDwellStepNo3.Status;
    4:
        bSwitchingConditionSatisfied:= fbTimerForDwellStepNo4.Status;
    ELSE
        bSwitchingConditionSatisfied:= FALSE;
END_CASE;
INC(bSwitchingConditionSatisfied, wStepNo);

//------Call for high-speed timer FB for dwell-----
fbTimerForDwellStepNo3(
    Coil:= (wStepNo = 3) ,
    Preset:= 500
);
fbTimerForDwellStepNo4(
    Coil:= (wStepNo = 4),
    Preset:= 1000
);
```

42.10 Speed Control (Including Position Loop)

MCv_SpeedControl

This FB performs speed control including the position loop.

Ladder	FBD/LD	ST
		MCv_SpeedControl(Axis:= ?AXIS_REF? , Execute:= ?BOOL? , ContinuousUpdate:= ?BOOL? , Velocity:= ?LREAL? , Acceleration:= ?LREAL? , Deceleration:= ?LREAL? , Jerk:= ?LREAL? , Direction:= ?INT? , BufferMode:= ?INT? , Options:= ?DWORD? , InVelocity=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_SpeedControl (Speed Control (Including Position Loop)) is executed.
ContinuousUpdate	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	Whether to enable or disable continuous change of Velocity (Velocity), Acceleration (Acceleration), and Deceleration (Deceleration) is set. • FALSE: Disabled • TRUE: Enabled
Velocity	Velocity	LREAL	At start/Retrigger possible/ Continuous update possible	0.0, ±0.0001 to ±2500000000.0	0.0	The command velocity is set. When the velocity is negative, the axis moves in the reverse direction. When this setting is omitted, the axis does not operate, but Axis status (AxisName.Md.AxisStatus) changes to "6: During continuous operation (ContinuousMotion)". For details, refer to the following. ☞ Page 1684 Velocity (Velocity)
Acceleration	Acceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The acceleration is set. For details, refer to the following. ☞ Page 1684 Acceleration (Acceleration)
Deceleration	Deceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration is set. For details, refer to the following. ☞ Page 1685 Deceleration (Deceleration)

Input variable	Name	Data type	Input	Valid range	Default value	Description
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	The jerk is set. For details, refer to the following. ☞ Page 1685 Jerk (Jerk)
Direction	Direction selection	INT (MC_DIRECTION)	At start	1, 2	0	The direction is set. • 1: Positive direction (mcPositiveDirection) • 2: Negative direction (mcNegativeDirection) *: When "2: Negative direction (mcNegativeDirection)" is set and Velocity (Velocity) is negative, the movement direction of the motor will be positive. *: If this setting is omitted, Out of Direction Selection Range (error code: 1AA5H) occurs.
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0 to 5	0	The buffer mode is set. • 0: Aborting(mcAborting) • 1: Buffered(mcBuffered) • 2: BlendingLow(mcBlendingLow) • 3: BlendingPrevious(mcBlendingPrevious) • 4: BlendingNext(mcBlendingNext) • 5: BlendingHigh(mcBlendingHigh) For details, refer to the following. ☞ Page 1685 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	00000000H to 00000001H	00000000H	The function options for MCv_SpeedControl (Speed Control (Including Position Loop)) are set in the specified bit position. For details, refer to the following. ☞ Page 1686 Options (Options)

■Output variables

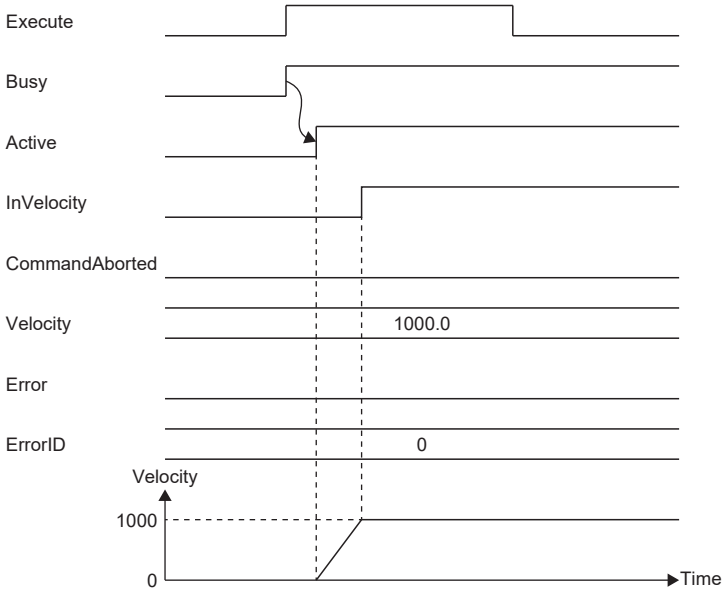
Output variable	Name	Data type	Default value	Description
InVelocity	Target velocity reached	BOOL	FALSE	This variable becomes TRUE when the command velocity reaches the target velocity. When the target velocity has been changed due to change at Continuous update (ContinuousUpdate) TRUE, this variable remains FALSE until the velocity reaches the post-change target velocity.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_SpeedControl (Speed Control (Including Position Loop)) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MCv_SpeedControl (Speed Control (Including Position Loop)) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MCv_SpeedControl (Speed Control (Including Position Loop)) is aborted. The variable becomes FALSE when Execution command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

This FB sets the control mode on the drive unit side to csp (cyclic synchronous position mode) and executes the speed control for the specified axis using the specified velocity. To stop the axis, use MC_Stop (Forced Stop) or start another motion FB. To perform the speed control with the control mode of the drive unit set to csv (cyclic synchronous velocity mode), use MC_MoveVelocity (Speed Control).

■Timing chart

• When the processing is completed successfully



• When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Velocity (Velocity)

The command velocity used in MCv_SpeedControl (Speed Control (Including Position Loop)) is set.

Setting range
0.0, ±0.0001 to ±2500000000.0[*1*2]

*1 To perform floating-point operation, the lower limit value of the command velocity is limited by the following restrictions. If the velocity converted from the command velocity to operation cycle is less than 0.00001, Out of Operation Cycle Converted Velocity Range (error code: 1B16H) (when changing the velocity, Out of Operation Cycle Converted Velocity Range Warning (event code: 00D2FH)) occurs. To improve the precision of floating-point operation, change Position command unit (AxisName.Pr.Unit_Position) and Speed command unit (AxisName.Pr.Unit_Velocity) and perform settings so that the velocity converted into the operation cycle is not less than "0.00001".

*2 When a motion control FB with a commanded velocity of "0.0" performs a multiple start, the commanded velocity of the previous motion control FB is used.

■Acceleration (Acceleration)

The acceleration used in MCv_SpeedControl (Speed Control (Including Position Loop)) is set.

The setting range differs depending on the setting for Acceleration/deceleration method setting (Options (Options): Bit 0 to 2).

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	0.000000[*1], positive numbers from 0.000001 to 8400.0 [s]

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Deceleration (Deceleration)

The deceleration used in MCv_SpeedControl (Speed Control (Including Position Loop)) is set.

Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Deceleration (Deceleration) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000*1, positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Jerk (Jerk)

The jerk used in MCv_SpeedControl (Speed Control (Including Position Loop)) is set.

Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Jerk (Jerk) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000, positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following buffer modes are settable in MCv_SpeedControl (Speed Control (Including Position Loop)).

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.
1: Buffered(mcBuffered)	Buffers the next FB on the under-control FB. If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after completion of the under-control FB.
2: BlendingLow(mcBlendingLow)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the lower target velocity between the under-control FB and the buffering FB is used as the switching speed.
3: BlendingPrevious(mcBlendingPrevious)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the under-control FB is used as the switching speed.
4: BlendingNext(mcBlendingNext)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the buffering FB is used as the switching speed.
5: BlendingHigh(mcBlendingHigh)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the higher target velocity between the under-control FB and the buffering FB is used as the switching speed.

*1 Stops are not performed between the under-control FB and the buffering FB.

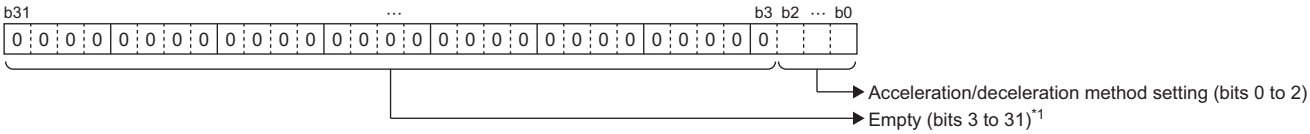
Point

For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

■Options (Options)

The function options for MCv_SpeedControl (Speed Control (Including Position Loop)) are set in the specified bit position.
The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

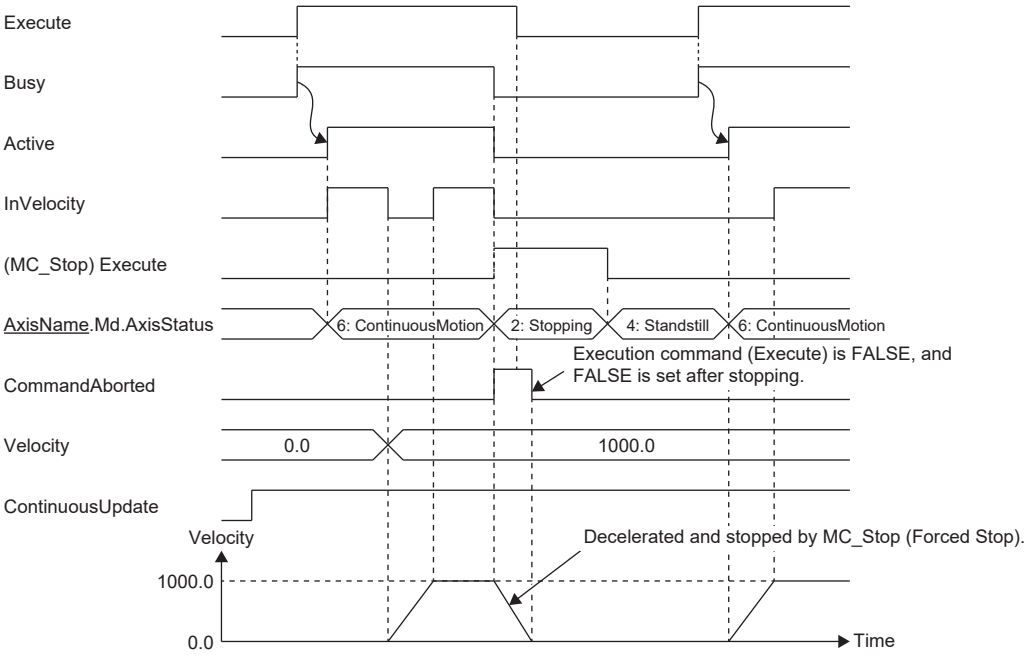
Bit	Name	Description
0 to 2	Acceleration/deceleration method setting	The acceleration/deceleration method used to perform control is set. • 0: Acceleration/deceleration specification method (mcAccDec) • 1: Acceleration/deceleration time-fixed method (mcFixedTime)

• Acceleration/deceleration method setting (bit 0 to 2)

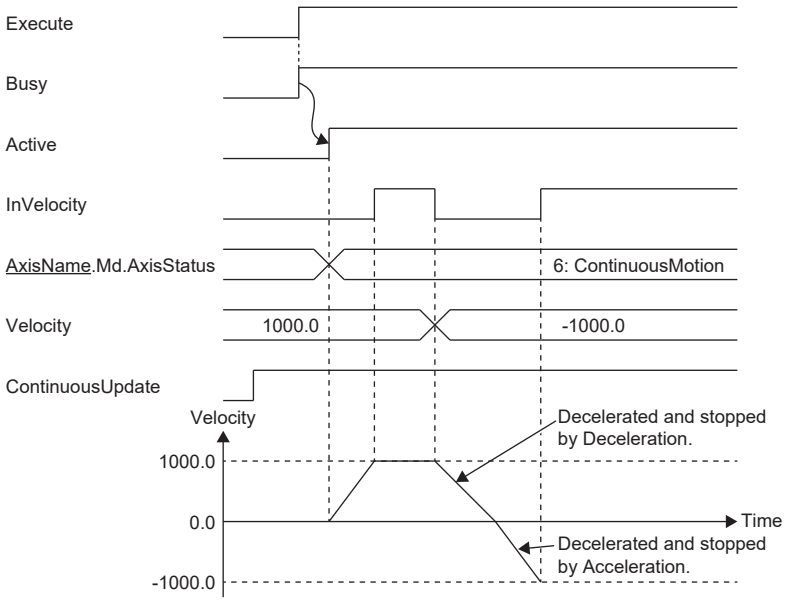
Setting value	Description
0: Acceleration/deceleration specification method (mcAccDec)	An acceleration/deceleration method using Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk) set in MCv_SpeedControl (Speed Control (Including Position Loop)).
1: Acceleration/deceleration time-fixed method (mcFixedTime)	An acceleration/deceleration method using the acceleration/deceleration time specified in MCv_SpeedControl (Speed Control (Including Position Loop)) regardless of the velocity. The acceleration/deceleration time does not use Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk).

■Operation overview

• The operation at start and when a stop cause occurs is shown below.



• If the sign of Velocity (Velocity) is reversed and the operation direction is changed when Continuous update (ContinuousUpdate) is TRUE, the axis accelerates toward the target velocity once after the deceleration stop.



■Necessary object data

When using MCv_SpeedControl (Speed Control (Including Position Loop)) for a single axis, set the following object data for the axis.

• Target position (607AH)

If the object data is not set, Necessary Object Data Unset (error code: 1AF7H) occurs and the axis will not start.

For details on object data settings, refer to the following.

📖 User's manual for the controller used

Program example

The following is a program example. The speed control command (bSpeedControl) is set to TRUE, the mode is switched to speed control including position loop, and speed control including position loop is performed for Axis 1 (Axis0001).

• Setting

Item	Setting value
Velocity	100000.0
Acceleration	50000.0
Deceleration	50000.0
Jerk	0.0

■Axis

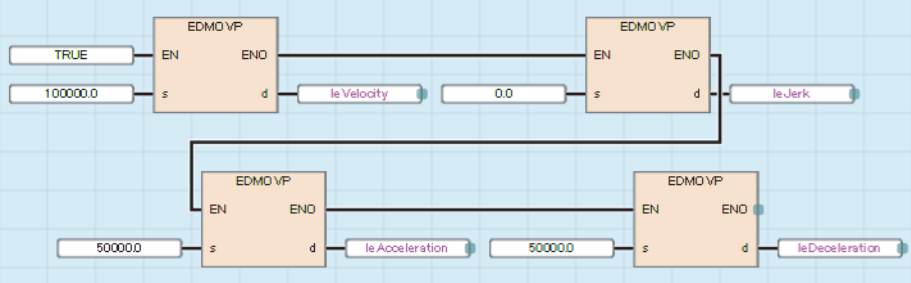
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1

■Labels used

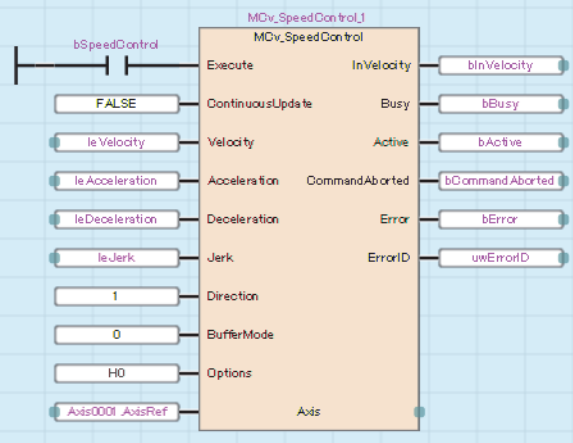
Label name	Data type	Comment
MCv_SpeedControl_1	MCv_SpeedControl	Speed Control FB
bSpeedControl	Bit	Speed control command
leVelocity	Double-precision real number	Command velocity
leAcceleration	Double-precision real number	Acceleration
leDeceleration	Double-precision real number	Deceleration
leJerk	Double-precision real number	Jerk
bInVelocity	Bit	Target velocity reached
bBusy	Bit	Executing
bActive	Bit	Controlling
bCommandAborted	Bit	Abortion of execution
bError	Bit	Error
uwErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data settings for speed control



• Speed Control (Including Position Loop)



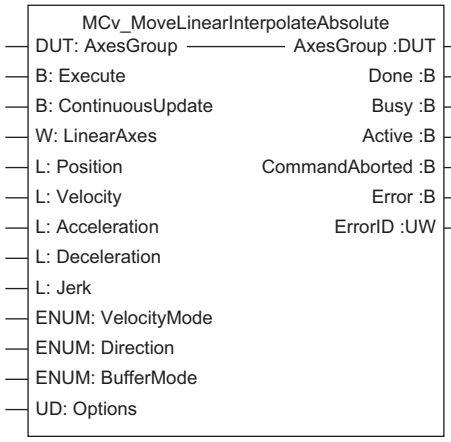
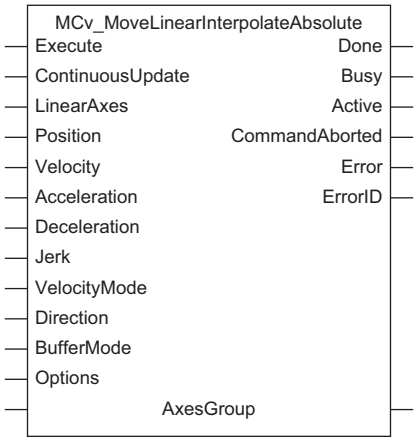
■ST program

```
//-----Data settings for speed control-----
leVelocity:= 100000.0;
leAcceleration:= 50000.0;
leDeceleration:= 50000.0;
leJerk:= 0.0;

//-----Speed Control (Including Position Loop)-----
MCv_SpeedControl_1(
    Axis:= Axis0001.AxisRef ,
    Execute:= bSpeedControl ,
    ContinuousUpdate:= FALSE ,
    Velocity:= leVelocity ,
    Acceleration:= leAcceleration ,
    Deceleration:= leDeceleration ,
    Jerk:= leJerk ,
    Direction:= MC_DIRECTION__mcPositiveDirection ,
    BufferMode:= MC_BUFFER_MODE__mcAborting ,
    Options:= H00000000 ,
    InVelocity=> bInVelocity ,
    Busy=> bBusy ,
    Active=> bActive ,
    CommandAborted=> bCommandAborted ,
    Error=> bError ,
    ErrorID=> uwErrorID
);
```

42.11 Absolute Value Linear Interpolation Control

MCv_MoveLinearInterpolateAbsolute

This FB specifies the target position of the absolute position of the set axes group and performs positioning by linear interpolation control.

Ladder	FBD/LD	ST
		MCv_MoveLinearInterpolateAbsolute(AxesGroup:= ?AXES_GROUP_REF? , Execute:= ?BOOL? , ContinuousUpdate:= ?BOOL? , LinearAxes:= ?INT(0..15)? , Position:= ?LREAL(0..15)? , Velocity:= ?LREAL? , Acceleration:= ?LREAL? , Deceleration:= ?LREAL? , Jerk:= ?LREAL? , VelocityMode:= ?INT? , Direction:= ?INT(0..15)? , BufferMode:= ?INT? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
AxesGroup	Axes group information	AXES_GROUP_REF	At start	—	Omission not possible	An axes group is set. For the variable (AxesGroupName.AxesGroupRef.) used, refer to the following. ☞ Page 1423 AxesGroupName.AxesGroupRef. (Axes group information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Start	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) is executed.
ContinuousUpdate	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	Whether to enable or disable continuous change of Velocity (Velocity), Acceleration (Acceleration), and Deceleration (Deceleration) is set. Continuous update re-imports input variables to the instance in execution without aborting the operation. • FALSE: Disabled • TRUE: Enabled
LinearAxes	Linear interpolation axis	INT[0..15]	At start	1 to 16	0	The axes to be used in linear interpolation control are set from the structuring axes. Specify the index numbers (1 to 16) of the structuring axes in the array. *: When Velocity mode (VelocityMode) is set to "2: Reference axis speed (ReferenceAxisSpeed)", the first element of the array is regarded as the reference axis. For details, refer to the following. ☞ Page 1693 Linear interpolation axis (LinearAxes)

Input variable	Name	Data type	Input	Valid range	Default value	Description
Position	Target position	LREAL[0..15]	At start	-10000000000.0 to 10000000000.0	0.0	The target position for linear interpolation is set. The value is set as one-dimensional array data. The data is treated as the absolute position for structuring axes 1 to 16. For details, refer to the following. ☞ Page 1694 Target position (Position)
Velocity	Velocity	LREAL	At start/Retrigger possible/ Continuous update possible	0.0, 0.0001 to 2500000000.0	0.0	The speed command value is set. For details, refer to the following. ☞ Page 1694 Velocity (Velocity)
Acceleration	Acceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The acceleration is set. For details, refer to the following. ☞ Page 1694 Acceleration (Acceleration)
Deceleration	Deceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration is set. For details, refer to the following. ☞ Page 1695 Deceleration (Deceleration)
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	The jerk is set. For details, refer to the following. ☞ Page 1695 Jerk (Jerk)
VelocityMode	Speed mode	INT (MC_INTERPOLATE_SPEED_MODE)	At start	0 to 2	0	The speed mode for interpolation control is set. • 0: Vector speed (VectorSpeed) • 1: Long axis speed (LongAxisSpeed) • 2: Reference axis speed (ReferenceAxisSpeed) For details, refer to the following. ☞ Page 1696 Velocity mode (VelocityMode)
Direction	Direction selection	INT (MC_DIRECTION[0..15])	At start	1 to 3	0	The direction is set. The value is set as one-dimensional array data. The data is treated as the direction for structuring axes 1 to 16. • 1: Positive direction (mcPositiveDirection) • 2: Negative direction (mcNegativeDirection) • 3: Shortest path (mcShortestWay) *: If this setting is omitted, Out of Direction Selection Range (error code: 1AA5H) occurs. For details, refer to the following. ☞ Page 1698 Direction selection (Direction)
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0 to 5	0	The buffer mode is set. • 0: Aborting(mcAborting) • 1: Buffered(mcBuffered) • 2: BlendingLow(mcBlendingLow) • 3: BlendingPrevious(mcBlendingPrevious) • 4: BlendingNext(mcBlendingNext) • 5: BlendingHigh(mcBlendingHigh) For details, refer to the following. ☞ Page 1698 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	00000000H to 00010001H	00000000H	The function options for MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) are set in the specified bit position. For details, refer to the following. ☞ Page 1699 Options (Options)

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Completion	BOOL	FALSE	This variable becomes TRUE when the control is completed. The value differs depending on the status of Start (Execute) at operation completion. ■When Start (Execute) is TRUE This variable remains TRUE until Start (Execute) is set to FALSE. ■When Start (Execute) is FALSE This variable becomes TRUE only for one cycle.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) is controlling the axis. When multiple MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) FBs are executed for the same axes group, Controlling (Active) of only one MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) FB becomes TRUE.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) is aborted because of an error or multiple start.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

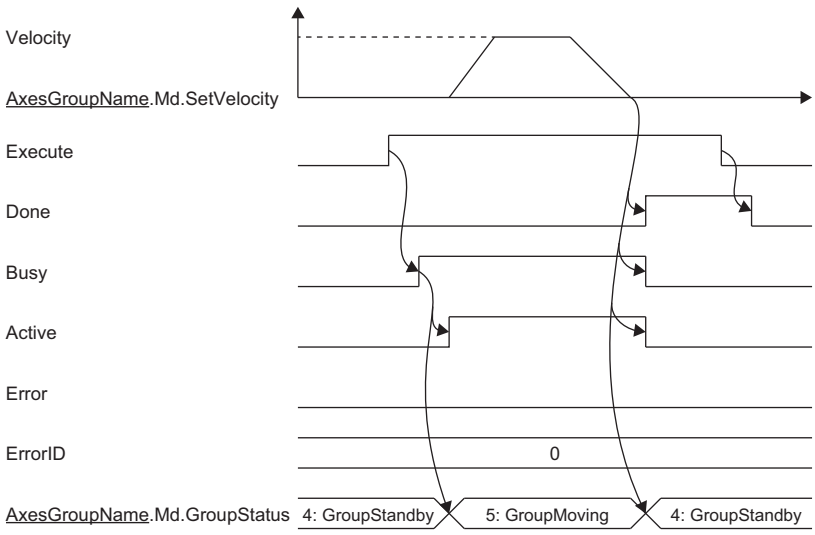
Processing details

- This FB specifies an axes group and performs interpolation control so that the path from the start point (move start point) to the end point becomes a straight line. In linear interpolation control, interpolation control using a maximum of 4 axes is performed.
- In MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control), specify the target position used for the absolute position and perform absolute value linear interpolation control.
- For the operation of Absolute Value Linear Interpolation Control for axes 1 to 3, refer to "Linear Interpolation Control" in the following.

📖 User's manual for the controller used

■Timing chart

- When the processing is completed successfully



- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Linear interpolation axis (LinearAxes)

In linear interpolation control, linear interpolation is performed using arbitrary axes from the structuring axes set to the axes group.

Set structuring axes to perform linear interpolation by Linear interpolation axis (LinearAxes) from the structuring axes of the axes group. Linear interpolation axis (LinearAxes) has 16 array elements. For the arrays, first set only the index numbers (1 to 16: structuring axes 1 to 16) of structuring axes to be used for interpolation control to the front, then set the rest of the arrays to "0". Arrays set to "0" can be omitted.

- The setting number of the linear interpolation axes must be set to the maximum number of the interpolation axes for interpolation control (4 axes) or less.
- The number of structuring axes to be used for interpolation control must be the number of axes registered to the structuring axes or less.
- When Velocity mode (VelocityMode) is "2: Reference axis speed (ReferenceAxisSpeed)", the structuring axis of the first element in Linear interpolation axis (LinearAxes) becomes the reference axis.
- If the following is specified, Linear Interpolation Axis Setting Incorrect (error code: 1B08H) occurs and the axis will not start.
  - When specifying a configuration axis where an axis is not set
  - When the first element of Linear interpolation axis (LinearAxes) is "0"
  - When specifying the same index No. to Linear interpolation axis (LinearAxes) in duplicate
  - When the number of axes set in Linear interpolation axis (LinearAxes) exceeds the maximum number of interpolation axes

Ex.

When structuring axis 2, 3, and 4 are set to the linear interpolation axis

LinearAxes[0]:= 2;

LinearAxes[1]:= 3;

LinearAxes[2]:= 4;

LinearAxes[3]:= 0;*1

⋮

LinearAxes[15]:= 0;*1

*1 "LinearAxes[3]:= 0; to LinearAxes[15]:= 0;" can be omitted.

■Target position (Position)

The target position for linear interpolation is set.

Target position (Position) has 16 array elements.

- The valid range differs depending on whether the software stroke limit is valid/invalid, Direction selection (Direction), and the setting for Target position specification exceeding the ring counter (Options (Options): Bit 16). For details, refer to the following.

☞ Page 1625 Target position (Position)

- Position[0..15] indicates the target positions of structuring axes 1 to 16. Set the position in relation to the structuring axes set in Linear interpolation axis (LinearAxes).
- Target position (Position) of structuring axes not set to the linear interpolation axes is ignored.
- Target position (Position) of axes after all structuring axes set to the linear interpolation axes can be omitted.

Ex.

When setting structuring axes 2, 3, and 4 to the linear interpolation axes, and setting the target positions of structuring axes 2, 3, and 4 to 2000.0, 3000.0, and 4000.0

```
LinearAxes[0]:= 2;
LinearAxes[1]:= 3;
LinearAxes[2]:= 4;
LinearAxes[3]:= 0;*1
⋮
LinearAxes[15]:= 0;*1
Position[0]:= 0.0;
Position[1]:= 2000.0;
Position[2]:= 3000.0;
Position[3]:= 4000.0;
Position[4]:= 0.0;*1
⋮
Position[15]:= 0.0;*1
```

*1 "LinearAxes[3]:= 0; to LinearAxes[15]:= 0;" and "Position[4]:= 0.0; to Position[15]:= 0.0;" can be omitted.

■Velocity (Velocity)

The maximum speed of the path used in MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) is set.

Setting range
0.0, 0.0001 to 2500000000.0*1*2

*1 To perform floating-point operation, the lower limit value of the command velocity is limited by the following restrictions.
If the velocity converted from the command velocity to operation cycle is less than 0.00001, Out of Operation Cycle Converted Velocity Range (error code: 1B16H) (when changing the velocity, Out of Operation Cycle Converted Velocity Range Warning (event code: 00D2FH)) occurs. To improve the precision of floating-point operation, change Position command unit (AxisName.Pr.Unit_Position) and Speed command unit (AxisName.Pr.Unit_Velocity) and perform settings so that the velocity converted into the operation cycle is not less than "0.00001".

*2 When a motion control FB with a commanded velocity of "0.0" performs a multiple start, the commanded velocity of the previous motion control FB is used.

■Acceleration (Acceleration)

The acceleration used in MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) is set.

The setting range differs depending on the setting for Acceleration/deceleration method setting (Options (Options): Bit 0 to 2).

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000*1, positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	0.000000*1, positive numbers from 0.000001 to 8400.0 [s]

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Deceleration (Deceleration)

The deceleration used in MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) is set.
Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Deceleration (Deceleration) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000*1, positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

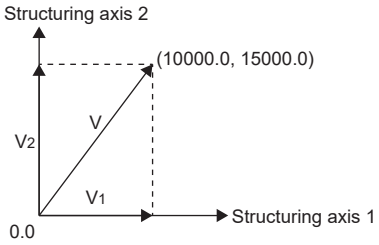
■Jerk (Jerk)

The jerk used in MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) is set.
Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Jerk (Jerk) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000, positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

■Velocity mode (VelocityMode)

The velocity mode used in linear interpolation control is set. When specifying "2: Reference axis speed (ReferenceAxisSpeed)" in Velocity mode (VelocityMode), the reference axis becomes the structuring axis specified to the first element of Linear interpolation axis (LinearAxes).

Setting value	Description
0: Vector speed (VectorSpeed)	Velocity (Velocity) is set as the vector speed. The positioning speed (V_n) of each axis is calculated from the movement amount (D_n) of each axis by the motion system based on the positioning speed (V) of the set control target. Example: When using 2-axis linear interpolation control  • Structuring axis 1 movement amount (D_1): 10000.0 [pulse] • Structuring axis 2 movement amount (D_2): 15000.0 [pulse] • Axes group speed unit: [s] • Vector speed (V): 7000.0 [pulse/s] For the above conditions, the positioning speed of each axis is calculated by the motion system with the following calculation formulas. • Structuring axis 1 positioning speed: $V_1 = V \times D_1 / \sqrt{D_1^2 + D_2^2}$ • Structuring axis 2 positioning speed: $V_2 = V \times D_2 / \sqrt{D_1^2 + D_2^2}$ ■Point • For the vector speed specification, the set speed limit value is valid as the vector speed for Velocity (Velocity).
1: Long axis speed (LongAxisSpeed)	Velocity (Velocity) is set as the velocity of the long axis. The control is carried out based on the positioning speed (the long axis velocity: V) of the interpolation axis whose movement amount is the largest in the specified positions set in each interpolation axis. The positioning speed (V_n) of the other interpolation axes is calculated by the motion system from the movement amount (D_n) of each interpolation axis. Example: When using 4-axis linear interpolation control • Structuring axis 1 movement amount (D_1): 10000.0 [pulse] • Structuring axis 2 movement amount (D_2): 15000.0 [pulse] • Structuring axis 3 movement amount (D_3): 5000.0 [pulse] • Structuring axis 4 movement amount (D_4): 20000.0 [pulse] • Structuring axis 4 speed unit: [s] • Long axis speed (V): 7000.0 [pulse/s] In the above case, the long axis is structuring axis 4 whose movement amount is the largest, and the control is carried out based on the long axis velocity of structuring axis 4. The positioning speed of the other structuring axes is calculated by the motion system with the following calculation formulas. • Structuring axis 1 positioning speed: $V_1 = D_1 / D_4 \times V$ • Structuring axis 2 positioning speed: $V_2 = D_2 / D_4 \times V$ • Structuring axis 3 positioning speed: $V_3 = D_3 / D_4 \times V$ ■Point • For the long axis speed specification, the set speed limit value is valid as the long axis velocity for Velocity (Velocity). • Note that the vector speed may be larger than the speed limit value when specifying Long axis speed. When a value is set as shown below in the 2-axis linear interpolation, the vector velocity exceeds the velocity limit value. Example: When the setting items and the setting values are as follows. • Linear interpolation axes: Configuration axis 1, Configuration axis 2 • Configuration axis 1 movement amount: 100 [pulses] • Configuration axis 2 movement amount: 200 [pulses] • Long axis velocity: 50 [pulses/s] • Velocity limit value of configuration axis 2: 55 [pulses/s] In the above case, the reference axis is configuration axis 2 whose movement amount is the largest, and the control is carried out with the velocity limit value set for configuration axis 2. The positioning speed and the vector velocity of each axis are shown below. • Configuration axis 1 positioning speed: $100/200 \times 50 = 25$ [pulses/s] • Configuration axis 2 positioning speed: 50 [pulses/s] • Vector velocity: $\sqrt{(25^2 + 50^2)} = 55.9$ [pulses/s] The vector velocity becomes a value that exceeds the velocity limit value "55" of configuration axis 2.

Setting value	Description
2: Reference axis speed (ReferenceAxisSpeed)	Velocity (Velocity) is set as the velocity of the reference axis. The motion system calculates the positioning speed (Vn) of the other interpolation axes from the movement amount (Dn) of each interpolation axis and performs control based on the positioning speed of the set reference axis (the reference axis velocity: V). If "0.0" is set for the movement amount of the configuration axis specified as the reference axis, Reference Axis Movement Amount 0 (error code: 1AFAH) occurs. Example: When using 4-axis linear interpolation control • Structuring axis 1 movement amount (D_1): 10000.0 [pulse] • Structuring axis 2 movement amount (D_2): 15000.0 [pulse] • Structuring axis 3 movement amount (D_3): 5000.0 [pulse] • Structuring axis 4 movement amount (D_4): 20000.0 [pulse] • Reference axis velocity (V): 7000.0 [pulse/s] In the above case, the long axis is structuring axis 4 whose movement amount is the largest, and the control is carried out based on the positioning speed set for structuring axis 4. The positioning speed of the other axes is calculated by the motion system with the following calculation formulas. • Structuring axis 1 positioning speed: $V_1 = D_1 / D_4 \times V$ • Structuring axis 2 positioning speed: $V_2 = D_2 / D_4 \times V$ • Structuring axis 3 positioning speed: $V_3 = D_3 / D_4 \times V$ ■Point • For the reference axis speed specification, the set speed limit value is valid as the reference axis velocity for Velocity (Velocity). • Note that the positioning speed may be larger than the set reference axis velocity for axes with a movement amount larger than the reference axis. When a value is set as shown below in the 2-axis linear interpolation, the positioning speed and the vector velocity of configuration axis 2 exceed the velocity limit value. Example: When the setting items and the setting values are as follows. • Linear interpolation axes: Configuration axis 1, Configuration axis 2 • Configuration axis 1 movement amount: 100 [pulses] • Configuration axis 2 movement amount: 200 [pulses] • Reference axis velocity: 50 [pulses/s] • Velocity limit value of configuration axis 1: 55 [pulses/s] In the above case, the reference axis is configuration axis 1, and control is carried out with the velocity limit value set for configuration axis 1. The positioning speed and the vector velocity of each axis are shown below. • Configuration axis 1 positioning speed: 50 [pulses/s] • Configuration axis 2 positioning speed: 200/100 × 50 = 100 [pulses/s] • Vector velocity: $\sqrt{(50^2 + 100^2)} = 111.8$ [pulses/s] The positioning speed and the vector velocity of configuration axis 2 become a value that exceeds the velocity limit value "55" of configuration axis 1.

Restriction

- When "0: Vector speed (VectorSpeed)" is set in Velocity mode (VelocityMode) and the movement value of each axis exceeds "4,294,967,296.0 ($=2^{32}$)", Out of Linear Movement Amount Range (error code: 1B10H) will occur and the axis will not start.
- When "0: Aborting (mcAborting)" is set in Buffer mode (BufferMode) and MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) is executed by multiple start, specify "0: Vector speed (VectorSpeed)" in Velocity mode (VelocityMode). If a value other than "0: Vector speed (VectorSpeed)" is set, Out of Speed Mode Range (error code: 1AC9H) occurs.

■Direction selection (Direction)

The direction used to move from the current position to the target position is set by selecting positive direction, negative direction, or shortest way. Direction selection (Direction) has 16 array elements.

- Direction[0..15] indicates the direction selection of structuring axes 1 to 16. Set the value for the structuring axis set in Linear interpolation axis (LinearAxes).
- The direction selection (Direction) of the structuring axis which is not set as the linear interpolation axis is ignored.
- Direction selection (Direction) of axes after all structuring axes set to the linear interpolation axes can be omitted.
- If a setting value is not set, Out of Direction Selection Range (error code: 1AA5H) occurs and the axis will not start.
- A linear axis where the software stroke limit is valid ignores the setting of Direction selection (Direction). For details on operation of each direction selection, refer to the following.

☞ Page 1627 Direction selection (Direction)

Setting value	Description
1: Positive direction (mcPositiveDirection)	Positioning is performed from the current position to the target position in the positive direction (address increasing).
2: Negative direction (mcNegativeDirection)	Positioning is performed from the current position to the target position in the negative direction (address decreasing).
3: Shortest path (mcShortestWay)	Based on the current position, positioning control is performed in the direction closest to the target position.

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following buffer modes are settable in MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control).

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.
1: Buffered(mcBuffered)	Buffers the next FB on the under-control FB. If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after completion of the under-control FB.
2: BlendingLow(mcBlendingLow)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the lower target velocity between the under-control FB and the buffering FB is used as the switching speed.
3: BlendingPrevious(mcBlendingPrevious)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the under-control FB is used as the switching speed.
4: BlendingNext(mcBlendingNext)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the buffering FB is used as the switching speed.
5: BlendingHigh(mcBlendingHigh)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the higher target velocity between the under-control FB and the buffering FB is used as the switching speed.

*1 Stops are not performed between the under-control FB and the buffering FB.

Point

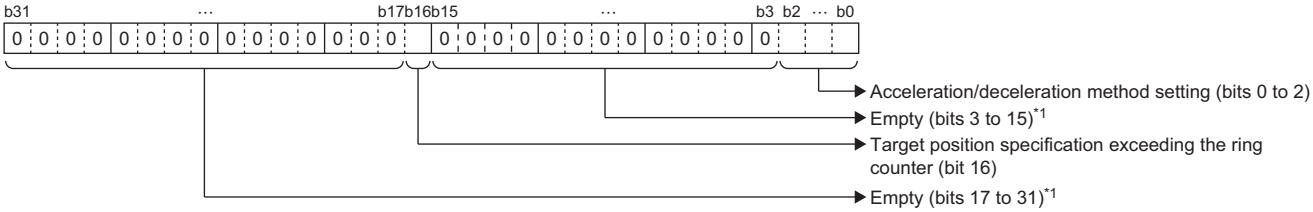
For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

■Options (Options)

The function options for MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
0 to 2	Acceleration/deceleration method setting	The acceleration/deceleration method used to perform control is set. • 0: Acceleration/deceleration specification method (mcAccDec) • 1: Acceleration/deceleration time-fixed method (mcFixedTime)
16	Target position specification exceeding the ring counter	When the software stroke limit is invalid, whether to allow a target position that exceeds the ring counter upper limit value or lower limit value is set. • 0: Disabled • 1: Enabled

• Acceleration/deceleration method setting (bit 0 to 2)

Setting value	Description
0: Acceleration/deceleration specification method (mcAccDec)	An acceleration/deceleration method using Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk) set in MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control).
1: Acceleration/deceleration time-fixed method (mcFixedTime)	An acceleration/deceleration method using the acceleration/deceleration time specified in MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) regardless of the velocity. The acceleration/deceleration time does not use Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk).

■Necessary object data

When using MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control), set the following object data for all structuring axes in the specified axes group.

• Target position (607AH)

If there is a structuring axis for which no object data is set, Necessary Object Data Unset (error code: 1AF7H) occurs and the axis will not start.

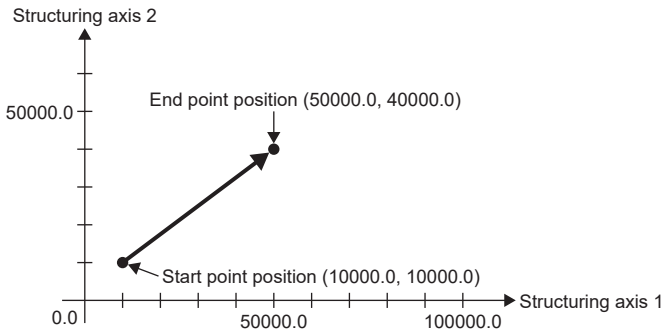
For details on object data settings, refer to the following.

📖 User's manual for the controller used

Program example

The following is a program example. The linear interpolation control start (bLinearInterpolateCMD) is set to TRUE, and Axes group 1 (AxesGroup001) is enabled. Then, 2-axis linear interpolation absolute positioning control is performed for Axis group 1 (AxesGroup001).

• Operation



• Axes group

Item	Setting value
Structuring axis[1]	Axis0001
Structuring axis[2]	Axis0002

• Setting

Item	Setting value	
	Structuring axis 1	Structuring axis 2
Target position	50000.0	40000.0
Velocity	50000.0	
Acceleration	50000.0	
Deceleration	50000.0	
Jerk	0.0	
Direction selection	Positive direction	Positive direction

■Axes

Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1
2	Axis0002	AXIS_REF	Axis 2

■Axes group

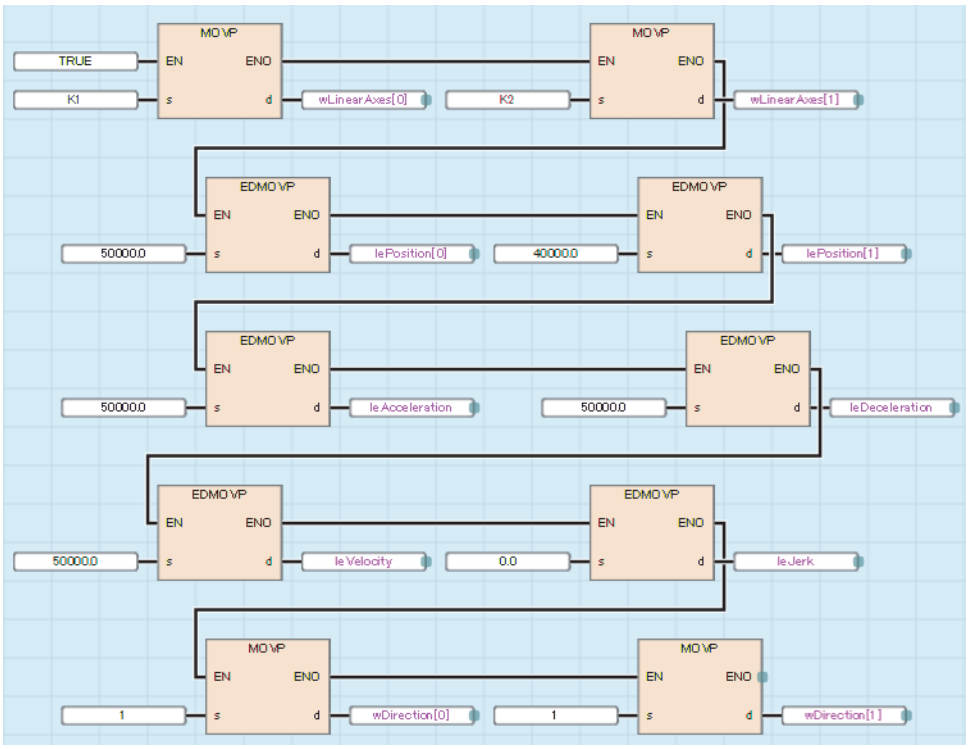
Axes group No.	Label name	Data type	Comment
1	AxesGroup001	AXES_GROUP_REF	Axes group 1

■Labels used

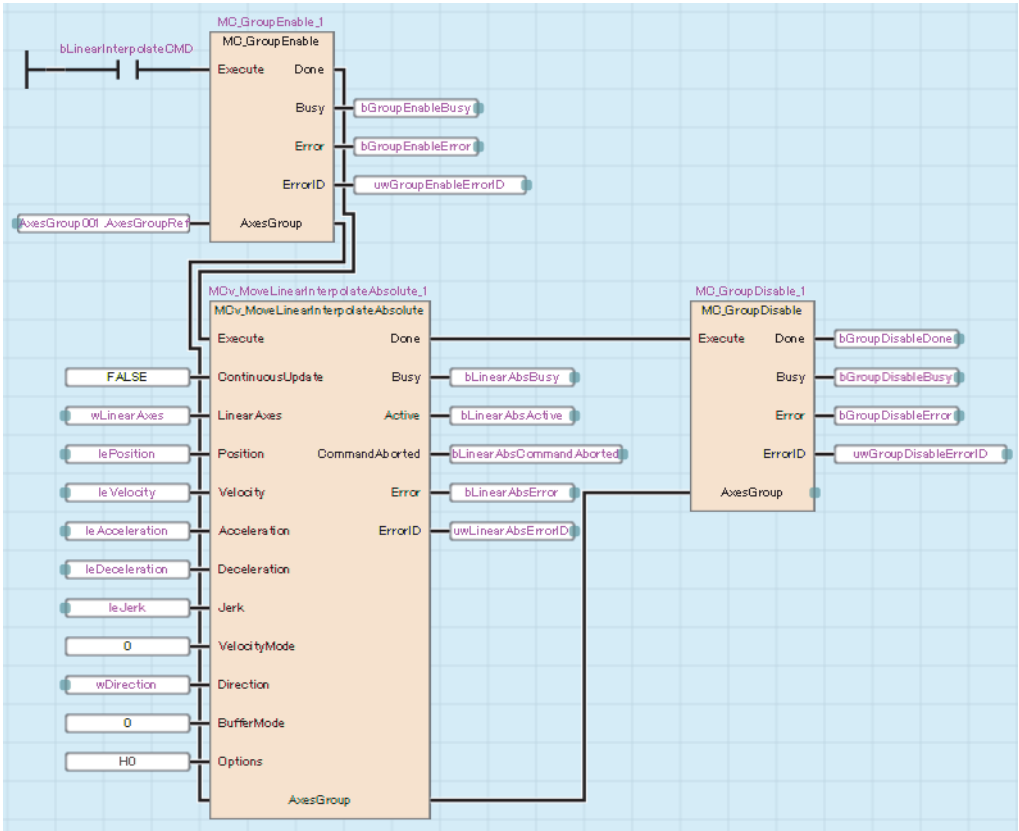
Label name	Data type	Comment
MCv_MoveLinearInterpolateAbsolute_1	MCv_MoveLinearInterpolateAbsolute	Absolute Value Linear Interpolation Control FB
wLinearAxes	Word [signed](0..15)	Linear interpolation axis
lePosition	Double-precision real number(0..15)	Target position
leVelocity	Double-precision real number	Velocity
leAcceleration	Double-precision real number	Acceleration
leDeceleration	Double-precision real number	Deceleration
leJerk	Double-precision real number	Jerk
wDirection	Word [signed](0..15)	Direction selection
bLinearAbsDone	Bit	Execution completion
bLinearAbsBusy	Bit	Executing
bLinearAbsActive	Bit	Controlling
bLinearAbsCommandAborted	Bit	Abortion of execution
bLinearAbsError	Bit	Error
uwLinearAbsErrorID	Word [unsigned]/bit string [16-bit]	Error code
MC_GroupEnable_1	MC_GroupEnable	Axes Group Enabled FB
bLinearInterpolateCMD	Bit	Linear interpolation control start
bGroupEnableDone	Bit	Axes Group Enabled Completion
bGroupEnableBusy	Bit	Executing
bGroupEnableError	Bit	Error
uwGroupEnableErrorID	Word [unsigned]/bit string [16-bit]	Error code
MC_GroupDisable_1	MC_GroupDisable	Axes Group Disabled FB
bGroupDisableDone	Bit	Axes Group Disabled Completion
bGroupDisableBusy	Bit	Executing
bGroupDisableError	Bit	Error
uwGroupDisableErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data setting for 2-axis linear interpolation control



• Axes Group Enabled/2-axis linear interpolation control/Axis Group Disabled



■ST program

```
//-----Data setting for 2-axis linear interpolation control-----
wLinearAxes[0]:= 1;
wLinearAxes[1]:= 2;
lePosition[0]:= 50000.0;
lePosition[1]:= 40000.0;
leVelocity:= 50000.0;
leAcceleration:= 50000.0;
leDeceleration:= 50000.0;
leJerk:= 0.0;
wDirection[0]:= MC_DIRECTION__mcPositiveDirection;
wDirection[1]:= MC_DIRECTION__mcPositiveDirection;

//-----Axes Group Enabled-----
MC_GroupEnable_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bLinearInterpolateCMD ,
   Done=> bGroupEnableDone ,
   Busy=> bGroupEnableBusy ,
   Error=> bGroupEnableError ,
   ErrorID=> uwGroupEnableErrorID
);

//-----2-axis linear interpolation control-----
MCv_MoveLinearInterpolateAbsolute_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bGroupEnableDone ,
   ContinuousUpdate:= FALSE ,
   LinearAxes:= wLinearAxes ,
   Position:= lePosition ,
   Velocity:= leVelocity ,
   Acceleration:= leAcceleration ,
   Deceleration:= leDeceleration ,
   Jerk:= leJerk ,
   VelocityMode:= MC_INTERPOLATE_SPEED_MODE__VectorSpeed ,
   Direction:= wDirection ,
   BufferMode:= MC_BUFFER_MODE__mcAborting ,
   Options:= H00000000 ,
   Done=> bLinearAbsDone ,
   Busy=> bLinearAbsBusy ,
   Active=> bLinearAbsActive ,
   CommandAborted=> bLinearAbsCommandAborted ,
   Error=> bLinearAbsError ,
   ErrorID=> uwLinearAbsErrorID
);
```

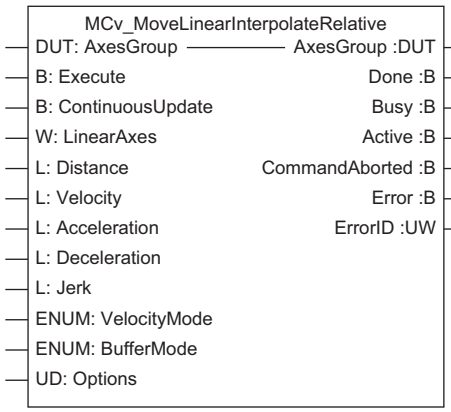
```
//-----Axes Group Disabled-----
MC_GroupDisable_1(
    AxesGroup:= AxesGroup001.AxesGroupRef ,
    Execute:= bLinearAbsDone ,
    Done=> bGroupDisableDone ,
    Busy=> bGroupDisableBusy ,
    Error=> bGroupDisableError ,
    ErrorID=> uwGroupDisableErrorID
);
```

42.12 Relative Value Linear Interpolation Control

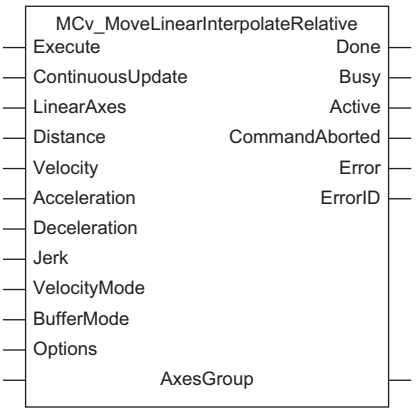
MCv_MoveLinearInterpolateRelative

This FB specifies the movement amount of the relative position of the set axes group and performs positioning by linear interpolation control.

Ladder



FBD/LD



ST

```
MCv_MoveLinearInterpolateRelative(
  AxesGroup:= ?AXES_GROUP_REF? ,
  Execute:= ?BOOL? ,
  ContinuousUpdate:= ?BOOL? ,
  LinearAxes:= ?INT(0..15)? ,
  Distance:= ?LREAL(0..15)? ,
  Velocity:= ?LREAL? ,
  Acceleration:= ?LREAL? ,
  Deceleration:= ?LREAL? ,
  Jerk:= ?LREAL? ,
  VelocityMode:= ?INT? ,
  BufferMode:= ?INT? ,
  Options:= ?DWORD? ,
  Done=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
AxesGroup	Axes group information	AXES_GROUP_REF	At start	—	Omission not possible	An axes group is set. For the variable (AxesGroupName.AxesGroupRef.) used, refer to the following. ☞ Page 1423 AxesGroupName.AxesGroupRef. (Axes group information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Start	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) is executed.
ContinuousUpdate	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	Whether to enable or disable continuous change of Velocity (Velocity), Acceleration (Acceleration), and Deceleration (Deceleration) is set. Continuous update re-imports input variables to the instance in execution without aborting the operation. • FALSE: Disabled • TRUE: Enabled
LinearAxes	Linear interpolation axis	INT[0..15]	At start	1 to 16	0	The axes to be used in linear interpolation control are set from the structuring axes. Specify the index numbers (1 to 16) of the structuring axes in the array. *: When Velocity mode (VelocityMode) is set to "2: Reference axis speed (ReferenceAxisSpeed)", the first element of the array is regarded as the reference axis. For details, refer to the following. ☞ Page 1708 Linear interpolation axis (LinearAxes)

Input variable	Name	Data type	Input	Valid range	Default value	Description
Distance	Movement amount	LREAL[0..15]	At start	-10000000000.0 to 10000000000.0	0.0	The movement amount from the current position at start to the end point is set. The value is set as one-dimensional array data. The data is treated as the relative position for structuring axes 1 to 16. For details, refer to the following. ☞ Page 1709 Movement amount (Distance)
Velocity	Velocity	LREAL	At start/Retrigger possible/ Continuous update possible	0.0, 0.0001 to 2500000000.0	0.0	The speed command value is set. For details, refer to the following. ☞ Page 1710 Velocity (Velocity)
Acceleration	Acceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The acceleration is set. For details, refer to the following. ☞ Page 1710 Acceleration (Acceleration)
Deceleration	Deceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration is set. For details, refer to the following. ☞ Page 1710 Deceleration (Deceleration)
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	The jerk is set. For details, refer to the following. ☞ Page 1710 Jerk (Jerk)
VelocityMode	Speed mode	INT (MC_INTERPOLATE_SPEED_MODE)	At start	0 to 2	0	The speed mode for interpolation control is set. • 0: Vector speed (VectorSpeed) • 1: Long axis speed (LongAxisSpeed) • 2: Reference axis speed (ReferenceAxisSpeed) For details, refer to the following. ☞ Page 1711 Velocity mode (VelocityMode)
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0 to 5	0	The buffer mode is set. • 0: Aborting(mcAborting) • 1: Buffered(mcBuffered) • 2: BlendingLow(mcBlendingLow) • 3: BlendingPrevious(mcBlendingPrevious) • 4: BlendingNext(mcBlendingNext) • 5: BlendingHigh(mcBlendingHigh) For details, refer to the following. ☞ Page 1713 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	00000000H to 00000005H	00000000H	The function options for MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) are set in the specified bit position. For details, refer to the following. ☞ Page 1714 Options (Options)

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Completion	BOOL	FALSE	This variable becomes TRUE when the control is completed. The value differs depending on the status of Start (Execute) at operation completion. ■When Start (Execute) is TRUE This variable remains TRUE until Start (Execute) is set to FALSE. ■When Start (Execute) is FALSE This variable becomes TRUE only for one cycle.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) is controlling the axis. When multiple MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) FBs are executed for the same axes group, Controlling (Active) of only one MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) FB becomes TRUE.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) is aborted because of an error or multiple start.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

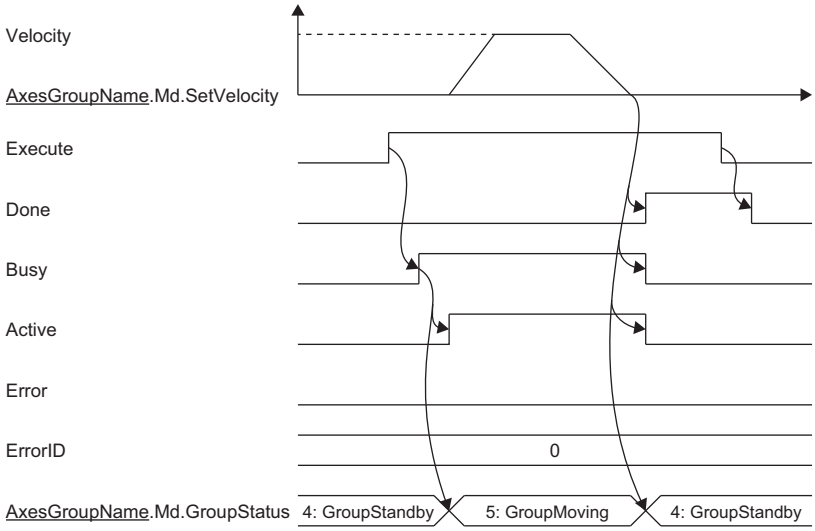
Processing details

- This FB specifies an axes group and performs interpolation control so that the path from the start point (move start point) to the end point becomes a straight line. In linear interpolation control, interpolation control using a maximum of 4 axes is performed.
- In MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control), specify the relative movement amount from the current position and perform relative value linear interpolation control.
- For the operation of Relative Value Linear Interpolation Control for axes 1 to 3, refer to "Linear Interpolation Control" in the following.

📖 User's manual for the controller used

■Timing chart

- When the processing is completed successfully



- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Linear interpolation axis (LinearAxes)

In linear interpolation control, linear interpolation is performed using arbitrary axes from the structuring axes set to the axes group.

Set structuring axes to perform linear interpolation by Linear interpolation axis (LinearAxes) from the structuring axes of the axes group. Linear interpolation axis (LinearAxes) has 16 array elements. For the arrays, first set only the index numbers (1 to 16: structuring axes 1 to 16) of structuring axes to be used for interpolation control to the front, then set the rest of the arrays to "0". Arrays set to "0" can be omitted.

- The setting number of the linear interpolation axes must be set to the maximum number of the interpolation axes for interpolation control (4 axes) or less.
- The number of structuring axes to be used for interpolation control must be the number of axes registered to the structuring axes or less.
- When Velocity mode (VelocityMode) is "2: Reference axis speed (ReferenceAxisSpeed)", the structuring axis of the first element in Linear interpolation axis (LinearAxes) becomes the reference axis.
- If the following is specified, Linear Interpolation Axis Setting Incorrect (error code: 1B08H) occurs and the axis will not start.
  - When specifying a configuration axis where an axis is not set
  - When the first element of Linear interpolation axis (LinearAxes) is "0"
  - When specifying the same index No. to Linear interpolation axis (LinearAxes) in duplicate
  - When the number of axes set in Linear interpolation axis (LinearAxes) exceeds the maximum number of interpolation axes

Ex.

When structuring axis 2, 3, and 4 are set to the linear interpolation axis

LinearAxes[0]:= 2;
LinearAxes[1]:= 3;
LinearAxes[2]:= 4;
LinearAxes[3]:= 0;*1
⋮
LinearAxes[15]:= 0;*1

*1 "LinearAxes[3]:= 0; to LinearAxes[15]:= 0;" can be omitted.

■Movement amount (Distance)

The movement amount from the current position at start to the end point is set. Movement amount (Distance) has 16 array elements.

- The input range for Movement amount (Distance) is -10000000000.0 to 10000000000.0.
- Distance[0..15] indicates the movement amount of structuring axes 1 to 16. Set the position in relation to the structuring axes set in Linear interpolation axis (LinearAxes).
- Movement amount (Distance) of structuring axes not set to the linear interpolation axes is ignored.
- Movement amount (Distance) of axes after all structuring axes set to the linear interpolation axes can be omitted.

Ex.

When setting structuring axes 2, 3, and 4 to the linear interpolation axes, and setting the target positions of structuring axes 2, 3, and 4 to 2000.0, 3000.0, and 4000.0

```
LinearAxes[0]:= 2;
LinearAxes[1]:= 3;
LinearAxes[2]:= 4;
LinearAxes[3]:= 0;*1
⋮
LinearAxes[15]:= 0;*1
Distance[0]:= 0.0;
Distance[1]:= 2000.0;
Distance[2]:= 3000.0;
Distance[3]:= 4000.0;
Distance[4]:= 0.0;*1
⋮
Distance[15]:= 0.0;*1
```

*1 "LinearAxes[3]:= 0; to LinearAxes[15]:= 0;" and "Distance[4]:= 0.0; to Distance[15]:= 0.0;" can be omitted.

■Velocity (Velocity)

The maximum speed of the path used in MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) is set.

Setting range
0.0, 0.0001 to 2500000000.0[*1*2]

*1 To perform floating-point operation, the lower limit value of the command velocity is limited by the following restrictions.
If the velocity converted from the command velocity to operation cycle is less than 0.00001, Out of Operation Cycle Converted Velocity Range (error code: 1B16H) (when changing the velocity, Out of Operation Cycle Converted Velocity Range Warning (event code: 00D2FH)) occurs. To improve the precision of floating-point operation, change Position command unit (AxisName.Pr.Unit_Position) and Speed command unit (AxisName.Pr.Unit_Velocity) and perform settings so that the velocity converted into the operation cycle is not less than "0.00001".

*2 When a motion control FB with a commanded velocity of "0.0" performs a multiple start, the commanded velocity of the previous motion control FB is used.

■Acceleration (Acceleration)

The acceleration used in MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) is set.
The setting range differs depending on the setting for Acceleration/deceleration method setting (Options (Options): Bit 0 to 2).

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	0.000000[*1], positive numbers from 0.000001 to 8400.0 [s]

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Deceleration (Deceleration)

The deceleration used in MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) is set.
Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Deceleration (Deceleration) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

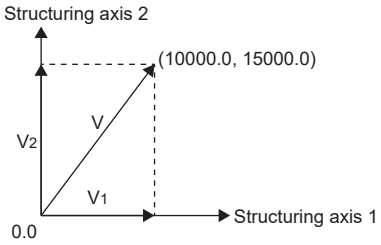
■Jerk (Jerk)

The jerk used in MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) is set.
Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Jerk (Jerk) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000, positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

■Velocity mode (VelocityMode)

The velocity mode used in linear interpolation control is set. When specifying "2: Reference axis speed (ReferenceAxisSpeed)" in Velocity mode (VelocityMode), the reference axis becomes the structuring axis specified to the first element of Linear interpolation axis (LinearAxes).

Setting value	Description
0: Vector speed (VectorSpeed)	Velocity (Velocity) is set as the vector speed. The positioning speed (V_n) of each axis is calculated from the movement amount (D_n) of each axis by the motion system based on the positioning speed (V) of the set control target. Example: When using 2-axis linear interpolation control  • Structuring axis 1 movement amount (D_1): 10000.0 [pulse] • Structuring axis 2 movement amount (D_2): 15000.0 [pulse] • Axes group speed unit: [s] • Vector speed (V): 7000.0 [pulse/s] For the above conditions, the positioning speed of each axis is calculated by the motion system with the following calculation formulas. • Structuring axis 1 positioning speed: $V_1 = V \times D_1 / \sqrt{D_1^2 + D_2^2}$ • Structuring axis 2 positioning speed: $V_2 = V \times D_2 / \sqrt{D_1^2 + D_2^2}$ ■Point • For the vector speed specification, the set speed limit value is valid as the vector speed for Velocity (Velocity).
1: Long axis speed (LongAxisSpeed)	Velocity (Velocity) is set as the velocity of the long axis. The control is carried out based on the positioning speed (the long axis velocity: V) of the interpolation axis whose movement amount is the largest in the specified positions set in each interpolation axis. The positioning speed (V_n) of the other interpolation axes is calculated by the motion system from the movement amount (Dn) of each interpolation axis. Example: When using 4-axis linear interpolation control • Structuring axis 1 movement amount (D_1): 10000.0 [pulse] • Structuring axis 2 movement amount (D_2): 15000.0 [pulse] • Structuring axis 3 movement amount (D_3): 5000.0 [pulse] • Structuring axis 4 movement amount (D_4): 20000.0 [pulse] • Structuring axis 4 speed unit: [s] • Long axis speed (V): 7000.0 [pulse/s] In the above case, the long axis is structuring axis 4 whose movement amount is the largest, and the control is carried out based on the long axis velocity of structuring axis 4. The positioning speed of the other structuring axes is calculated by the motion system with the following calculation formulas. • Structuring axis 1 positioning speed: $V_1 = D_1 / D_4 \times V$ • Structuring axis 2 positioning speed: $V_2 = D_2 / D_4 \times V$ • Structuring axis 3 positioning speed: $V_3 = D_3 / D_4 \times V$ ■Point • For the long axis speed specification, the set speed limit value is valid as the long axis velocity for Velocity (Velocity). • Note that the vector speed may be larger than the speed limit value when specifying Long axis speed. When a value is set as shown below in the 2-axis linear interpolation, the vector velocity exceeds the velocity limit value. Example: When the setting items and the setting values are as follows. • Linear interpolation axes: Configuration axis 1, Configuration axis 2 • Configuration axis 1 movement amount: 100 [pulses] • Configuration axis 2 movement amount: 200 [pulses] • Long axis velocity: 50 [pulses/s] • Velocity limit value of configuration axis 2: 55 [pulses/s] In the above case, the reference axis is configuration axis 2 whose movement amount is the largest, and the control is carried out with the velocity limit value set for configuration axis 2. The positioning speed and the vector velocity of each axis are shown below. • Configuration axis 1 positioning speed: 100/200 × 50 = 25 [pulses/s] • Configuration axis 2 positioning speed: 50 [pulses/s] • Vector velocity: $\sqrt{(25^2 + 50^2)} = 55.9$ [pulses/s] The vector velocity becomes a value that exceeds the velocity limit value "55" of configuration axis 2.

Setting value	Description
2: Reference axis speed (ReferenceAxisSpeed)	Velocity (Velocity) is set as the velocity of the reference axis. The motion system calculates the positioning speed (Vn) of the other interpolation axes from the movement amount (Dn) of each interpolation axis and performs control based on the positioning speed of the set reference axis (the reference axis velocity: V). If "0.0" is set for the movement amount of the configuration axis specified as the reference axis, Reference Axis Movement Amount 0 (error code: 1AFAH) occurs. Example: When using 4-axis linear interpolation control • Structuring axis 1 movement amount (D_1): 10000.0 [pulse] • Structuring axis 2 movement amount (D_2): 15000.0 [pulse] • Structuring axis 3 movement amount (D_3): 5000.0 [pulse] • Structuring axis 4 movement amount (D_4): 20000.0 [pulse] • Reference axis velocity (V): 7000.0 [pulse/s] In the above case, the long axis is structuring axis 4 whose movement amount is the largest, and the control is carried out based on the positioning speed set for structuring axis 4. The positioning speed of the other axes is calculated by the motion system with the following calculation formulas. • Structuring axis 1 positioning speed: $V_1 = D_1 / D_4 \times V$ • Structuring axis 2 positioning speed: $V_2 = D_2 / D_4 \times V$ • Structuring axis 3 positioning speed: $V_3 = D_3 / D_4 \times V$ ■Point • For the reference axis speed specification, the set speed limit value is valid as the reference axis velocity for Velocity (Velocity). • Note that the positioning speed may be larger than the set reference axis velocity for axes with a movement amount larger than the reference axis. When a value is set as shown below in the 2-axis linear interpolation, the positioning speed and the vector velocity of configuration axis 2 exceed the velocity limit value. Example: When the setting items and the setting values are as follows. • Linear interpolation axes: Configuration axis 1, Configuration axis 2 • Configuration axis 1 movement amount: 100 [pulses] • Configuration axis 2 movement amount: 200 [pulses] • Reference axis velocity: 50 [pulses/s] • Velocity limit value of configuration axis 1: 55 [pulses/s] In the above case, the reference axis is configuration axis 1, and control is carried out with the velocity limit value set for configuration axis 1. The positioning speed and the vector velocity of each axis are shown below. • Configuration axis 1 positioning speed: 50 [pulses/s] • Configuration axis 2 positioning speed: 200/100 × 50 = 100 [pulses/s] • Vector velocity: $\sqrt{(50^2 + 100^2)} = 111.8$ [pulses/s] The positioning speed and the vector velocity of configuration axis 2 become a value that exceeds the velocity limit value "55" of configuration axis 1.

Restriction

- When "0: Vector speed (VectorSpeed)" is set in Velocity mode (VelocityMode) and the movement value of each axis exceeds "4,294,967,296.0 ($=2^{32}$)", Out of Linear Movement Amount Range (error code: 1B10H) will occur and the axis will not start.
- When "0: Aborting (mcAborting)" is set in Buffer mode (BufferMode) and MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) is executed by multiple start, specify "0: Vector speed (VectorSpeed)" in Velocity mode (VelocityMode). If a value other than "0: Vector speed (VectorSpeed)" is set, Out of Speed Mode Range (error code: 1AC9H) occurs.

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following buffer modes are settable in MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control).

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.
1: Buffered(mcBuffered)	Buffers the next FB on the under-control FB. If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after completion of the under-control FB.
2: BlendingLow(mcBlendingLow)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the lower target velocity between the under-control FB and the buffering FB is used as the switching speed.
3: BlendingPrevious(mcBlendingPrevious)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the under-control FB is used as the switching speed.
4: BlendingNext(mcBlendingNext)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the buffering FB is used as the switching speed.
5: BlendingHigh(mcBlendingHigh)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the higher target velocity between the under-control FB and the buffering FB is used as the switching speed.

*1 Stops are not performed between the under-control FB and the buffering FB.

Point

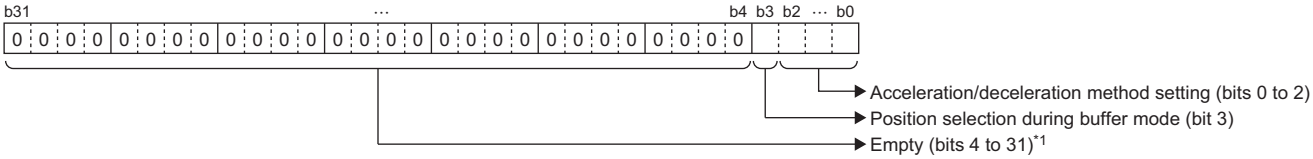
For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

■Options (Options)

The function options for MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



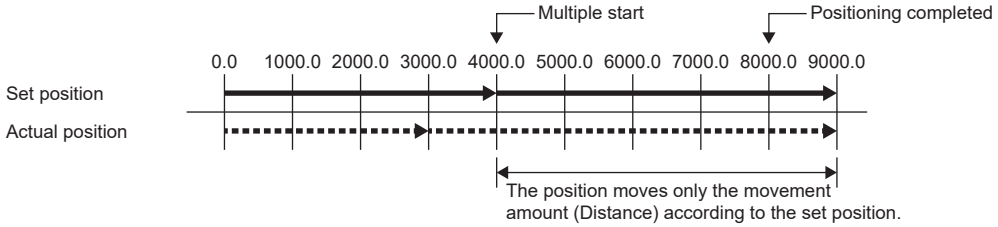
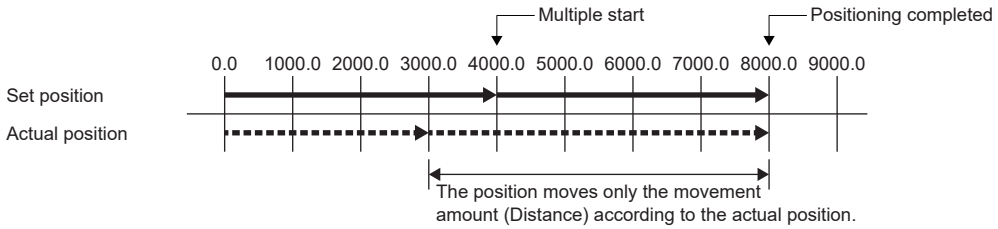
*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
0 to 2	Acceleration/deceleration method setting	The acceleration/deceleration method used to perform control is set. • 0: Acceleration/deceleration specification method (mcAccDec) • 1: Acceleration/deceleration time-fixed method (mcFixedTime)
3	Position selection during buffer mode	The position when executing relative positioning control by multiple start is set. • 0: Set position • 1: Actual position *: This bit is valid when setting "0: Aborting (mcAborting)" in Buffer mode (BufferMode).

• Acceleration/deceleration method setting (bit 0 to 2)

Setting value	Description
0: Acceleration/deceleration specification method (mcAccDec)	An acceleration/deceleration method using Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk) set in MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control).
1: Acceleration/deceleration time-fixed method (mcFixedTime)	An acceleration/deceleration method using the acceleration/deceleration time specified in MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) regardless of the velocity. The acceleration/deceleration time does not use Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk).

• Position selection during buffer mode (Bit 3)

Setting value	Description
0: Set position	Relative positioning control is performed from the set position. Example: When multiple start is performed with Movement amount (Distance) set to "5000.0" and Options (Options) set to "00000000H (Bit 3 is 0: Set position)" 
1: Actual position	Relative positioning control is performed from the actual position. Example: When multiple start is performed with Movement amount (Distance) set to "5000.0" and Options (Options) set to "00000008H (Bit 3 is 1: Actual position)". 

■Necessary object data

When using MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control), set the following object data for all structuring axes in the specified axes group.

- Target position (607AH)

If there is a structuring axis for which no object data is set, Necessary Object Data Unset (error code: 1AF7H) occurs and the axis will not start.

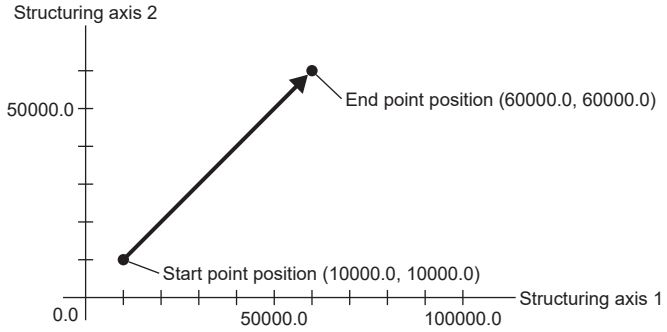
For details on object data settings, refer to the following.

📖 User's manual for the controller used

Program example

The following is a program example. The linear interpolation control start (bLinearInterpolateCMD) is set to TRUE, and Axes group 1 (AxesGroup001) is enabled. Then, 2-axis linear interpolation relative positioning control is performed for Axis group 1 (AxesGroup001).

- Operation



- Axes group

Item	Setting value
Structuring axis[1]	Axis0001
Structuring axis[2]	Axis0002

- Setting

Item	Setting value	
	Structuring axis 1	Structuring axis 2
Target position	50000.0	50000.0
Velocity	50000.0	
Acceleration	50000.0	
Deceleration	50000.0	
Jerk	0.0	

■Axes

Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1
2	Axis0002	AXIS_REF	Axis 2

■Axes group

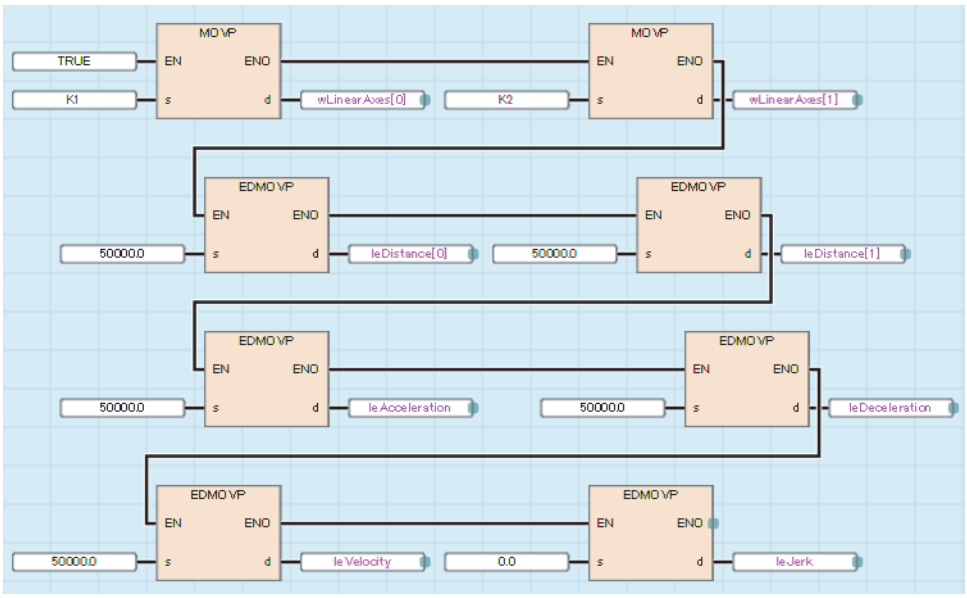
Axes group No.	Label name	Data type	Comment
1	AxesGroup001	AXES_GROUP_REF	Axes group 1

■Labels used

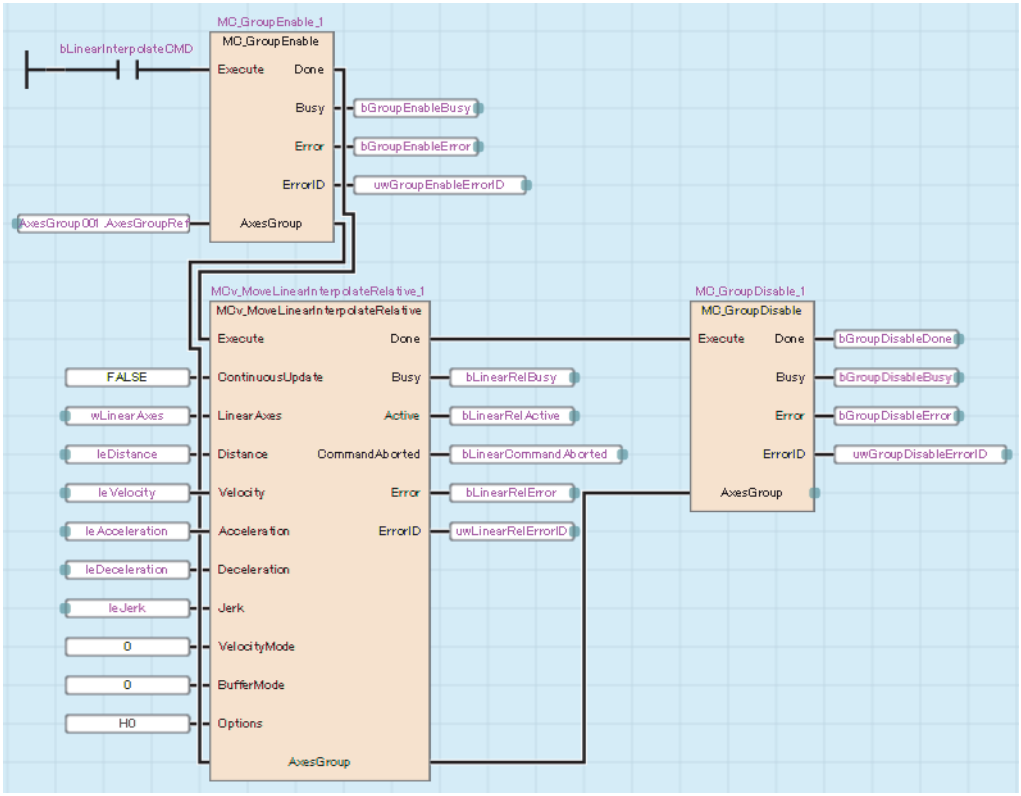
Label name	Data type	Comment
MCv_MoveLinearInterpolateRelative_1	MCv_MoveLinearInterpolateRelative	Relative Value Linear Interpolation Control FB
wLinearAxes	Word [signed](0..15)	Linear interpolation axis
leDistance	Double-precision real number(0..15)	Movement amount
leVelocity	Double-precision real number	Velocity
leAcceleration	Double-precision real number	Acceleration
leDeceleration	Double-precision real number	Deceleration
leJerk	Double-precision real number	Jerk
bLinearRelDone	Bit	Execution completion
bLinearRelBusy	Bit	Executing
bLinearRelActive	Bit	Controlling
bLinearCommandAborted	Bit	Abortion of execution
bLinearRelError	Bit	Error
uwLinearRelErrorID	Word [unsigned]/bit string [16-bit]	Error code
MC_GroupEnable_1	MC_GroupEnable	Axes Group Enabled FB
bLinearInterpolateCMD	Bit	Linear interpolation control start
bGroupEnableDone	Bit	Axes Group Enabled Completion
bGroupEnableBusy	Bit	Executing
bGroupEnableError	Bit	Error
uwGroupEnableErrorID	Word [unsigned]/bit string [16-bit]	Error code
MC_GroupDisable_1	MC_GroupDisable	Axes Group Disabled FB
bGroupDisableDone	Bit	Axes Group Disabled Completion
bGroupDisableBusy	Bit	Executing
bGroupDisableError	Bit	Error
uwGroupDisableErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data setting for 2-axis linear interpolation control



• Axes Group Enabled/2-axis linear interpolation control/Axis Group Disabled



■ST program

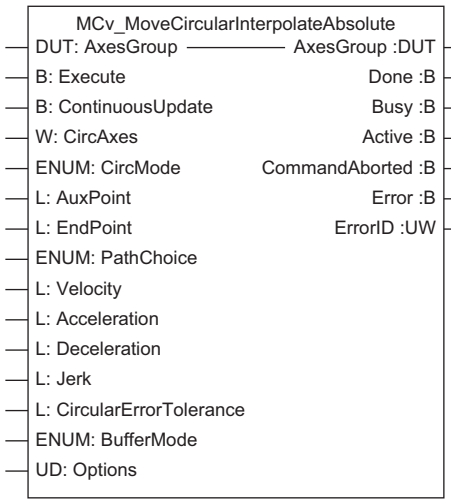
```
//-----Data setting for 2-axis linear interpolation control-----
wLinearAxes[0]:= 1;
wLinearAxes[1]:= 2;
leDistance[0]:= 50000.0;
leDistance[1]:= 50000.0;
leVelocity:= 50000.0;
leAcceleration:= 50000.0;
leDeceleration:= 50000.0;
leJerk:= 0.0;

//-----Axes Group Enabled-----
MC_GroupEnable_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bLinearInterpolateCMD ,
   Done=> bGroupEnableDone ,
   Busy=> bGroupEnableBusy ,
   Error=> bGroupEnableError ,
   ErrorID=> uwGroupEnableErrorID
);

//-----2-axis linear interpolation control-----
MCv_MoveLinearInterpolateRelative_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bGroupEnableDone ,
   ContinuousUpdate:= FALSE ,
   LinearAxes:= wLinearAxes ,
   Distance:= leDistance ,
   Velocity:= leVelocity ,
   Acceleration:= leAcceleration ,
   Deceleration:= leDeceleration ,
   Jerk:= leJerk ,
   VelocityMode:= MC_INTERPOLATE_SPEED_MODE__VectorSpeed ,
   BufferMode:= MC_BUFFER_MODE__mcAborting ,
   Options:= H00000000 ,
   Done=> bLinearRelDone ,
   Busy=> bLinearRelBusy ,
   Active=> bLinearRelActive ,
   CommandAborted=> bLinearRelCommandAborted ,
   Error=> bLinearRelError ,
   ErrorID=> uwLinearRelErrorID
);

//-----Axes Group Disabled-----
MC_GroupDisable_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bLinearRelDone ,
   Done=> bGroupDisableDone ,
   Busy=> bGroupDisableBusy ,
   Error=> bGroupDisableError ,
   ErrorID=> uwGroupDisableErrorID
);
```

42.13 Absolute Value Circular Interpolation Control

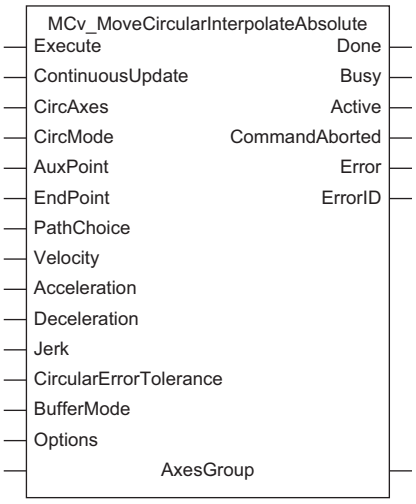
MCv_MoveCircularInterpolateAbsolute

This FB performs positioning with 2-axis circular interpolation using the structuring axes of the set axes group by setting the end point and sub point of the absolute position.

Ladder



FBD/LD



ST

```
MCv_MoveCircularInterpolateAbsolute(
  AxesGroup:= ?AXES_GROUP_REF? ,
  Execute:= ?BOOL? ,
  ContinuousUpdate:= ?BOOL? ,
  CircAxes:= ?INT(0..1)? ,
  CircMode:= ?INT? ,
  AuxPoint:= ?LREAL(0..15)? ,
  EndPoint:= ?LREAL(0..15)? ,
  PathChoice:= ?INT? ,
  Velocity:= ?LREAL? ,
  Acceleration:= ?LREAL? ,
  Deceleration:= ?LREAL? ,
  Jerk:= ?LREAL? ,
  CircularErrorTolerance:= ?LREAL? ,
  BufferMode:= ?INT? ,
  Options:= ?DWORD? ,
  Done=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
AxesGroup	Axes group information	AXES_GROUP_REF	At start	—	Omission not possible	An axes group is set. For the variable (AxesGroupName.AxesGroupRef.) used, refer to the following. ☞ Page 1423 AxesGroupName.AxesGroupRef. (Axes group information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Start	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) is executed.
ContinuousUpdate	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	Whether to enable or disable continuous change of Velocity (Velocity), Acceleration (Acceleration), and Deceleration (Deceleration) is set. Continuous update re-imports input variables to the instance in execution without aborting the operation. • FALSE: Disabled • TRUE: Enabled

Input variable	Name	Data type	Input	Valid range	Default value	Description
CircAxes	Circular interpolation axis	INT[0..1]	At start	1 to 16	0	The axes to be used in circular interpolation control are set from the structuring axes. Specify the index numbers (1 to 16) of the structuring axes in the array. The first element of the array is regarded as the reference axis. For details, refer to the following. ☞ Page 1723 Circular interpolation axis (CircAxes)
CircMode	Circular interpolation mode	INT (MC_CIRC_MODE)	At start	0 to 2	0	The circular interpolation specification method is set. • 0: Border point specification (mcBorder) • 1: Center point specification (mcCenter) • 2: Radius specification (mcRadius) For details, refer to the following. ☞ Page 1723 Circular interpolation mode (CircMode)
AuxPoint	Sub point	LREAL[0..15]	At start	■When "0: Border point specification (mcBorder)" or "1: Center point specification (mcCenter)" is set: -10000000000.0 to 10000000000.0 ■When "2: Radius specification (mcRadius)" is set: 0.000001 to 2147483647.0	0.0	The absolute position of the sub point (border point, center point, and radius) according to unit of the axis is set. The value is set as one-dimensional array data. The set value differs depending on the setting of Circular interpolation mode (CircMode). ■When "0: Border point specification (mcBorder)" or "1: Center point specification (mcCenter)" is set: The data is treated as the absolute position for structuring axes 1 to 16. The absolute positions of the border point and center point are set. ■When "2: Radius specification (mcRadius)" is set: The first element is treated as the radius and the second and subsequent elements are ignored. For details, refer to the following. ☞ Page 1724 Sub point (AuxPoint)
EndPoint	End point	LREAL[0..15]	At start	-10000000000.0 to 10000000000.0	0.0	The position of the end point is set. The value is set as one-dimensional array data. The data is treated as the absolute position for structuring axes 1 to 16. For details, refer to the following. ☞ Page 1724 End point (EndPoint)
PathChoice	Path selection	INT (MC_CIRC_PATHCHOICE)	At start	0 to 5	0	The rotation direction of the circular interpolation is set. • 0: CW(mcCW) • 1: CCW(mcCCW) • 2: Shortcut (mcShortWay) • 3: Detour (mcLongWay) • 4: CW Detour (mcCWLongWay) • 5: CCW Detour (mcCCWLongWay) *: When Circular interpolation mode (CircMode) is set to "0: Border point specification (mcBorder)", the input is ignored. *: When Circular interpolation mode (CircMode) is set to "2: Radius specification (mcRadius)", "0: CW (mcCW)" means the CW shortcut and "1: CCW (mcCCW)" means the CCW shortcut. For details, refer to the following. ☞ Page 1725 Path selection (PathChoice)
Velocity	Velocity	LREAL	At start/Retrigger possible/ Continuous update possible	0.0, 0.0001 to 2500000000.0	0.0	The speed command value is set. For details, refer to the following. ☞ Page 1727 Velocity (Velocity)
Acceleration	Acceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The acceleration is set. For details, refer to the following. ☞ Page 1727 Acceleration (Acceleration)

Input variable	Name	Data type	Input	Valid range	Default value	Description
Deceleration	Deceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration is set. For details, refer to the following. ☞ Page 1727 Deceleration (Deceleration)
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	The jerk is set. For details, refer to the following. ☞ Page 1727 Jerk (Jerk)
CircularErrorTolerance	Circular interpolation error tolerance	LREAL	At start	0.000001 to 100000.0	100.0	The allowable range of circular interpolation errors is set. For details, refer to the following. ☞ Page 1728 Circular interpolation error tolerance (CircularErrorTolerance)
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0 to 5	0	The buffer mode is set. • 0: Aborting(mcAborting) • 1: Buffered(mcBuffered) • 2: BlendingLow(mcBlendingLow) • 3: BlendingPrevious(mcBlendingPrevious) • 4: BlendingNext(mcBlendingNext) • 5: BlendingHigh(mcBlendingHigh) For details, refer to the following. ☞ Page 1728 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	00000000H to 00000001H	00000000H	The function options for MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) are set in the specified bit position. For details, refer to the following. ☞ Page 1729 Options (Options)

■Output variables

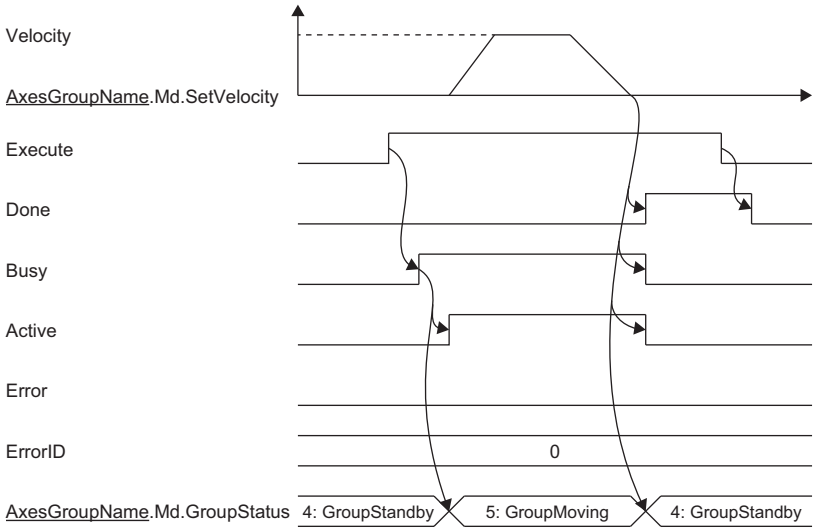
Output variable	Name	Data type	Default value	Description
Done	Completion	BOOL	FALSE	This variable becomes TRUE when the control is completed. The value differs depending on the status of Start (Execute) at operation completion. ■When Start (Execute) is TRUE This variable remains TRUE until Start (Execute) is set to FALSE. ■When Start (Execute) is FALSE This variable becomes TRUE only for one cycle.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) is controlling the axis. When multiple MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) FBs are executed for the same axes group, Controlling (Active) of only one MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) FB becomes TRUE.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) is aborted because of an error or multiple start.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

- This FB specifies an axes group and performs interpolation control mechanically so that the path from the start point (move start point) to the end point forms an arc for a linear axis which is orthogonally arranged.
- In circular interpolation control, interpolation control is performed using two arbitrary axes from the structuring axes set to the axes group.
- For circular interpolation control, the border point specification, center point specification, and radius specification circular interpolation methods are settable.
- In MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control), 2-axis circular interpolation control is performed by specifying the end point or sub point of the absolute position.

■Timing chart

• When the processing is completed successfully



• When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Circular interpolation axis (CircAxes)

In circular interpolation control, circular interpolation is performed by using two arbitrary axes from the structuring axes set to the axes group.

The axes used to perform circular interpolation are specified from the structuring axes in the axes group by using Circular interpolation axis (CircAxes). The Circular interpolation axis (CircAxes) has two array elements. For the arrays, specify the index numbers (1 to 16: structuring axes 1 to 16) of structuring axes to be used for interpolation control.

The first element of the array is regarded as the reference axis.

If the configuration axis specified does not have an axis set, Circular Interpolation Axis Unset (error code: 1ACAH) occurs.

Ex.

When setting configuration axes 2 and 3 as the circular interpolation axes

CircAxes[0]:= 2;

CircAxes[1]:= 3;

■Circular interpolation mode (CircMode)

The circular interpolation method (border point specification, center point specification, or radius specification) used to perform circular interpolation control is set.

The contents set in Sub point (AuxPoint) differ depending on the setting of Circular interpolation mode (CircMode).

Setting value	Sub point (AuxPoint)
0: Border point specification (mcBorder)	Sub point (AuxPoint) sets the border point on the arc connecting the start point and the end point. Positioning control with arc path that passes through the specified border point is performed using the two structuring axes set in Circular interpolation axis (CircAxes).
1: Center point specification (mcCenter)	Sub point (AuxPoint) sets the center point of the arc. Positioning control with an arc path having the center point at its center is performed using the two structuring axes set in Circular interpolation axis (CircAxes).
2: Radius specification (mcRadius)	Sub point (AuxPoint) sets the radius of the arc. Positioning control with an arc path having the specified radius is performed using the two structuring axes set in Circular interpolation axis (CircAxes).

For the operation of each circular interpolation control set in Circular interpolation mode (CircMode), refer to the following.

📖 User's manual for the controller used

■Sub point (AuxPoint)

The position of the sub point (border point, center point, or radius) used to perform circular interpolation is set.

Sub point (AuxPoint) has 16 array elements. The contents set in Sub point (AuxPoint) differ depending on the setting of Circular interpolation mode (CircMode).

Circular interpolation mode (CircMode)	Setting range	Sub point (AuxPoint) setting
0: Border point specification (mcBorder)	-10000000000.0 ≤ Set value < 10000000000.0*1	Sub point (AuxPoint) sets the border point on the arc connecting the start point and the end point. AuxPoint[0..15] indicates the sub point positions of structuring axes 1 to 16. • Sub point (AuxPoint) of structuring axes not set to the circular interpolation axes is ignored. • Sub point (AuxPoint) of axes after all structuring axes set to the circular interpolation axes can be omitted.
1: Center point specification (mcCenter)		Sub point (AuxPoint) sets the center point of the arc. AuxPoint[0..15] indicates the sub point positions of structuring axes 1 to 16. • Sub point (AuxPoint) of structuring axes not set to the circular interpolation axes is ignored. • Sub point (AuxPoint) of axes after all structuring axes set to the circular interpolation axes can be omitted.
2: Radius specification (mcRadius)	0.000001 to 2147483647.0	Sub point (AuxPoint) sets the radius of the arc. AuxPoint[0] indicates the radius of the circle. • Inputs after AuxPoint[1] are ignored and can be omitted.

*1 When the ring counter is enabled, the ring counter range is used.

Ex.

When setting structuring axis 2 and 3 to the circular interpolation axes and setting the absolute position of the sub point for structuring axis 2 and 3 as 2000.0 and 3000.0

CircAxes[0]:= 2;

CircAxes[1]:= 3;

AuxPoint[0]:= 0.0;

AuxPoint[1]:= 2000.0;

AuxPoint[2]:= 3000.0;

AuxPoint[3]:= 0.0;*2

⋮

AuxPoint[15]:= 0.0;*2

*2 "AuxPoint[3]:= 0.0; to AuxPoint[15]:= 0.0;" can be omitted.

■End point (EndPoint)

The end point of the absolute position is set.

End point (EndPoint) has 16 array elements. EndPoint[0..15] indicates the end point positions of structuring axes 1 to 16.

- End point (EndPoint) for structuring axes which are not set to the circular interpolation axis is ignored.
- End point (EndPoint) after all of the structuring axes which are set to the circular interpolation axis can be omitted.

Set End point (EndPoint) within the following range.

Setting range
-10000000000.0 ≤ Set value < 10000000000.0*1

*1 When the ring counter is enabled, the ring counter range is used.

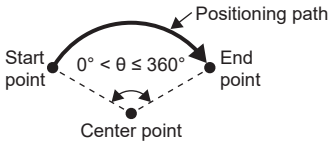
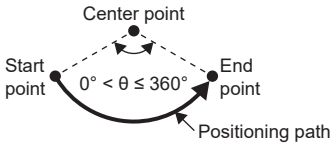
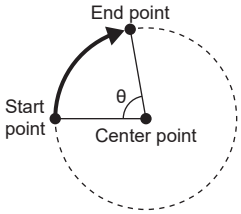
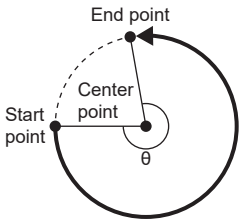
■Path selection (PathChoice)

The rotation direction for circular interpolation is set. For the rotation direction, the first element of the array set in Circular interpolation axis (CircAxes) is regarded as the reference axis.

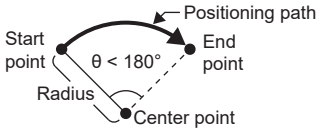
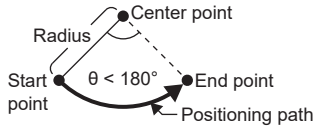
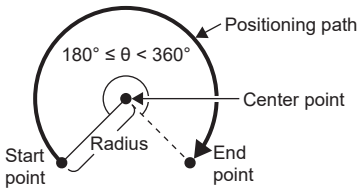
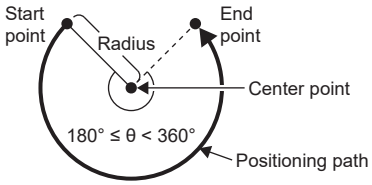
Path selection (PathChoice) is set when Circular interpolation mode (CircMode) is set to "1: Center point specification (mcCenter)" or "2: Radius specification (mcRadius)". When "0: Border point specification (mcBorder)" is set, this input is omitted.

The following shows the setting value, setting description, arc center angle that can be controlled, and path of Path selection (PathChoice) according to the setting of Circular interpolation mode (CircMode).

• Path for "1: Center point specification (mcCenter)"

Setting value	Setting description	Arc center angle that can be controlled	Path
0: CW(mcCW)	CW	$0° < \theta \leq 360°$	
1: CCW (mcCCW)	CCW	$0° < \theta \leq 360°$	
2: Shortcut (mcShortWay)	Shortcut • When the start point, center point, and end point are on one straight line, becomes CW. (A semicircle is drawn in the CW direction.) • If the start point is the same as the end point, Path Choice Setting Incorrect (error code: 1B06H) occurs.	$0° < \theta \leq 180°$	
3: Detour (mcLongWay)	Detour • When the start point, center point, and end point are on one straight line, becomes CW. (A semicircle is drawn in the CW direction.) • When the start point is the same as the end point, the direction is CW. (A perfect circle is drawn in CW direction.)	$180° \leq \theta \leq 360°$	

• Path for "2: Radius specification (mcRadius)"

Setting value	Setting description	Arc center angle that can be controlled	Path
0: CW(mcCW)	CW (CW shortcut) • If the start point, center point, and end point are on one straight line, Path Choice Setting Incorrect (error code: 1B06H) occurs.	0° < θ < 180°	
1: CCW(mcCCW)	CCW (CCW shortcut) • If the start point, center point, and end point are on one straight line, Path Choice Setting Incorrect (error code: 1B06H) occurs.	0° < θ < 180°	
4: CW Detour (mcCWLongWay)	CW detour • When the start point, end point, and center point are on one straight line, a semicircle is drawn.	180° ≤ θ < 360°	
5: CCW Detour (mcCCWLongWay)	CCW detour • When the start point, end point, and center point are on one straight line, a semicircle is drawn.	180° ≤ θ < 360°	

■Velocity (Velocity)

The maximum speed of the path used in MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) is set.

Setting range
0.0, 0.0001 to 2500000000.0[*1*2]

*1 To perform floating-point operation, the lower limit value of the command velocity is limited by the following restrictions.
If the velocity converted from the command velocity to operation cycle is less than 0.00001, Out of Operation Cycle Converted Velocity Range (error code: 1B16H) (when changing the velocity, Out of Operation Cycle Converted Velocity Range Warning (event code: 00D2FH)) occurs. To improve the precision of floating-point operation, change Position command unit (AxisName.Pr.Unit_Position) and Speed command unit (AxisName.Pr.Unit_Velocity) and perform settings so that the velocity converted into the operation cycle is not less than "0.00001".

*2 When a motion control FB with a commanded velocity of "0.0" performs a multiple start, the commanded velocity of the previous motion control FB is used.

■Acceleration (Acceleration)

The acceleration used in MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) is set.
The setting range differs depending on the setting for Acceleration/deceleration method setting (Options (Options): Bit 0 to 2).

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	0.000000[*1], positive numbers from 0.000001 to 8400.0 [s]

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Deceleration (Deceleration)

The deceleration used in MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) is set.
Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Deceleration (Deceleration) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Jerk (Jerk)

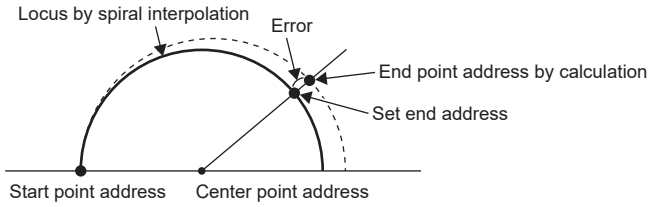
The jerk used in MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) is set.
Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Jerk (Jerk) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000, positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

■Circular interpolation error tolerance (CircularErrorTolerance)

In circular interpolation control with center point specification, the arc path calculated from the start position and center position may deviate from the end position set in End point (EndPoint).

Circular interpolation error tolerance (CircularErrorTolerance) sets the allowable width of errors between the calculated arc path and the end position.

Setting range	Description
Calculated error ≤ Circular interpolation error tolerance (CircularErrorTolerance)	Circular interpolation control to the set end point address is performed while the error compensation is carried out by spiral interpolation. 
Calculated error > Circular interpolation error tolerance (CircularErrorTolerance)	At the positioning start, Circular interpolation error allowable value over (error code: 1AD9H) occurs and the control will not start. The machine will immediately stop if the error is detected during positioning control.

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following buffer modes are settable in MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control).

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.
1: Buffered(mcBuffered)	Buffers the next FB on the under-control FB. If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after completion of the under-control FB.
2: BlendingLow(mcBlendingLow)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the lower target velocity between the under-control FB and the buffering FB is used as the switching speed.
3: BlendingPrevious(mcBlendingPrevious)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the under-control FB is used as the switching speed.
4: BlendingNext(mcBlendingNext)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the buffering FB is used as the switching speed.
5: BlendingHigh(mcBlendingHigh)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the higher target velocity between the under-control FB and the buffering FB is used as the switching speed.

*1 Stops are not performed between the under-control FB and the buffering FB.

Point

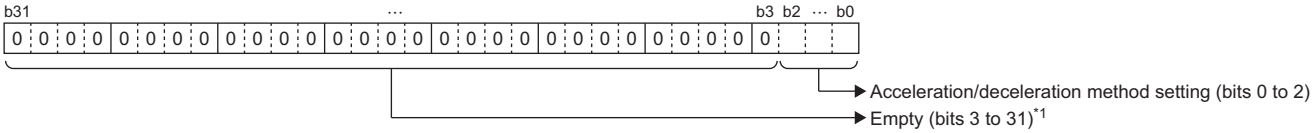
For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

■Options (Options)

The function options for MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
0 to 2	Acceleration/deceleration method setting	The acceleration/deceleration method used to perform control is set. • 0: Acceleration/deceleration specification method (mcAccDec) • 1: Acceleration/deceleration time-fixed method (mcFixedTime)

• Acceleration/deceleration method setting (bit 0 to 2)

Setting value	Description
0: Acceleration/deceleration specification method (mcAccDec)	An acceleration/deceleration method using Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk) set in MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control).
1: Acceleration/deceleration time-fixed method (mcFixedTime)	An acceleration/deceleration method using the acceleration/deceleration time specified in MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) regardless of the velocity. The acceleration/deceleration time does not use Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk).

■Necessary object data

When using MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control), set the following object data for all structuring axes in the specified axes group.

• Target position (607AH)

If there is a structuring axis for which no object data is set, Necessary Object Data Unset (error code: 1AF7H) occurs and the axis will not start.

For details on object data settings, refer to the following.

📖 User's manual for the controller used

Precautions

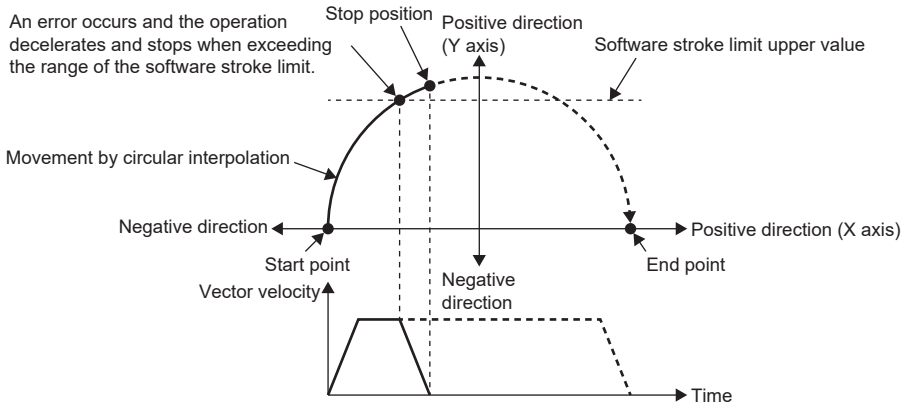
- Circular interpolation including an axis whose stroke limit is set to invalid cannot be performed. Software stroke limit invalid at circular interpolation (error code: 1ADAH) occurs and the operation does not start.
- When the interpolation path exceeds the stroke limit range during interpolation operation, Software stroke limit over (positive direction) (error code: 1A83H) or Software stroke limit over (negative direction) (error code: 1A84H) occurs and the operation will stop.

Ex.

When the upper limit of the software stroke limit is exceeded in the positive direction of the Y-axis

For an immediate stop, stops at the same time that the error occurs.

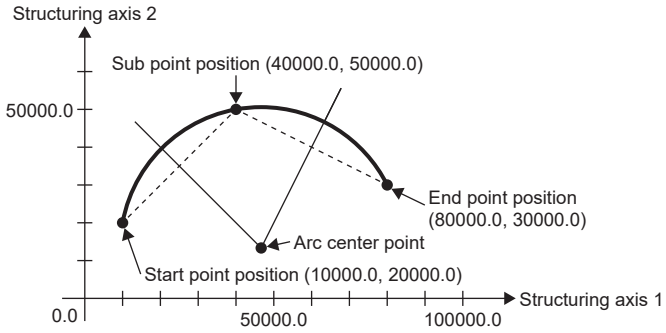
For a deceleration stop, decelerates to a stop along the arc of the circle as shown below.



Program example

The following is a program example. The circular interpolation control start (bCircularInterpolateCMD) is set to TRUE, and Axes group 1 (AxesGroup001) is enabled. Then, 2-axis circular interpolation (sub point specification) absolute positioning control is performed for Axis group 1 (AxesGroup001).

• Operation



• Axes group

Item	Setting value
Structuring axis[1]	Axis0001
Structuring axis[2]	Axis0002

• Setting

Item	Setting value	
	Structuring axis 1	Structuring axis 2
Sub point	40000.0	50000.0
End point	80000.0	30000.0
Velocity	50000.0	
Acceleration	50000.0	
Deceleration	50000.0	
Jerk	0.0	
Circular interpolation error tolerance	100.0	

■Axes

Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1
2	Axis0002	AXIS_REF	Axis 2

■Axes group

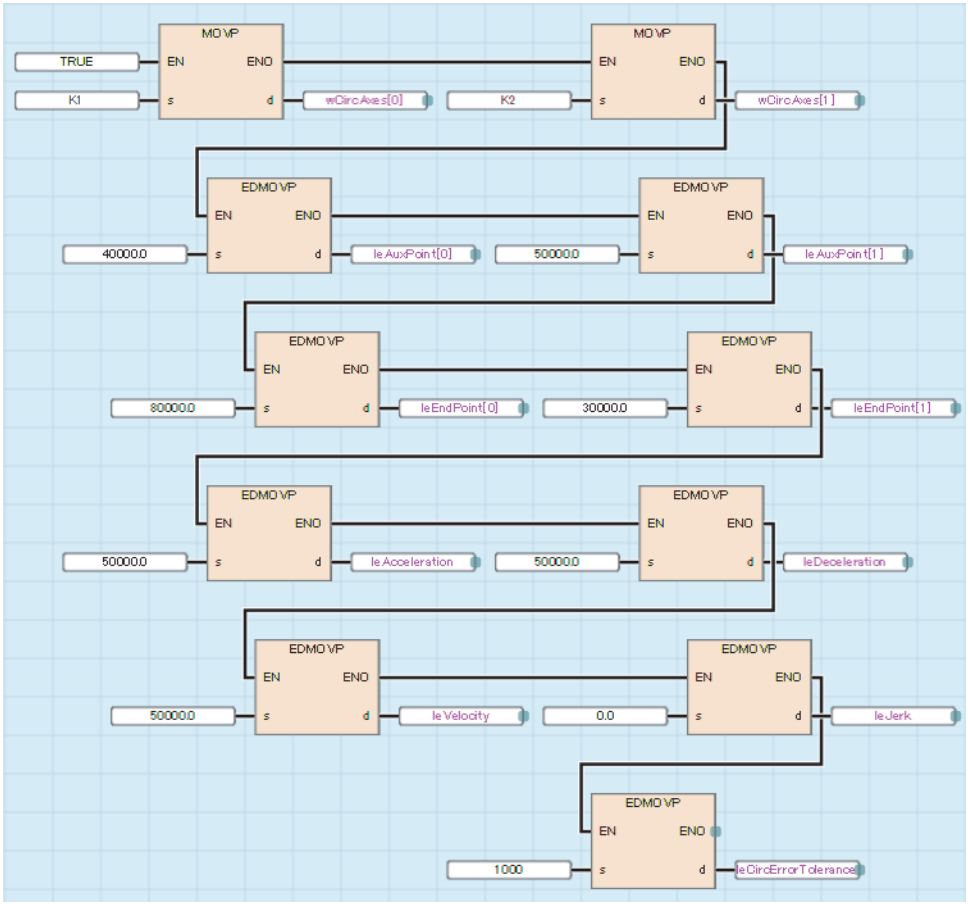
Axes group No.	Label name	Data type	Comment
1	AxesGroup001	AXES_GROUP_REF	Axes group 1

■Labels used

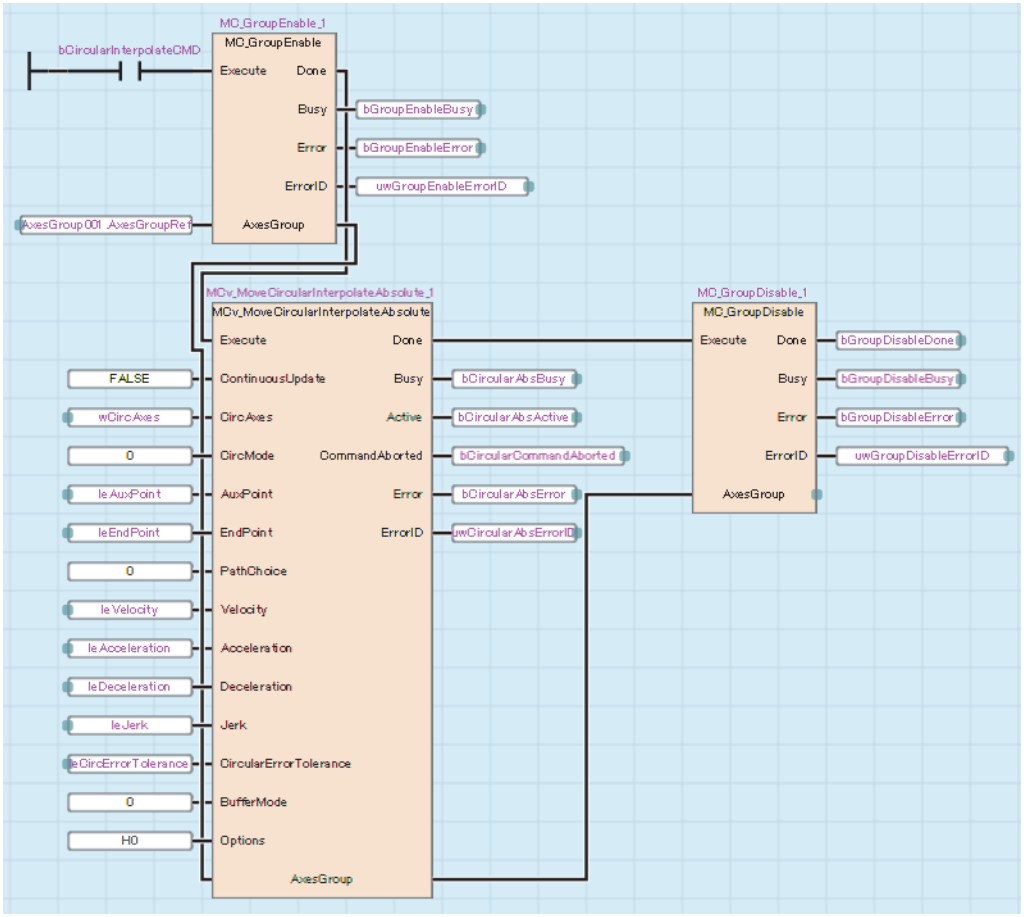
Label name	Data type	Comment
MCv_MoveCircularInterpolateAbsolute_1	MCv_MoveCircularInterpolateAbsolute	Absolute Value Circular Interpolation Control FB
wCircAxes	Word [signed] (0..1)	Circular interpolation axis
leAuxPoint	Double-precision real number(0..15)	Sub point
leEndPoint	Double-precision real number(0..15)	End point
leVelocity	Double-precision real number	Velocity
leAcceleration	Double-precision real number	Acceleration
leDeceleration	Double-precision real number	Deceleration
leJerk	Double-precision real number	Jerk
leCircErrorTolerance	Double-precision real number	Circular interpolation error tolerance
bCircularAbsDone	Bit	Execution completion
bCircularAbsBusy	Bit	Executing
bCircularAbsActive	Bit	Controlling
bCircularCommandAborted	Bit	Abortion of execution
bCircularAbsError	Bit	Error
uwCircularAbsErrorID	Word [unsigned]/bit string [16-bit]	Error code
MC_GroupEnable_1	MC_GroupEnable	Axes Group Enabled FB
bCircularInterpolateCMD	Bit	Circular interpolation control start
bGroupEnableDone	Bit	Axes Group Enabled Completion
bGroupEnableBusy	Bit	Executing
bGroupEnableError	Bit	Error
uwGroupEnableErrorID	Word [unsigned]/bit string [16-bit]	Error code
MC_GroupDisable_1	MC_GroupDisable	Axes Group Disabled FB
bGroupDisableDone	Bit	Axes Group Disabled Completion
bGroupDisableBusy	Bit	Executing
bGroupDisableError	Bit	Error
uwGroupDisableErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data setting for 2-axis circular interpolation control



- Axes Group Enabled/Circular interpolation control (Border point specification)/Axis Group Disabled



■ST program

```
//-----Data setting for 2-axis circular interpolation control-----
wCircAxes[0]:= 1;
wCircAxes[1]:= 2;
leAuxPoint[0]:= 40000.0;
leAuxPoint[1]:= 50000.0;
leEndPoint[0]:= 80000.0;
leEndPoint[1]:= 30000.0;
leVelocity:= 50000.0;
leAcceleration:= 50000.0;
leDeceleration:= 50000.0;
leJerk:= 0.0;
leCircularErrorTolerance:= 100.0;

//-----Axes Group Enabled-----
MC_GroupEnable_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bCircularInterpolateCMD ,
   Done=> bGroupEnableDone ,
   Busy=> bGroupEnableBusy ,
   Error=> bGroupEnableError ,
   ErrorID=> uwGroupEnableErrorID
);

//-----Circular interpolation control (Border point specification)-----
MCv_MoveCircularInterpolateAbsolute_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bGroupEnableDone ,
   ContinuousUpdate:= FALSE ,
   CircAxes:= wCircAxes ,
   CircMode:= MC_CIRC_MODE__mcBorder ,
   AuxPoint:= leAuxPoint ,
   EndPoint:= leEndPoint ,
   PathChoice:= MC_CIRC_PATHCHOICE__mcCW ,
   Velocity:= leVelocity ,
   Acceleration:= leAcceleration ,
   Deceleration:= leDeceleration ,
   Jerk:= leJerk ,
   CircularErrorTolerance:= leCircularErrorTolerance ,
   BufferMode:= MC_BUFFER_MODE__mcAborting ,
   Options:= H00000000 ,
   Done=> bCircularAbsDone ,
   Busy=> bCircularAbsBusy ,
   Active=> bCircularAbsActive ,
   CommandAborted=> bCircularAbsCommandAborted ,
   Error=> bCircuAbsError ,
   ErrorID=> uwCircularAbsErrorID
);
```

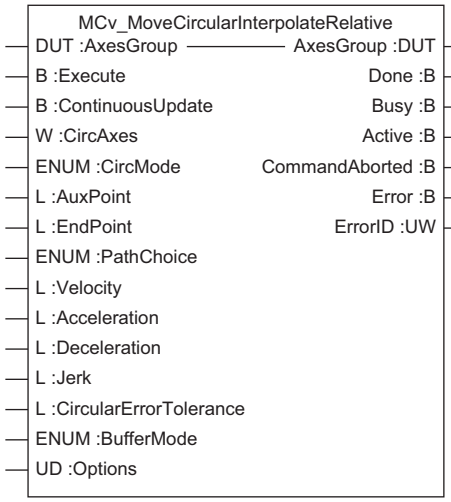
```
//-----Axes Group Disabled-----
MC_GroupDisable_1(
    AxesGroup:= AxesGroup001.AxesGroupRef ,
    Execute:= bCircularAbsDone ,
    Done=> bGroupDisableDone ,
    Busy=> bGroupDisableBusy ,
    Error=> bGroupDisableError ,
    ErrorID=> uwGroupDisableErrorID
);
```

42.14 Relative Value Circular Interpolation Control

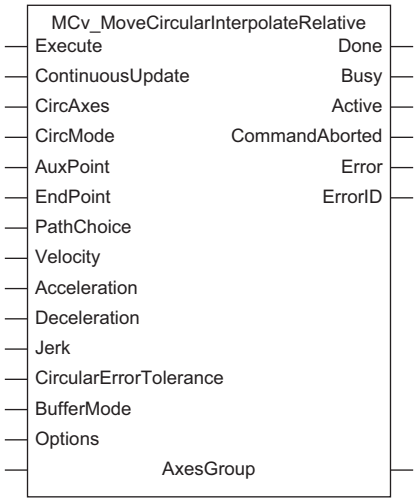
MCv_MoveCircularInterpolateRelative

This FB performs positioning with 2-axis circular interpolation using the structuring axes of the set axes group by setting the relative position from the current position at start to the end point and sub point.

Ladder



FBD/LD



ST

```
MCv_MoveCircularInterpolateRelative(
  AxesGroup:= ?AXES_GROUP_REF? ,
  Execute:= ?BOOL? ,
  ContinuousUpdate:= ?BOOL? ,
  CircAxes:= ?INT(0..1)? ,
  CircMode:= ?INT? ,
  AuxPoint:= ?LREAL(0..15)? ,
  EndPoint:= ?LREAL(0..15)? ,
  PathChoice:= ?INT? ,
  Velocity:= ?LREAL? ,
  Acceleration:= ?LREAL? ,
  Deceleration:= ?LREAL? ,
  Jerk:= ?LREAL? ,
  CircularErrorTolerance:= ?LREAL? ,
  BufferMode:= ?INT? ,
  Options:= ?DWORD? ,
  Done=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
AxesGroup	Axes group information	AXES_GROUP_REF	At start	—	Omission not possible	An axes group is set. For the variable (AxesGroupName.AxesGroupRef.) used, refer to the following. ☞ Page 1423 AxesGroupName.AxesGroupRef. (Axes group information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Start	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) is executed.
Continuous Update	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	Whether to enable or disable continuous change of Velocity (Velocity), Acceleration (Acceleration), and Deceleration (Deceleration) is set. Continuous update re-imports input variables to the instance in execution without aborting the operation. • FALSE: Disabled • TRUE: Enabled

Input variable	Name	Data type	Input	Valid range	Default value	Description
CircAxes	Circular interpolation axis	INT[0..1]	At start	1 to 16	0	The axes to be used in circular interpolation control are set from the structuring axes. Specify the index numbers (1 to 16) of the structuring axes in the array. The first element of the array is regarded as the reference axis. For details, refer to the following. ☞ Page 1739 Circular interpolation axis (CircAxes)
CircMode	Circular interpolation mode	INT (MC_CIRC_MODE)	At start	0 to 2	0	The circular interpolation specification method is set. • 0: Border point specification (mcBorder) • 1: Center point specification (mcCenter) • 2: Radius specification (mcRadius) For details, refer to the following. ☞ Page 1740 Circular interpolation mode (CircMode)
AuxPoint	Sub point	LREAL[0..15]	At start	■When "0: Border point specification (mcBorder)" or "1: Center point specification (mcCenter)" is set: -10000000000.0 to 10000000000.0 ■When "2: Radius specification (mcRadius)" is set: 0.000001 to 2147483647.0	0.0	The relative position of the sub point (border point, center point, and radius) from the start position is set. The value is set as one-dimensional array data. The set value differs depending on the setting of Circular interpolation mode (CircMode). ■When "0: Border point specification (mcBorder)" or "1: Center point specification (mcCenter)" is set: The data is treated as the relative position for structuring axes 1 to 16. The absolute positions of the border point and center point are set. ■When "2: Radius specification (mcRadius)" is set: The first element is treated as the radius and the second and subsequent elements are ignored. For details, refer to the following. ☞ Page 1740 Sub point (AuxPoint)
EndPoint	End point	LREAL[0..15]	At start	-10000000000.0 to 10000000000.0	0.0	The movement amount from the current position at start to the end point is set. The value is set as one-dimensional array data. The data is treated as the absolute position for structuring axes 1 to 16. For details, refer to the following. ☞ Page 1740 End point (EndPoint)
PathChoice	Path selection	INT (MC_CIRC_PATHCHOICE)	At start	0 to 5	0	The rotation direction of the circular interpolation is set. • 0: CW(mcCW) • 1: CCW(mcCCW) • 2: Shortcut (mcShortWay) • 3: Detour (mcLongWay) • 4: CW Detour (mcCWLongWay) • 5: CCW Detour (mcCCWLongWay) *: When Circular interpolation mode (CircMode) is set to "0: Border point specification (mcBorder)", the input is ignored. *: When Circular interpolation mode (CircMode) is set to "2: Radius specification (mcRadius)", "0: CW (mcCW)" means the CW shortcut and "1: CCW (mcCCW)" means the CCW shortcut. For details, refer to the following. ☞ Page 1741 Path selection (PathChoice)

Input variable	Name	Data type	Input	Valid range	Default value	Description
Velocity	Velocity	LREAL	At start/Retrigger possible/ Continuous update possible	0.0, 0.0001 to 2500000000.0	0.0	The speed command value is set. For details, refer to the following. ☞ Page 1743 Velocity (Velocity)
Acceleration	Acceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The acceleration is set. For details, refer to the following. ☞ Page 1743 Acceleration (Acceleration)
Deceleration	Deceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration is set. For details, refer to the following. ☞ Page 1743 Deceleration (Deceleration)
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	The jerk is set. For details, refer to the following. ☞ Page 1743 Jerk (Jerk)
CircularError Tolerance	Circular interpolation error tolerance	LREAL	At start	0.000001 to 100000.0	100.0	The allowable range of circular interpolation errors is set. For details, refer to the following. ☞ Page 1744 Circular interpolation error tolerance (CircularErrorTolerance)
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0 to 5	0	The buffer mode is set. • 0: Aborting(mcAborting) • 1: Buffered(mcBuffered) • 2: BlendingLow(mcBlendingLow) • 3: BlendingPrevious(mcBlendingPrevious) • 4: BlendingNext(mcBlendingNext) • 5: BlendingHigh(mcBlendingHigh) For details, refer to the following. ☞ Page 1744 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	00000000H to 00000005H	00000000H	The function options for MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) are set in the specified bit position. For details, refer to the following. ☞ Page 1745 Options (Options)

■Output variables

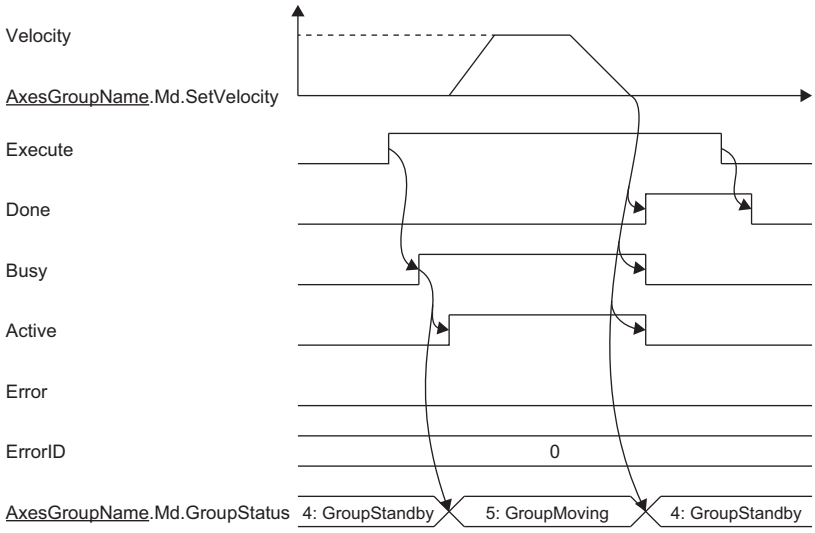
Output variable	Name	Data type	Default value	Description
Done	Completion	BOOL	FALSE	This variable becomes TRUE when the control is completed. The value differs depending on the status of Start (Execute) at operation completion. ■When Start (Execute) is TRUE This variable remains TRUE until Start (Execute) is set to FALSE. ■When Start (Execute) is FALSE This variable becomes TRUE only for one cycle.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) is controlling the axis. When multiple MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) FBs are executed for the same axes group, Controlling (Active) of only one MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) FB becomes TRUE.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) is aborted because of an error or multiple start.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

- This FB specifies an axes group and performs interpolation control mechanically so that the path from the start point (move start point) to the end point forms an arc for a linear axis which is orthogonally arranged.
- In circular interpolation control, interpolation control is performed using two arbitrary axes from the structuring axes set to the axes group.
- For circular interpolation control, the border point specification, center point specification, and radius specification circular interpolation methods are settable.
- In MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control), 2-axis circular interpolation control is performed by specifying the movement amount from the current position to the end point or sub point.

■Timing chart

- When the processing is completed successfully



- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Circular interpolation axis (CircAxes)

In circular interpolation control, circular interpolation is performed by using two arbitrary axes from the structuring axes set to the axes group.

The axes used to perform circular interpolation are specified from the structuring axes in the axes group by using Circular interpolation axis (CircAxes). The Circular interpolation axis (CircAxes) has two array elements. For the arrays, specify the index numbers (1 to 16: structuring axes 1 to 16) of structuring axes to be used for interpolation control.

The first element of the array is regarded as the reference axis.

If the configuration axis specified does not have an axis set, Circular Interpolation Axis Unset (error code: 1ACAH) occurs.

Ex.

When setting configuration axes 2 and 3 as the circular interpolation axes

CircAxes[0]:= 2;

CircAxes[1]:= 3;

■Circular interpolation mode (CircMode)

The circular interpolation method (border point specification, center point specification, or radius specification) used to perform circular interpolation control is set.

The contents set in Sub point (AuxPoint) differ depending on the setting of Circular interpolation mode (CircMode).

Setting value	Sub point (AuxPoint)
0: Border point specification (mcBorder)	Sub point (AuxPoint) sets the border point on the arc connecting the start point and the end point. Positioning control with arc path that passes through the specified border point is performed using the two structuring axes set in Circular interpolation axis (CircAxes).
1: Center point specification (mcCenter)	Sub point (AuxPoint) sets the center point of the arc. Positioning control with an arc path having the center point at its center is performed using the two structuring axes set in Circular interpolation axis (CircAxes).
2: Radius specification (mcRadius)	Sub point (AuxPoint) sets the radius of the arc. Positioning control with an arc path having the specified radius is performed using the two structuring axes set in Circular interpolation axis (CircAxes).

For the operation of each circular interpolation control set in Circular interpolation mode (CircMode), refer to the following.

User's manual for the controller used

■Sub point (AuxPoint)

The position of the sub point (border point, center point, or radius) used to perform circular interpolation is set.

Sub point (AuxPoint) has 16 array elements. The contents set in Sub point (AuxPoint) differ depending on the setting of Circular interpolation mode (CircMode).

Circular interpolation mode (CircMode)	Setting range	Sub point (AuxPoint) setting
0: Border point specification (mcBorder)	-10000000000.0 to 10000000000.0	Sub point (AuxPoint) sets the border point on the arc connecting the start point and the end point. AuxPoint[0..15] indicates the sub point positions of structuring axes 1 to 16. • Sub point (AuxPoint) of structuring axes not set to the circular interpolation axes is ignored. • Sub point (AuxPoint) of axes after all structuring axes set to the circular interpolation axes can be omitted.
1: Center point specification (mcCenter)		Sub point (AuxPoint) sets the center point of the arc. AuxPoint[0..15] indicates the sub point positions of structuring axes 1 to 16. • Sub point (AuxPoint) of structuring axes not set to the circular interpolation axes is ignored. • Sub point (AuxPoint) of axes after all structuring axes set to the circular interpolation axes can be omitted.
2: Radius specification (mcRadius)	0.000001 to 2147483647.0	Sub point (AuxPoint) sets the radius of the arc. AuxPoint[0] indicates the radius of the circle. • Inputs after AuxPoint[1] are ignored and can be omitted.

Ex.

When setting structuring axis 2 and 3 to the circular interpolation axes and setting the relative position of the sub point for structuring axis 2 and 3 as 2000.0 and 3000.0

CircAxes[0]:= 2;

CircAxes[1]:= 3;

AuxPoint[0]:= 0.0;

AuxPoint[1]:= 2000.0;

AuxPoint[2]:= 3000.0;

AuxPoint[3]:= 0.0;*1

⋮

AuxPoint[15]:= 0.0;*1

*1 "AuxPoint[3]:= 0.0; to AuxPoint[15]:= 0.0;" can be omitted.

■End point (EndPoint)

The movement amount from the current position at start to the end point is set.

End point (EndPoint) has 16 array elements. EndPoint[0..15] indicates the end point positions of structuring axes 1 to 16.

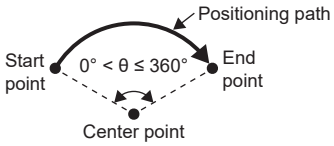
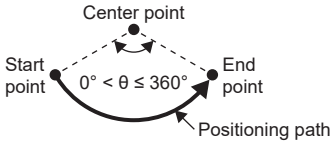
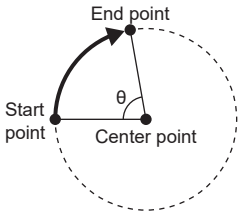
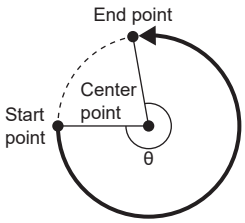
- End point (EndPoint) for structuring axes which are not set to the circular interpolation axis is ignored.
- End point (EndPoint) after all of the structuring axes which are set to the circular interpolation axis can be omitted.

■Path selection (PathChoice)

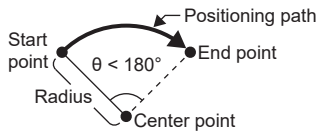
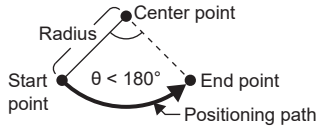
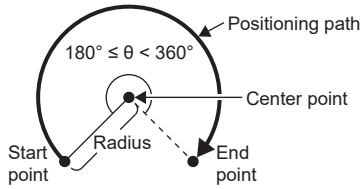
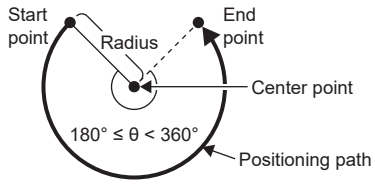
The rotation direction for circular interpolation is set. For the rotation direction, the first element of the array set in Circular interpolation axis (CircAxes) is regarded as the reference axis. Path selection (PathChoice) is set when Circular interpolation mode (CircMode) is set to "1: Center point specification (mcCenter)" or "2: Radius specification (mcRadius)". When "0: Border point specification (mcBorder)" is set, this input is omitted.

The following shows the setting value, setting description, arc center angle that can be controlled, and path of Path selection (PathChoice) according to the setting of Circular interpolation mode (CircMode).

• Path for "1: Center point specification (mcCenter)"

Setting value	Setting description	Arc center angle that can be controlled	Path
0: CW(mcCW)	CW	$0° < \theta \leq 360°$	
1: CCW (mcCCW)	CCW	$0° < \theta \leq 360°$	
2: Shortcut (mcShortWay)	Shortcut • When the start point, center point, and end point are on one straight line, becomes CW. (A semicircle is drawn in the CW direction.) • If the start point is the same as the end point, Path Choice Setting Incorrect (error code: 1B06H) occurs.	$0° < \theta \leq 180°$	
3: Detour (mcLongWay)	Detour • When the start point, center point, and end point are on one straight line, becomes CW. (A semicircle is drawn in the CW direction.) • When the start point is the same as the end point, the direction is CW. (A perfect circle is drawn in CW direction.)	$180° \leq \theta \leq 360°$	

• Path for "2: Radius specification (mcRadius)"

Setting value	Setting description	Arc center angle that can be controlled	Path
0: CW(mcCW)	CW • If the start point, center point, and end point are on one straight line, Path Choice Setting Incorrect (error code: 1B06H) occurs.	0° < θ < 180°	
1: CCW (mcCCW)	CCW • If the start point, center point, and end point are on one straight line, Path Choice Setting Incorrect (error code: 1B06H) occurs.	0° < θ < 180°	
4: CW Detour (mcCWLongWay)	CW detour • When the start point, end point, and center point are on one straight line, a semicircle is drawn.	180° ≤ θ < 360°	
5: CCW Detour (mcCCWLongWay)	CCW detour • When the start point, end point, and center point are on one straight line, a semicircle is drawn.	180° ≤ θ < 360°	

■Velocity (Velocity)

The maximum speed of the path used in MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) is set.

Setting range
0.0, 0.0001 to 2500000000.0[*1*2]

*1 To perform floating-point operation, the lower limit value of the command velocity is limited by the following restrictions.
If the velocity converted from the command velocity to operation cycle is less than 0.00001, Out of Operation Cycle Converted Velocity Range (error code: 1B16H) (when changing the velocity, Out of Operation Cycle Converted Velocity Range Warning (event code: 00D2FH)) occurs. To improve the precision of floating-point operation, change Position command unit (AxisName.Pr.Unit_Position) and Speed command unit (AxisName.Pr.Unit_Velocity) and perform settings so that the velocity converted into the operation cycle is not less than "0.00001".

*2 When a motion control FB with a commanded velocity of "0.0" performs a multiple start, the commanded velocity of the previous motion control FB is used.

■Acceleration (Acceleration)

The acceleration used in MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) is set.
The setting range differs depending on the setting for Acceleration/deceleration method setting (Options (Options): Bit 0 to 2).

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	0.000000[*1], positive numbers from 0.000001 to 8400.0 [s]

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Deceleration (Deceleration)

The deceleration used in MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) is set.
Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Deceleration (Deceleration) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Jerk (Jerk)

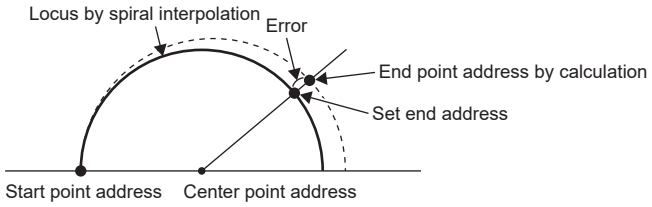
The jerk used in MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) is set.
Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 2) is set to "0: Acceleration/deceleration specification method (mcAccDec)". Jerk (Jerk) is not used if the acceleration/deceleration method is set to "1: Acceleration/deceleration time-fixed method (mcFixedTime)".

Acceleration/deceleration method setting (Options (Options): Bit 0 to 2)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000, positive numbers from 0.0001 to 2147483647.0 [U/s^2]
1: Acceleration/deceleration time-fixed method (mcFixedTime)	Not used

■Circular interpolation error tolerance (CircularErrorTolerance)

In circular interpolation control with center point specification, the arc path calculated from the start position and center position may deviate from the end position set in End point (EndPoint).

Circular interpolation error tolerance (CircularErrorTolerance) sets the allowable width of errors between the calculated arc path and the end position.

Setting range	Description
Calculated error ≤ Circular interpolation error tolerance (CircularErrorTolerance)	Circular interpolation control to the set end point address is performed while the error compensation is carried out by spiral interpolation. 
Calculated error > Circular interpolation error tolerance (CircularErrorTolerance)	At the positioning start, Circular interpolation error allowable value over (error code: 1AD9H) occurs and the control will not start. The machine will immediately stop if the error is detected during positioning control.

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following buffer modes are settable in MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control).

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.
1: Buffered(mcBuffered)	Buffers the next FB on the under-control FB. If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after completion of the under-control FB.
2: BlendingLow(mcBlendingLow)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the lower target velocity between the under-control FB and the buffering FB is used as the switching speed.
3: BlendingPrevious(mcBlendingPrevious)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the under-control FB is used as the switching speed.
4: BlendingNext(mcBlendingNext)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the buffering FB is used as the switching speed.
5: BlendingHigh(mcBlendingHigh)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the higher target velocity between the under-control FB and the buffering FB is used as the switching speed.

*1 Stops are not performed between the under-control FB and the buffering FB.

Point

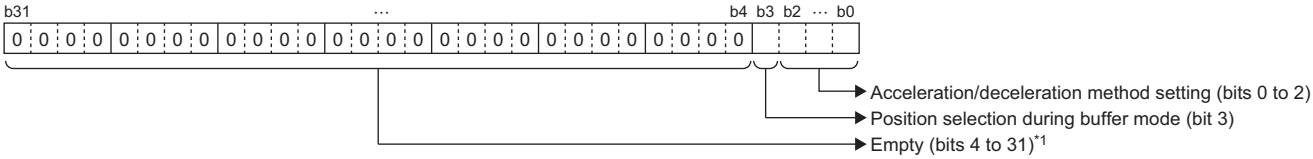
For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

■Options (Options)

The function options for MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



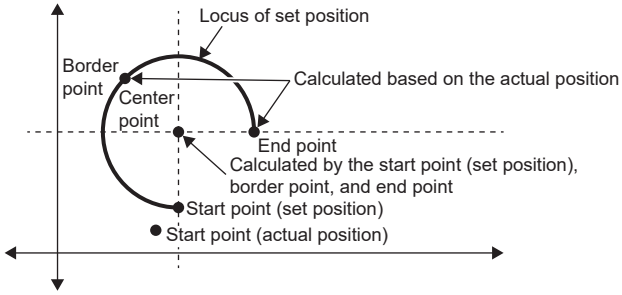
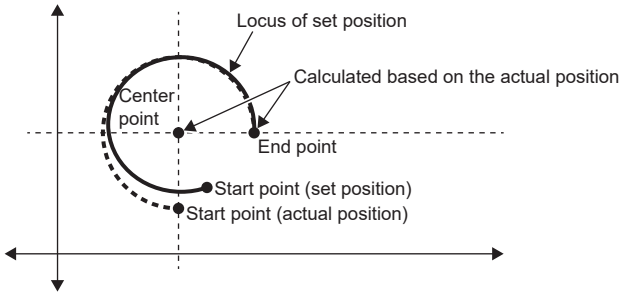
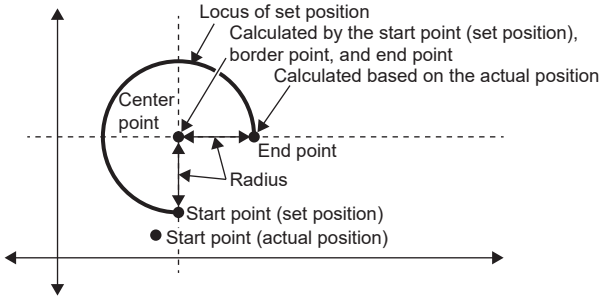
*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
0 to 2	Acceleration/deceleration method setting	The acceleration/deceleration method used to perform control is set. • 0: Acceleration/deceleration specification method (mcAccDec) • 1: Acceleration/deceleration time-fixed method (mcFixedTime)
3	Position selection during buffer mode	The position when executing relative positioning control by multiple start is set. • 0: Set position • 1: Actual position *: This bit is valid when setting "0: Aborting (mcAborting)" in Buffer mode (BufferMode).

• Acceleration/deceleration method setting (bit 0 to 2)

Setting value	Description
0: Acceleration/deceleration specification method (mcAccDec)	An acceleration/deceleration method using Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk) set in MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control).
1: Acceleration/deceleration time-fixed method (mcFixedTime)	An acceleration/deceleration method using the acceleration/deceleration time specified in MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) regardless of the velocity. The acceleration/deceleration time does not use Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk).

• Position selection during buffer mode (Bit 3)

Setting value	Description
0: Set position	Relative positioning control is performed from the set position.
1: Actual position	Relative positioning control is performed from the actual position. The sub point (border point, center point, and radius) and end point are calculated based on the actual position, and the set position is used as the start point. The operation for each type of Circular interpolation mode (CircMode) is shown below. ■When "0: Border point specification (mcBorder)" is set The border point and end point is calculated based on the actual position. The center point is determined by the start point (set position), border point, and end point.  ■When "1: Center point specification (mcCenter)" is set The center point and end point are calculated based on the actual position. The arc of the circle may be an elliptic circle in some cases.  ■When "2: Radius specification (mcRadius)" is set The end point is calculated based on the actual position. The center point is determined by the start point (set position), radius, and end point. 

■Necessary object data

When using MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control), set the following object data for all structuring axes in the specified axes group.

- Target position (607AH)

If there is a structuring axis for which no object data is set, Necessary Object Data Unset (error code: 1AF7H) occurs and the axis will not start.

For details on object data settings, refer to the following.

📖 User's manual for the controller used

Precautions

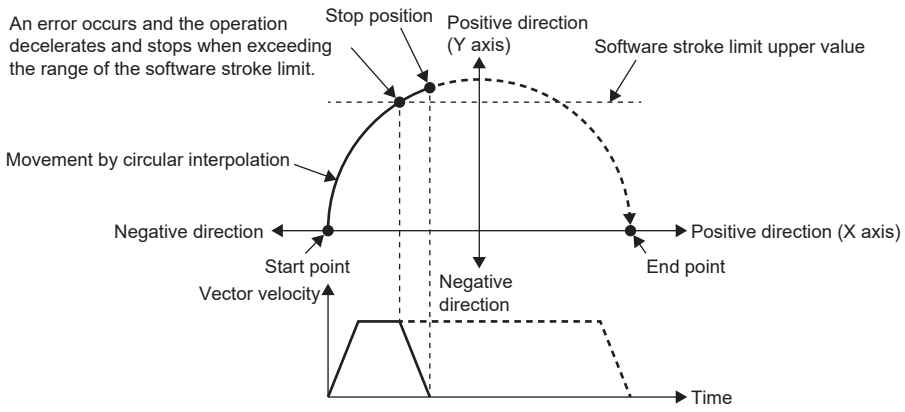
- Circular interpolation including an axis whose stroke limit is set to invalid cannot be performed. Software stroke limit invalid at circular interpolation (error code: 1ADAH) occurs and the operation does not start.
- When the interpolation path exceeds the stroke limit range during interpolation operation, Software stroke limit over (positive direction) (error code: 1A83H) or Software stroke limit over (negative direction) (error code: 1A84H) occurs and the operation will stop.

Ex.

When the upper limit of the software stroke limit is exceeded in the positive direction of the Y-axis

For an immediate stop, stops at the same time that the error occurs.

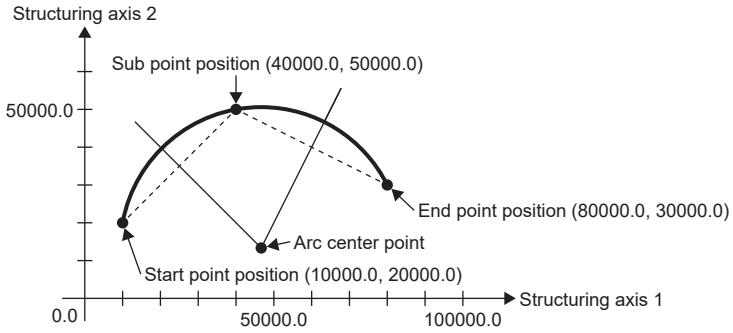
For a deceleration stop, decelerates to a stop along the arc of the circle as shown below.



Program example

The following is a program example. The circular interpolation control start (bCircularInterpolateCMD) is set to TRUE, and Axes group 1 (AxesGroup001) is enabled. Then, 2-axis circular interpolation (sub point specification) relative positioning control is performed for Axis group 1 (AxesGroup001).

- Operation



- Axes group

Item	Setting value
Structuring axis[1]	Axis0001
Structuring axis[2]	Axis0002

- Setting

Item	Setting value	
	Structuring axis 1	Structuring axis 2
Sub point	30000.0	30000.0
End point	70000.0	10000.0
Velocity	50000.0	
Acceleration	50000.0	
Deceleration	50000.0	
Jerk	0.0	
Circular interpolation error tolerance	100.0	

■Axes

Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1
2	Axis0002	AXIS_REF	Axis 2

■Axes group

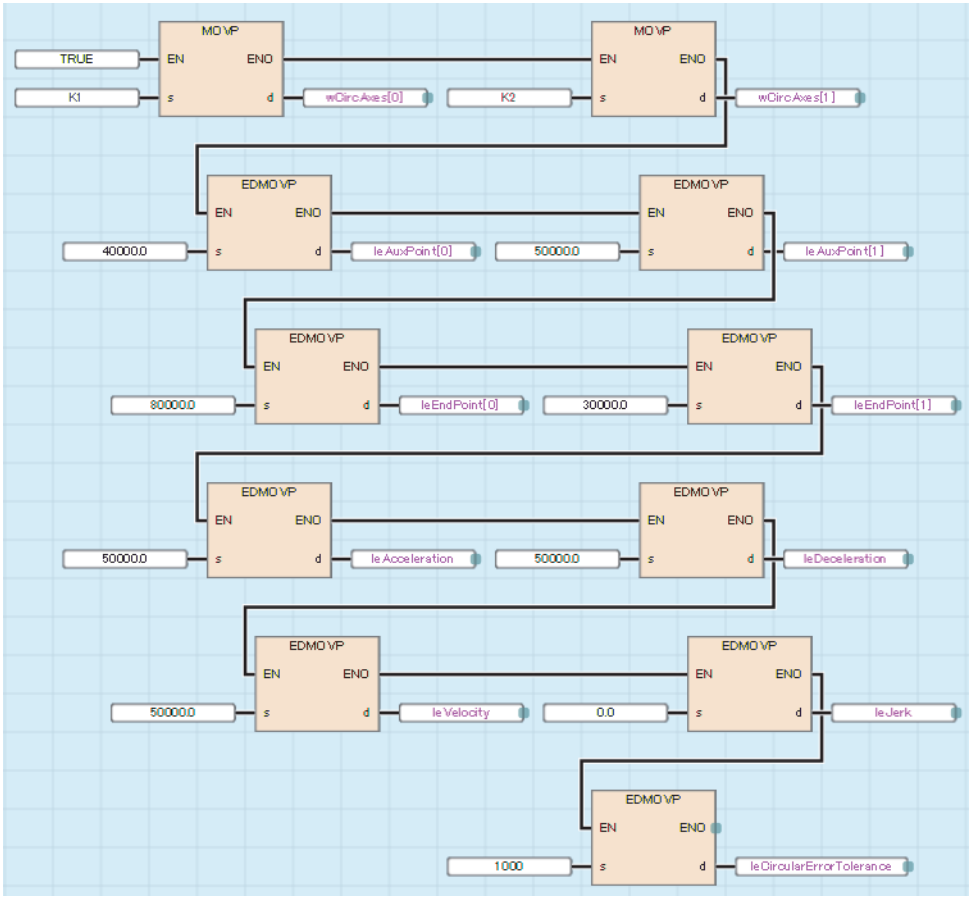
Axes group No.	Label name	Data type	Comment
1	AxesGroup001	AXES_GROUP_REF	Axes group 1

■Labels used

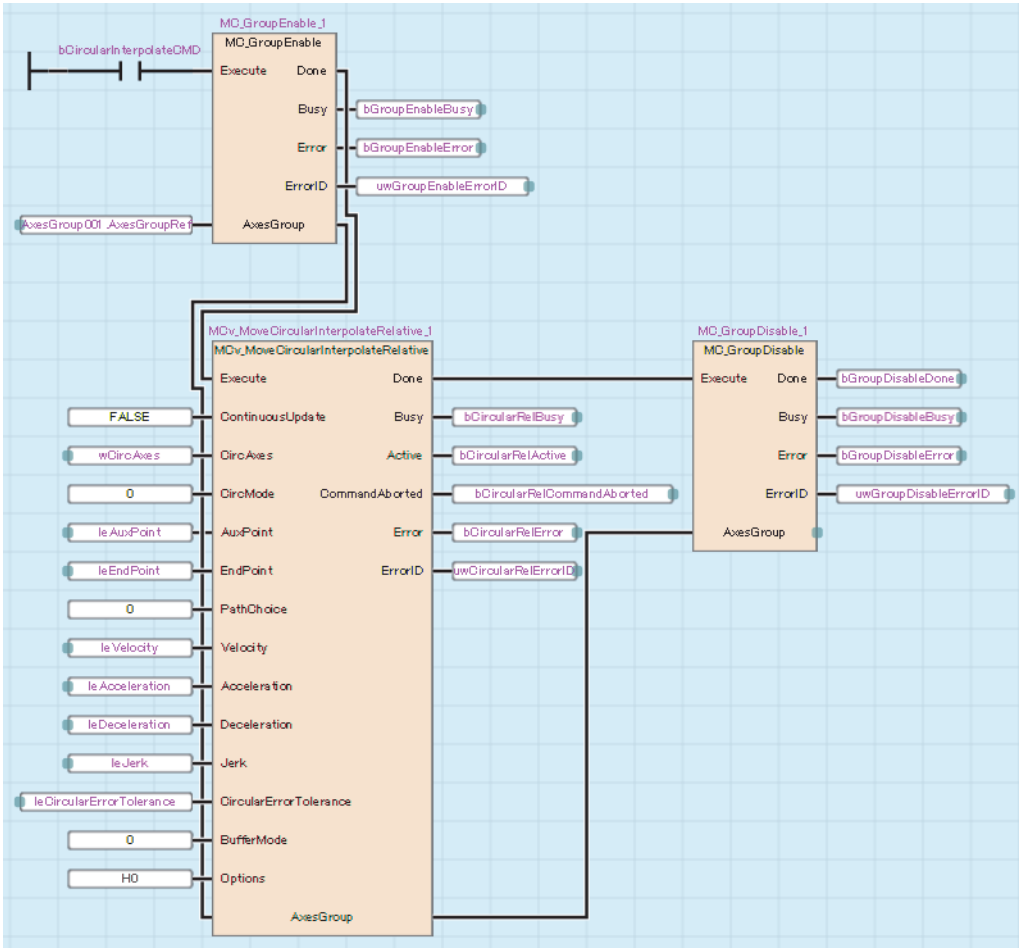
Label name	Data type	Comment
MCv_MoveCircularInterpolateRelative_1	MCv_MoveCircularInterpolateRelative	Relative Value Circular Interpolation Control FB
wCircAxes	Word [signed] (0..1)	Circular interpolation axis
leAuxPoint	Double-precision real number(0..15)	Sub point
leEndPoint	Double-precision real number(0..15)	End point
leVelocity	Double-precision real number	Velocity
leAcceleration	Double-precision real number	Acceleration
leDeceleration	Double-precision real number	Deceleration
leJerk	Double-precision real number	Jerk
leCircularErrorTolerance	Double-precision real number	Circular interpolation error tolerance
bCircularRelDone	Bit	Execution completion
bCircularRelBusy	Bit	Executing
bCircularRelActive	Bit	Controlling
bCircularRelCommandAborted	Bit	Abortion of execution
bCircularRelError	Bit	Error
uwCircularRelErrorID	Word [unsigned]/bit string [16-bit]	Error code
MC_GroupEnable_1	MC_GroupEnable	Axes Group Enabled FB
bCircularInterpolateCMD	Bit	Circular interpolation control start
bGroupEnableDone	Bit	Axes Group Enabled Completion
bGroupEnableBusy	Bit	Executing
bGroupEnableError	Bit	Error
uwGroupEnableErrorID	Word [unsigned]/bit string [16-bit]	Error code
MC_GroupDisable_1	MC_GroupDisable	Axes Group Disabled FB
bGroupDisableDone	Bit	Axes Group Disabled Completion
bGroupDisableBusy	Bit	Executing
bGroupDisableError	Bit	Error
uwGroupDisableErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data setting for 2-axis circular interpolation control



• Axes Group Enabled/Circular interpolation control (Border point specification)/Axis Group Disabled



■ST program

```
//-----Data setting for 2-axis circular interpolation control-----
wCircAxes[0]:= 1;
wCircAxes[1]:= 2;
leAuxPoint[0]:= 30000.0;
leAuxPoint[1]:= 30000.0;
leEndPoint[0]:= 70000.0;
leEndPoint[1]:= 10000.0;
leVelocity:= 50000.0;
leAcceleration:= 50000.0;
leDeceleration:= 50000.0;
leJerk:= 0.0;
leCircularErrorTolerance:= 100.0;

//-----Axes Group Enabled-----
MC_GroupEnable_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bCircularInterpolateCMD ,
   Done=> bGroupEnableDone ,
   Busy=> bGroupEnableBusy ,
   Error=> bGroupEnableError ,
   ErrorID=> uwGroupEnableErrorID
);

//-----Circular interpolation control (Border point specification)-----
MCv_MoveCircularInterpolateRelative_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bGroupEnableDone ,
   ContinuousUpdate:= FALSE ,
   CircAxes:= wCircAxes ,
   CircMode:= MC_CIRC_MODE__mcBorder ,
   AuxPoint:= leAuxPoint ,
   EndPoint:= leEndPoint ,
   PathChoice:= MC_CIRC_PATHCHOICE__mcCW ,
   Velocity:= leVelocity ,
   Acceleration:= leAcceleration ,
   Deceleration:= leDeceleration ,
   Jerk:= leJerk ,
   CircularErrorTolerance:= leCircularErrorTolerance ,
   BufferMode:= MC_BUFFER_MODE__mcAborting ,
   Options:= H00000000 ,
   Done=> bCircularRelDone ,
   Busy=> bCircularRelBusy ,
   Active=> bCircularRelActive ,
   CommandAborted=> bCircularRelCommandAborted ,
   Error=> bCircularRelError ,
   ErrorID=> uwCircularRelErrorID
);
```

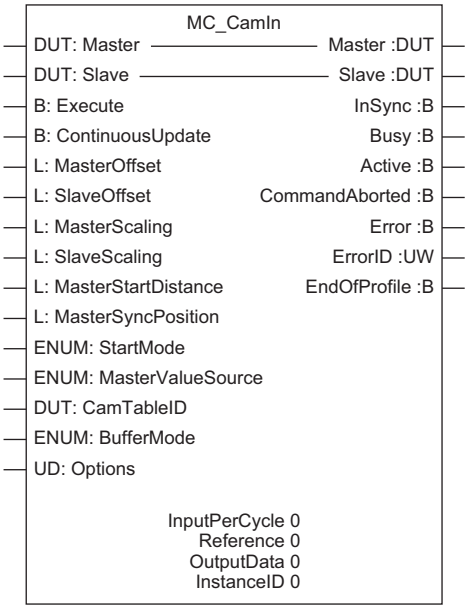
```
//-----Axes Group Disabled-----
MC_GroupDisable_1(
    AxesGroup:= AxesGroup001.AxesGroupRef ,
    Execute:= bCircularRelDone ,
    Done=> bGroupDisableDone ,
    Busy=> bGroupDisableBusy ,
    Error=> bGroupDisableError ,
    ErrorID=> uwGroupDisableErrorID
);
```

42.15 Cam Operation Start

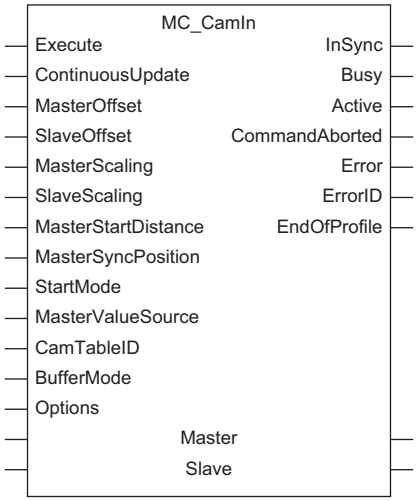
MC_CamIn

This FB starts cam operation according to the specified cam data.

Ladder



FBD/LD



ST

```
MC_CamIn(
  Master:= ?AXIS_REF? ,
  Slave:= ?AXIS_REF? ,
  Execute:= ?BOOL? ,
  ContinuousUpdate:= ?BOOL? ,
  MasterOffset:= ?LREAL? ,
  SlaveOffset:= ?LREAL? ,
  MasterScaling:= ?LREAL? ,
  SlaveScaling:= ?LREAL? ,
  MasterStartDistance:= ?LREAL? ,
  MasterSyncPosition:= ?LREAL? ,
  StartMode:= ?INT? ,
  MasterValueSource:= ?INT? ,
  CamTableID:= ?MC_CAM_ID? ,
  BufferMode:= ?INT? ,
  Options:= ?DWORD? ,
  InSync=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD? ,
  EndOfProfile=> ?BOOL?
);
```

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Master	Master axis	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)
Slave	Slave axis	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_ CamIn (Cam Operation Start) is executed.
ContinuousUpdate	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	Whether to enable or disable continuous change of Master axis offset (MasterOffset), Slave axis offset (SlaveOffset), Master axis scaling (MasterScaling), Slave axis scaling (SlaveScaling), and Cam table ID (CamTableID) is set. • FALSE: Disabled • TRUE: Enabled
MasterOffset	Master axis offset	LREAL	At start/ Retrigger possible/ Continuous update possible	-10000000000.0 to 10000000000.0	0.0	The phase of Master axis (Master) is shifted by the offset amount. For details, refer to the following. ☞ Page 1768 Master axis offset (MasterOffset)

Input variable	Name	Data type	Input	Valid range	Default value	Description
SlaveOffset	Slave axis offset	LREAL	At start/ Retrigger possible/ Continuous update possible	-10000000000.0 to 10000000000.0	0.0	The displacement of the slave axis (Slave) is shifted by the offset amount. For details, refer to the following. ☞ Page 1769 Slave axis offset (SlaveOffset)
MasterScaling	Master axis scaling	LREAL	At start/ Retrigger possible/ Continuous update possible	0.01 to 10.0	1.0	The cam table length per cycle is expanded or reduced. For details, refer to the following. ☞ Page 1770 Master axis scaling (MasterScaling)
SlaveScaling	Slave axis scaling	LREAL	At start/ Retrigger possible/ Continuous update possible	0.01 to 10.0	1.0	The stroke amount of the cam table is expanded or reduced. For details, refer to the following. ☞ Page 1770 Slave axis scaling (SlaveScaling)
MasterStartDistance	Master axis follow-up distance	LREAL	At start	-10000000000.0 to 10000000000.0	0.0	The position (the relative position from Master axis synchronization start position (MasterSyncPosition)) of Master axis (Master) that will start synchronization by Output axis (OutputData) is set. This variable is enabled when Start mode (StartMode) is set to "2: Relative (mcRelative)" or "3: Ramp-in (mcRampIn)".
MasterSyncPosition	Master axis synchronization start position	LREAL	At start	-10000000000.0 to 10000000000.0	0.0	The position of Master axis (Master) that will start synchronization by Current value per cycle (InputPerCycle) is set. *: If Specifying target of Master axis synchronization start position pass check (Options (Options): Bit 21) is set to "1: Master axis set position, or the feedback position", the valid range is within the ring counter range.
StartMode	Start mode	INT (MC_START_MODE)	At start	0, 1, 2, 3	0	The timing for starting cam operation is set. • 0: Immediate (mcImmediate) • 1: Absolute (mcAbsolute) • 2: Relative (mcRelative) • 3: Ramp-in (mcRampIn) For details, refer to the following. ☞ Page 1771 Start mode (StartMode)
MasterValueSource	Master axis data source selection	INT (MC_SOURCE)	At start	1, 2, 101, 102	1	The data source of Master axis (Master) is set. • 1: Set value (mcSetValue) • 2: Actual value (mcActualValue) • 101: Latest set value (mcLatestSetValue) • 102: Latest actual value (mcLatestActualValue) For details, refer to the following. ☞ Page 1775 Master axis data source selection (MasterValueSource)
CamTableID	Cam table ID	MC_CAM_ID	At start/ Retrigger possible/ Continuous update possible	1 to 60000	0	The cam ID is set. Open the cam ID to the open area with MC_CamTableSelect (Cam Table Selection) before use. Set MC_CAM_REF.CamID or MC_CAM_REF.ProfileData.ID. ☞ Page 1555 Cam Table Selection
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0, 1	0	The buffer mode is set. • 0: Aborting(mcAborting) • 1: Buffered(mcBuffered) For details, refer to the following. ☞ Page 1776 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	00000000H to 00210000H	00000000H	The function options for MC_CamIn (Cam Operation Start) are set in the specified bit position. For details, refer to the following. ☞ Page 1780 Options (Options)

■Output variables

Output variable	Name	Data type	Default value	Description
InSync	In synchronization	BOOL	FALSE	This variable becomes TRUE when Output value (OutputData) starts synchronization.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_CamIn (Cam Operation Start) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE when Current value per cycle (InputPerCycle) starts synchronization.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MC_CamIn (Cam Operation Start) is aborted. The variable becomes FALSE when Execution command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used
EndOfProfile	Cam cycle completion	BOOL	FALSE	After Controlling (Active) becomes TRUE, this variable becomes TRUE only for one execution cycle of the POU (Program Organization Unit) that calls FB each time movement equal to the cycle length occurs. For details, refer to the following. ☞ Page 1780 Cam cycle completion (EndOfProfile)

■Public variables

Public variable	Name	Data type	Default value	Description
InputPerCycle	Current value per cycle	LREAL	0.0	The current value per cycle is stored. For details, refer to the following. ☞ Page 1780 Current value per cycle (InputPerCycle)
Reference	Reference value	LREAL	0.0	The reference value is stored. For details, refer to the following. ☞ Page 1781 Reference value (Reference)
OutputData	Output value	LREAL	0.0	The output value is stored. For details, refer to the following. ☞ Page 1783 Output value (OutputData)
InstanceID	Instance ID	WORD(UINT)	0	The instance ID is stored. This variable is automatically set by the system when an instance is created. This instance ID is used in FB input or others.

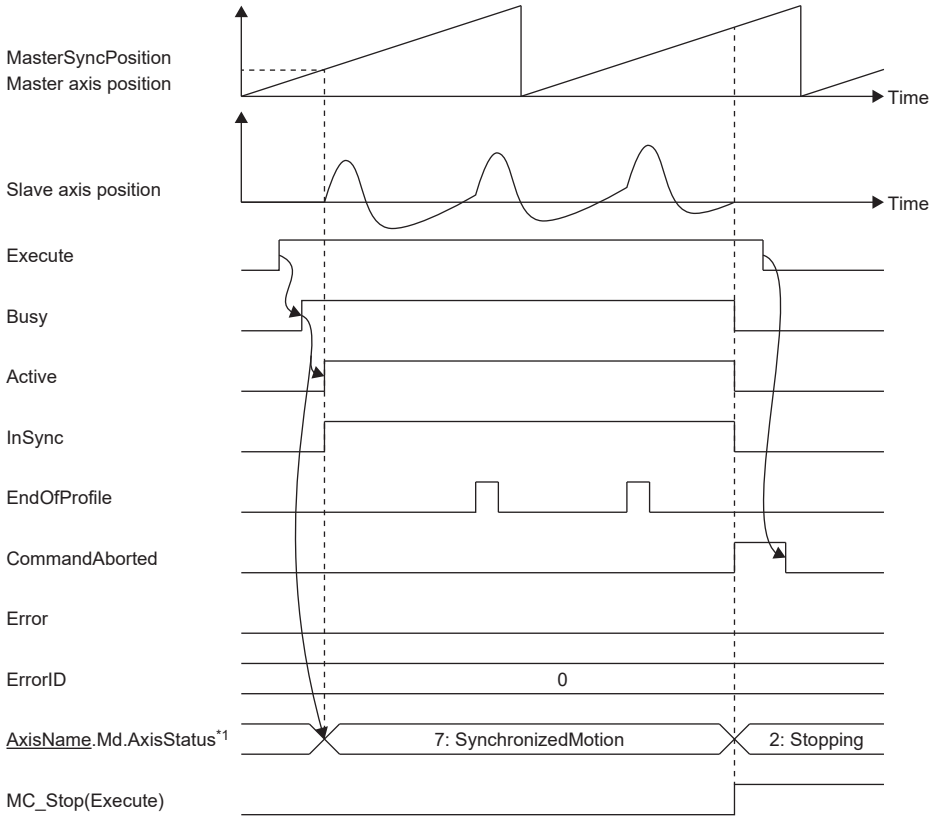
Processing details

- This FB sets Master axis offset (MasterOffset), Slave axis offset (SlaveOffset), Master axis scaling (MasterScaling), Slave axis scaling (SlaveScaling), Master axis follow-up distance (MasterStartDistance), Master axis synchronization start position (MasterSyncPosition), Start mode (StartMode), Master axis data source selection (MasterValueSource), Cam tableID (CamTableID), and Buffer mode (BufferMode), and performs the cam operation.
- To stop the operation, perform MC_Stop (Forced Stop).

■Timing chart

• When the processing is completed successfully

When Start Mode (StartMode) is set to "1: Absolute (mcAbsolute)" and Master axis absolute coordinate (MasterAbsolute) set in MC_CamTableSelect (Cam Table Selection) is "FALSE (Relative coordinate)"



*1 Axis status of Slave axis (Slave)

• When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Starting cam operation

By setting Start Mode (StartMode), the synchronization timing for cam operation and reflection timing for the cam control data can be set.

• Synchronization timing for cam operation

The following shows the synchronization timing for Current value per cycle (InputPerCycle), Reference value (Reference), and Output value (OutputData) based on Start mode (StartMode) and related parameters (Master axis synchronization start position (MasterSyncPosition) and Master axis follow-up distance (MasterStartDistance)).

Start mode (StartMode)*1	Timing for starting synchronization		
	Current value per cycle (InputPerCycle)	Reference value (Reference)	Output value (OutputData)
0: Immediate (mcImmediate)	Synchronization starts when MC_CamIn (Cam Operation Start) is executed. Master axis synchronization start position (MasterSyncPosition) and Master axis follow-up distance (MasterStartDistance) are ignored.		
1: Absolute (mcAbsolute)	Synchronization starts when the master axis position*2 passes Master axis synchronization start position (MasterSyncPosition)*3. Master axis follow-up distance (MasterStartDistance) is ignored.		
2: Relative (mcRelative)	Synchronization starts when the master axis position*2 passes Master axis synchronization start position (MasterSyncPosition)*3.		Synchronization starts when the position of the Master axis*2 has passed the Master axis synchronization start position (MasterSyncPosition) and then passes "Master axis synchronization start position (MasterSyncPosition) + Master axis follow-up distance (MasterStartDistance)"*3.
3: Ramp-in (mcRampIn)			Synchronization starts when the position of the Master axis*2 has passed the Master axis synchronization start position (MasterSyncPosition) and then passes "Master axis synchronization start position (MasterSyncPosition) + Master axis follow-up distance (MasterStartDistance)"*3.*4

*1 For details on Start mode (StartMode), refer to the following.
☞ Page 1771 Start mode (StartMode)

*2 Set the position data to be used based on Specifying target of Master axis synchronization start position pass check (Options (Options): Bit 21). For details, refer to the following.
☞ Page 1780 Options (Options)

*3 The position of the Master axis has crossed over Master axis synchronization start position (MasterSyncPosition) (or, the Master axis synchronization start position (MasterSyncPosition) + Master axis follow-up distance (MasterStartDistance)). When these values are the same, the Master axis is considered to have not passed, and should the Master axis move from this value, it will be considered to have passed.

*4 After the position of the Master axis passes the Master axis synchronization start position (MasterSyncPosition), the Slave axis is controlled toward the Slave axis position at the start of synchronization until the Master axis passes "Master axis synchronization start position (MasterSyncPosition) + Master axis follow-up distance (MasterStartDistance)". For details, refer to the following.
☞ Page 1771 Start mode (StartMode)

After Cam Operation Start (MC_CamIn) is executed, Controlling (Active) becomes TRUE if Current value per cycle (InputPerCycle) is synchronized, and In synchronization (InSync) becomes TRUE if Output value (OutputData) is synchronized.

• Timing of control change by FB restart/continuous update

The following shows the timing for reflecting changed parameters in control by FB restart/continuous update based on Start mode (StartMode).

Start mode (StartMode)	Timing for reflecting parameters
0: Immediate (mcImmediate)	Immediate
1: Absolute (mcAbsolute)	When the current value per cycle (InputPerCycle) passes the 1st point on the CamTable
2: Relative (mcRelative)	
3: Ramp-in (mcRampIn)	

When the changed values exceed the range, the corresponding error code or event code is output and the control is maintained by keeping the value before.

Parameter	Error
Master axis offset (MasterOffset)	Out of master axis offset range (error code: 1AAFH)
Slave axis offset (SlaveOffset)	Out of slave axis offset range (error code: 1AB0H)
Master axis scaling (MasterScaling)	Out of master axis coefficient range (error code: 1AB1H)
Slave axis scaling (SlaveScaling)	Out of slave axis coefficient range (error code: 1AB2H)
Cam table ID (CamTableID)	• When the Cam table ID (CamTableID) value is out of range Out of cam table ID range (error code: 1AAEH) • When there is no Cam table ID (CamTableID) open area No Cam Table warning (event code: 00D44H)

■Cam table information

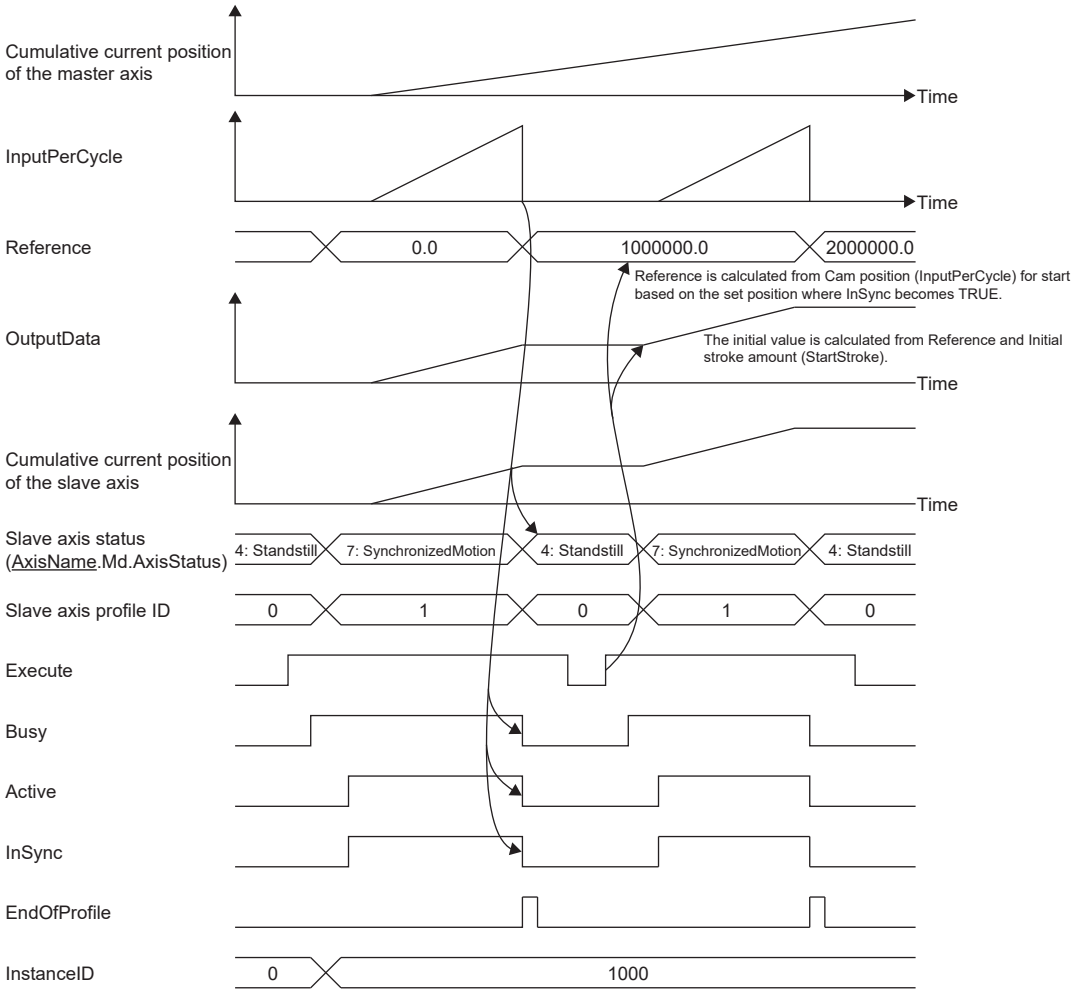
For cam table information, the following shows the cam operation at execution of MC_CamIn (Cam Operation Start) depending on the periodic mode, master axis absolute coordinate, and slave axis absolute coordinate set in MC_CamTableSelect (Cam Table Selection).

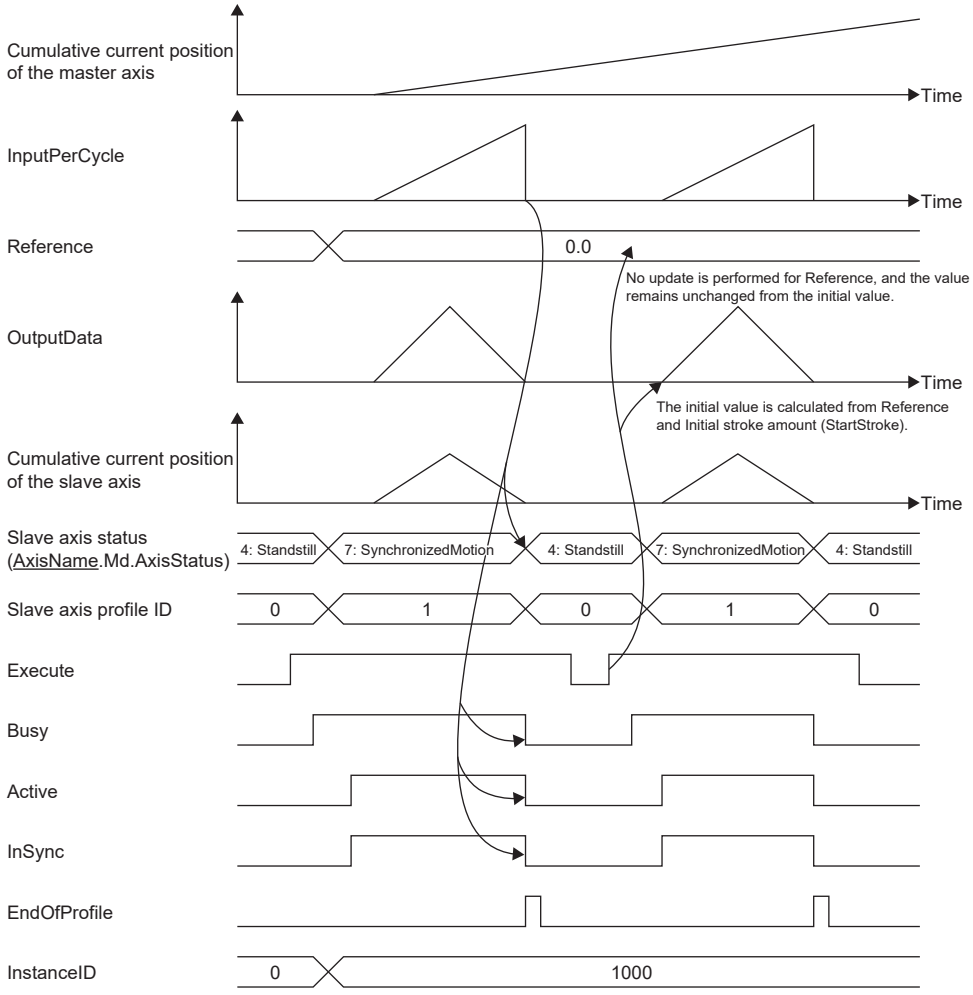
○: Used in control, ×: Not used in control

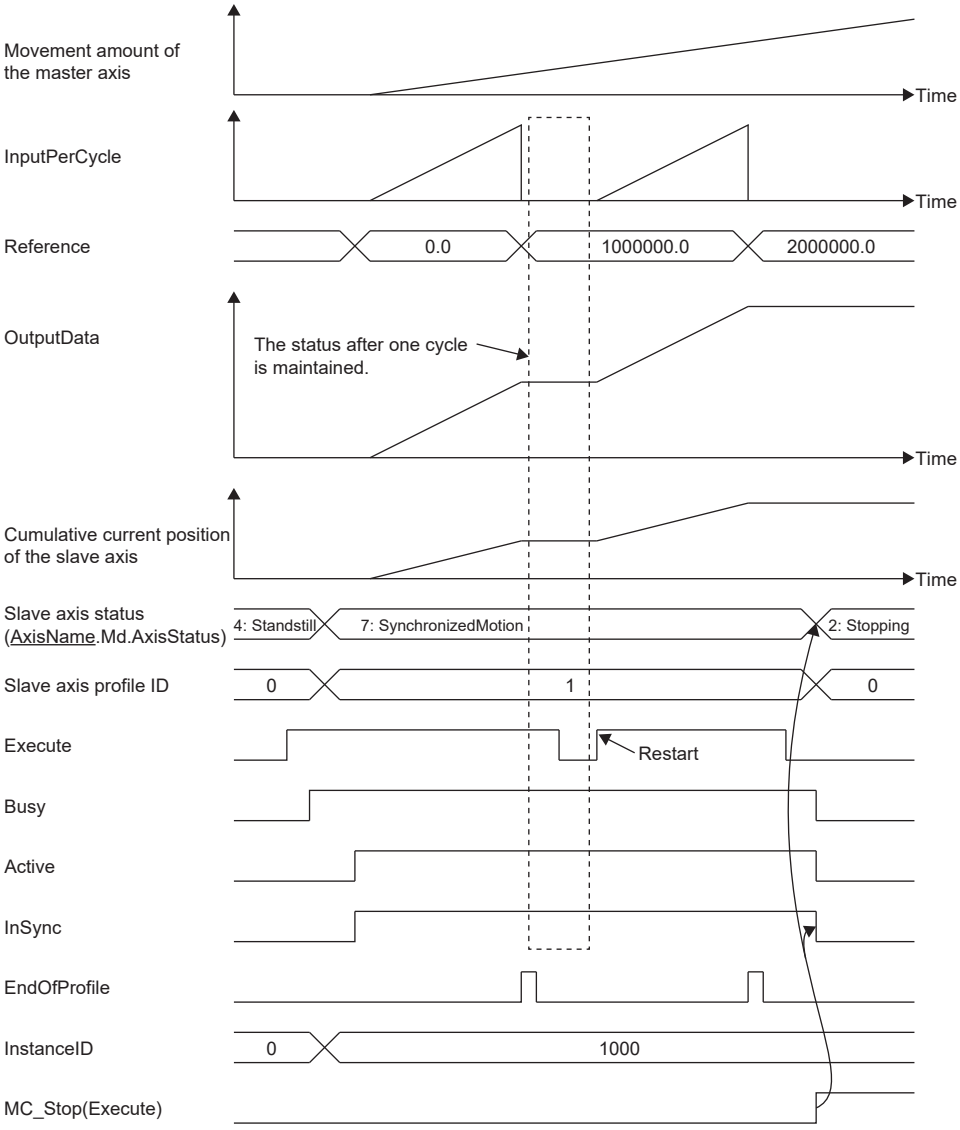
Name	Variable name	Specifying interpolation method (Interpolate)			Remark
		0: Linear interpolation	1: Specifying for each section	2: Spline interpolation	
Periodic	Periodic	○	○	○	These items can be set in the expand setting of the operation profile data using the engineering tool, or in the MC_CamTableSelect (Cam Table Selection) input variables.
Master axis absolute coordinate	MasterAbsolute	○	○	○	
Slave axis absolute coordinate	SlaveAbsolute	○	○	○	
Specifying interpolation method	Interpolate	○	○	○	These are items related to the control of data opened to the open area from a cam data format/rotary cutter format operation profile data.
Length per cycle	CycleLength	○	○	○	
Stroke amount	Stroke	○	○	○	
Start point	StartPoint	×	×	×	
Initial stroke amount	StartStroke	×	○	○	

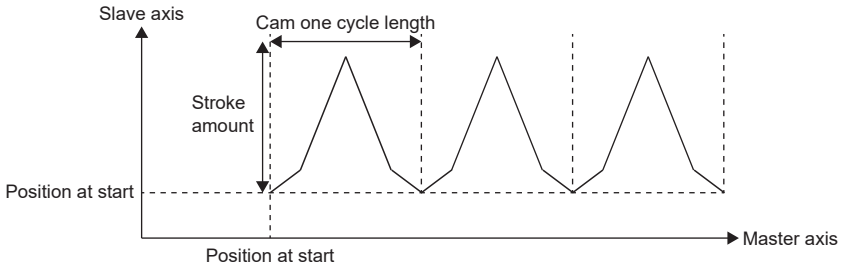
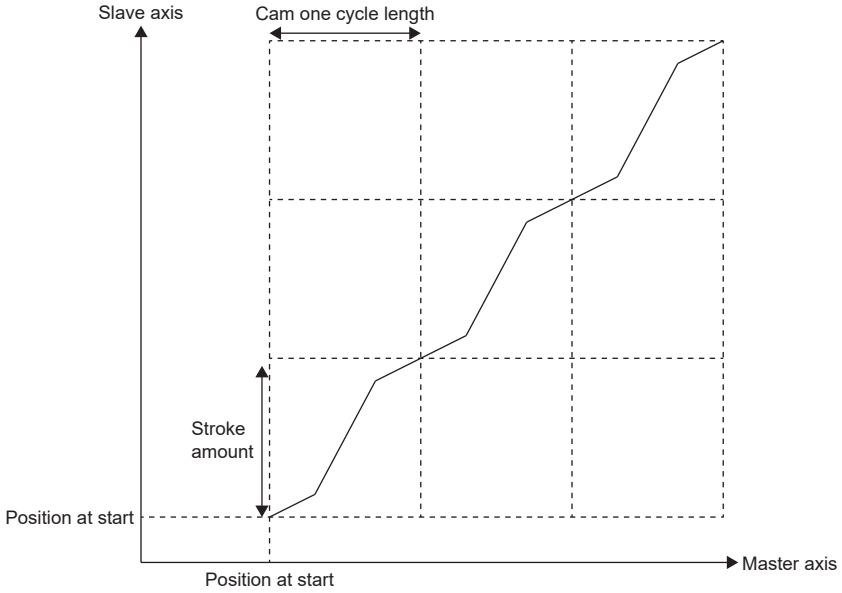
• Periodic (Periodic)

The following shows the operation at execution of MC_CamIn (Cam Operation Start) by the setting of Periodic (Periodic).

Periodic (Periodic)	Description
FALSE (Non periodic)	Operation is performed for one cycle only after Controlling (Active) becomes TRUE. Following this, the operation set in Specifying operation after one cycle operation (Options (Options): Bit 16) is performed. ■When Specifying operation after one cycle operation (Options (Options): Bit 16) is set to "0: Complete" Synchronization status is cancelled after one cycle, and Reference value (Reference) and Output value (OutputData) are initialized when Execution command (Execute) is started again. • When Slave axis absolute coordinate (SlaveAbsolute) is set to FALSE (Relative coordinate) 

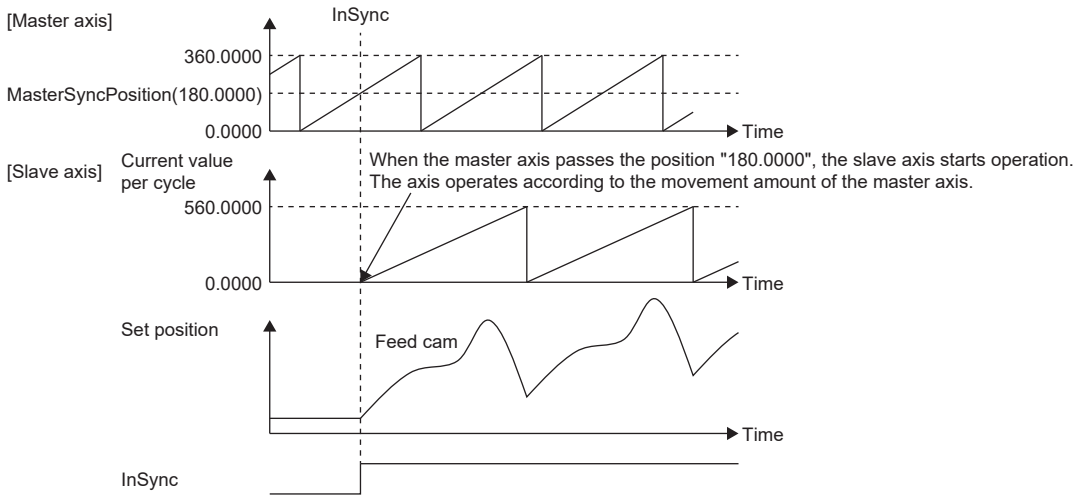
Periodic (Periodic)	Description
FALSE (Non periodic)	• When Slave axis absolute coordinate (SlaveAbsolute) is set to TRUE (Absolute Coordinate) 

Periodic (Periodic)	Description
FALSE (Non periodic)	■When Specifying operation after one cycle operation (Options (Options): Bit 16) is set to "1: Wait for restart" Synchronization status is maintained after one cycle, and Reference value (Reference) and Output value (OutputData) keep their value when Execution command (Execute) is started again. Input stop cause to the slave axis to cancel synchronization status. During a restart after one cycle has completed, even when values of input labels where continuous update is possible are changed, any input information will not be imported until Execution command (Execute) is started. If Execution command (Execute) is started, regardless of the Start mode (StartMode) settings, the slave axis will immediately start moving. While waiting for a restart after one cycle has completed, if a change to the current value per cycle is executed, this will be valid from the next cycle. 

Periodic (Periodic)	Description
TRUE (Periodic)	Continuously repeats cam table execution. The following shows the operation based on the cam operation (two-way cam or feed cam). ■Two-way cam When the position of Slave axis (Slave) used by Periodic for the cam table start point is the same each time, operates as a two-way cam.  ■Feed cam When the position of Slave axis (Slave) used by Periodic for the cam table start point has a different set position at the start point and the end point, operates as a feed cam. 

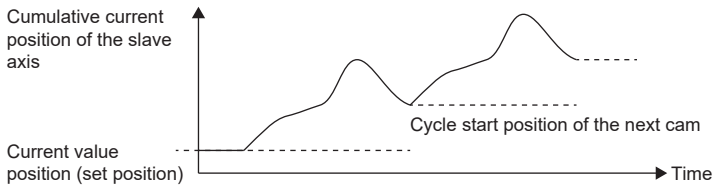
• Master axis absolute coordinate (MasterAbsolute)

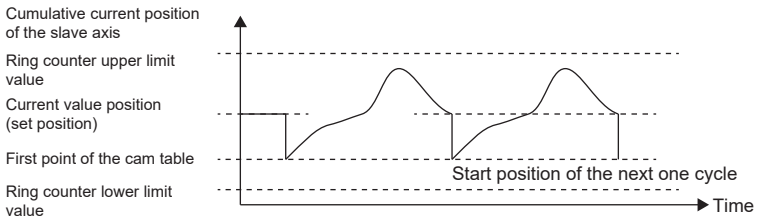
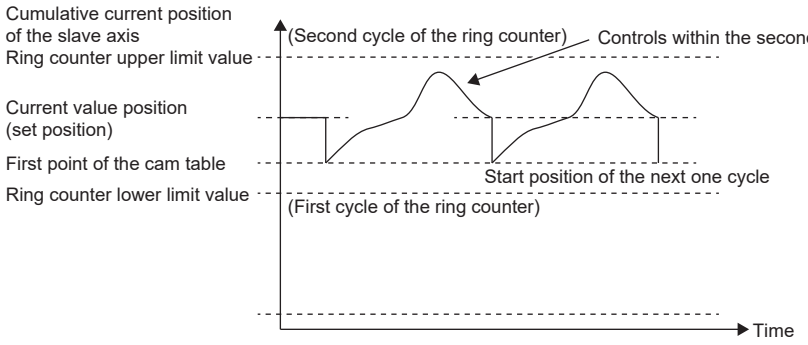
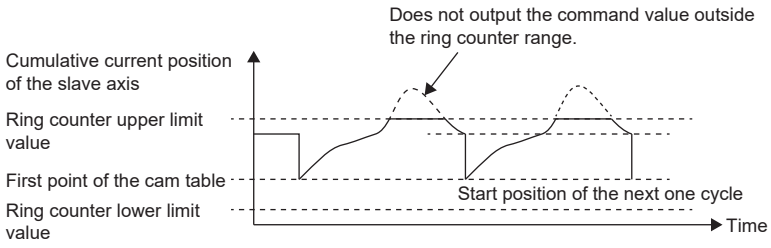
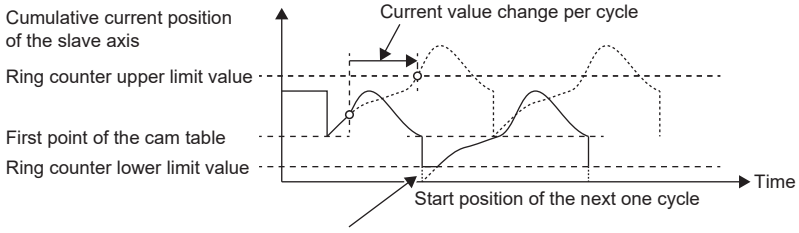
The following shows the cam operation depending on the setting of Master axis absolute coordinate (MasterAbsolute).

Master axis absolute coordinate (MasterAbsolute)	Description
FALSE (Relative coordinate)	The master axis synchronization start position (MasterSyncPosition) becomes the start point of the cam data. When In synchronization (InSync) becomes TRUE, cam operation is executed corresponding to the relative movement amount of Master axis (Master). Cam operation is executed continuously, even when the ring counters of Master axis (Master) and the cam table are not matched. Example: When the following settings are used • Master axis: Ring counter of the current value: 0.0000 to 360.0000 [degree] • Master axis synchronization start position (MasterSyncPosition): 180.0000 [degree] • Cam: Cam length per cycle: 540.0000 [degree] 

• Slave axis absolute coordinate (SlaveAbsolute)

The following shows the cam operation depending on the setting of Slave axis absolute coordinate (SlaveAbsolute).

Slave axis absolute coordinate (SlaveAbsolute)	Description
FALSE (Relative coordinate)	When In synchronization (InSync) becomes TRUE, the slave axis starts the operation of the cam table from the current position (set position). When Periodic (Periodic) of the cam table is set to "TRUE: Periodic", the next cycle starts from the stroke position (set position) where the last cycle completed. 

Slave axis absolute coordinate (SlaveAbsolute)	Description
TRUE (Absolute coordinate)	The cam is operated within the ring counter range including current values of the Slave axis (Slave) at the point when In synchronization (InSync) becomes TRUE. Command is output per operation cycle to return the set position of the Slave axis (Slave) to the start point of the cam table when In synchronization (InSync) and when the cam cycle completion (EndOfProfile) become TRUE. If the command at this time is too large, the position command and speed command to the drive unit also become too large, causing a servo error ("AL.035 (Command frequency error)") for the MR-J5(W)-G. The set position of the Slave axis (Slave) will not output any values outside the ring counter range. ■When the Stroke Range is within the ring counter range of the Slave axis  ■When the Stroke Range is within the ring counter range of the Slave axis (Cam started on the 2nd cycle)  ■When the Stroke Range is outside the ring counter range of the Slave axis  ■When current value per cycle is changed • To compensate the Reference value (Reference) so that the Slave axis (Slave) does not operate, any range outside of the ring counter range will be changed. When the Slave axis (Slave) has moved and the current value per cycle has changed, change the Master axis offset (MasterOffset). 

• Length per cycle (CycleLength)

For the length per cycle, the input amount that is required for one cycle is set. For details, refer to the following.

☞ Page 1780 Current value per cycle (InputPerCycle)

• Stroke amount (Stroke)

For the stroke amount, the stroke amount corresponding to the stroke ratio at 100% is set. For details, refer to the following.

☞ Page 1783 Output value (OutputData)

• Start point (StartPoint) and Initial stroke amount (StartStroke)

The Start Point (StartPoint) used for cam operation and the Initial stroke amount (StartStroke) will differ depending on the Specifying interpolation method (Interpolate).

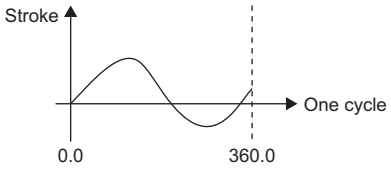
Specifying interpolation method (Interpolate)	Cam operation Start Point	Cam operation initial stroke amount
0: Linear interpolation	Input value of the cam table 1st point	Output value of the cam table 1st point
1: Specifying for each section	Start Point (StartPoint) of the cam table 1st point	Initial stroke amount (StartStroke) of the cam table
2: Spline interpolation		

■Cam operation Start Point

Operating with a Current value per cycle (InputPerCycle) of "0.0".

• Cam table

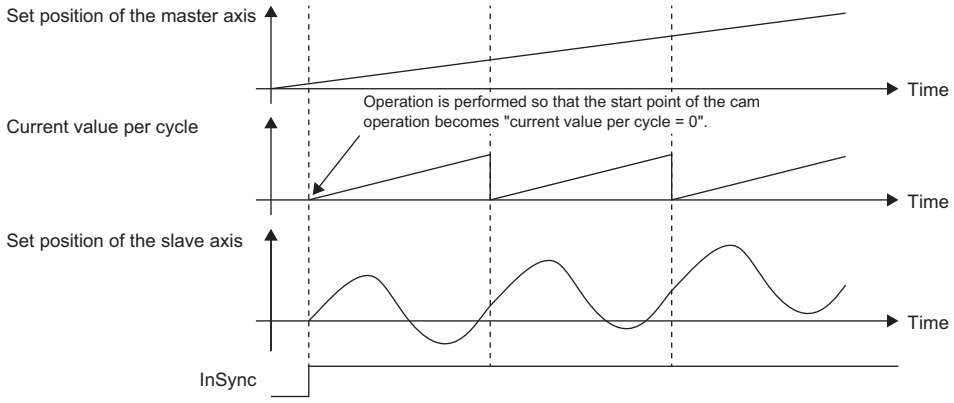
■Cam (1) (Cam operation start point "0.0")



• Interpolation method: Specifying for each section
• Length per cycle: 360.0
• Start point: 0.0
• Initial stroke amount: 0.0

• Cam operation

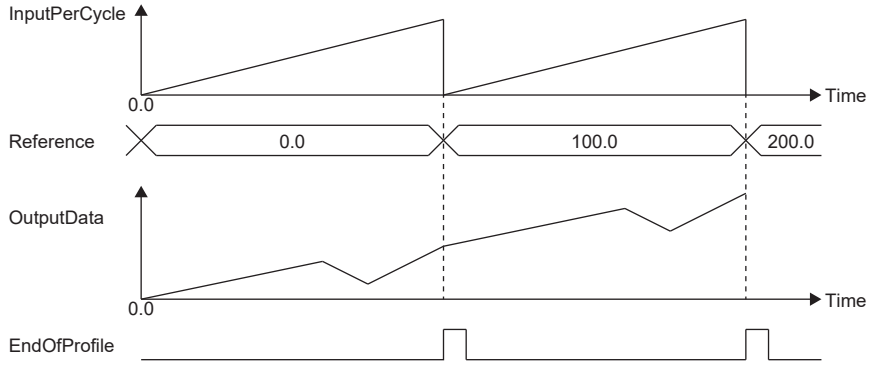
The operation is shown below.



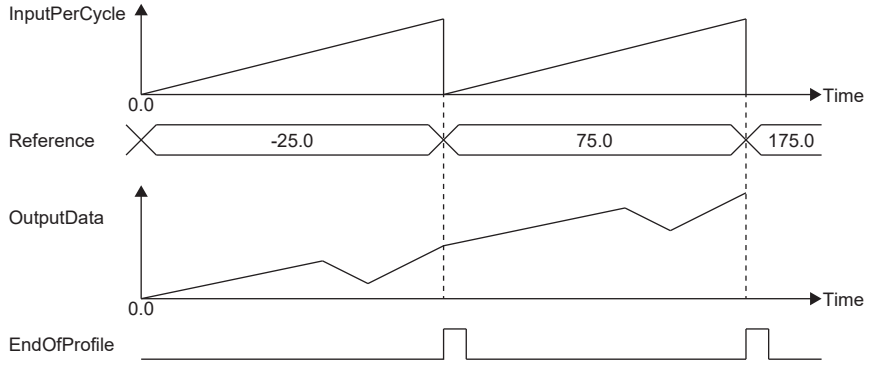
■Cam operation initial stroke amount

Operation is performed so that the initial stroke amount equals the Output value (OutputData) at the start of synchronization. Because the initial stroke amount of the cam operation stops the Slave axis (Slave) from operating, the initial stroke amount from the reference axis (Reference) is decreased.

• When cam operation and Initial stroke amount (StartStroke) is "0.0"



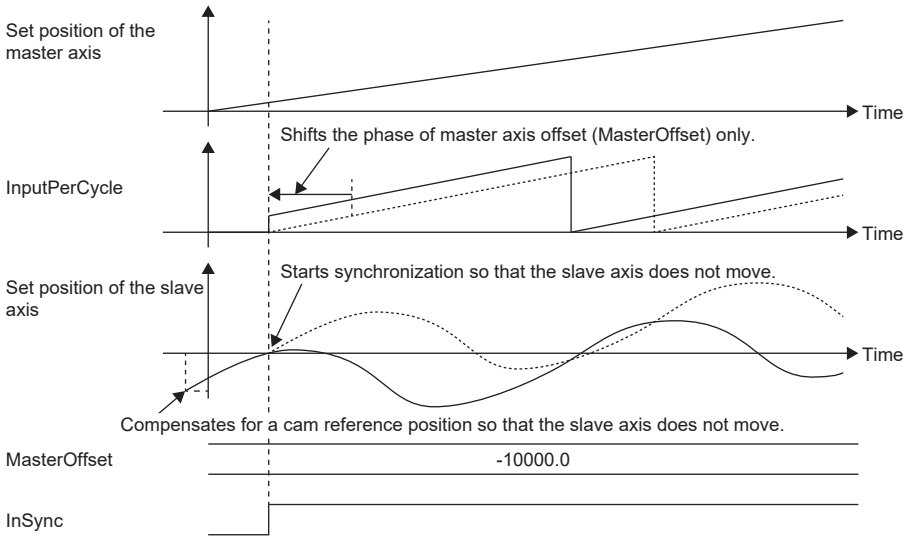
• When cam operation and Initial stroke amount (StartStroke) is "25.0"



■Master axis offset (MasterOffset)

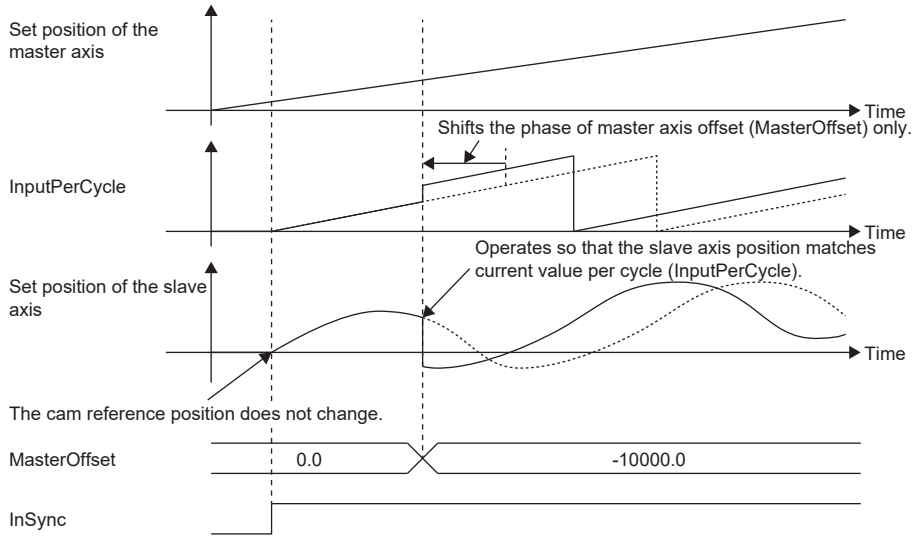
Setting Master axis offset (MasterOffset) shifts the phase of Master axis (Master) by the offset amount. (This does not affect Master axis follow-up distance (MasterStartDistance) or Master axis synchronization start position (MasterSyncPosition).) When a value other than "0.0" is set at operation start, the position of Master axis (Master) is changed at the rising edge of In synchronization (InSync) to the cam position per cycle position with the offset amount added. When this happens, the Reference value (Reference) is compensated so that the Slave axis (Slave) does not operate.

- When Start Mode (StartMode) is "0: Immediate (mcImmediate)"



When the master axis offset (MasterOffset) is changed while In Synchronization (InSync) is TRUE, the position of the Slave axis (Slave) operates matching the current value per cycle with Master axis Offset (MasterOffset) added.

- When Start Mode (StartMode) is "0: Immediate (mcImmediate)"

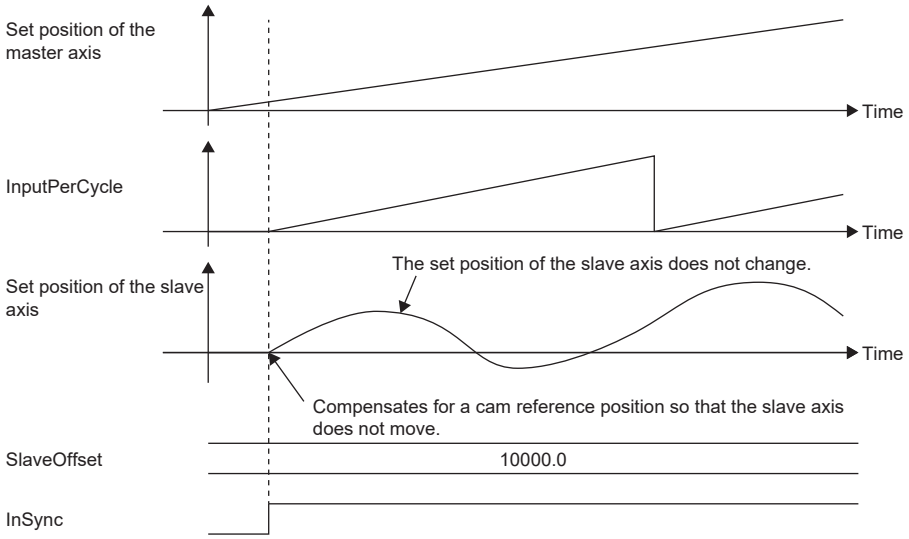


■Slave axis offset (SlaveOffset)

This variable shifts the displacement of the slave axis (Slave) by the offset amount.

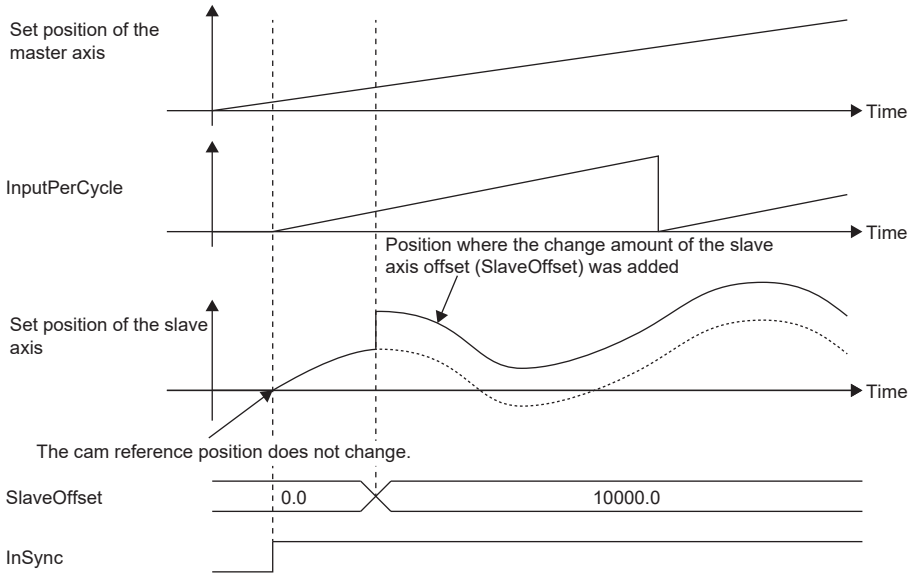
When a value other than "0.0" is set before the rising edge of In synchronization (InSync), the Reference value (Reference) is compensated so that the Slave axis (Slave) does not operate at the rising edge of In synchronization (InSync).

• When Start Mode (StartMode) is "0: Immediate (mcImmediate)"



When the slave axis offset (SlaveOffset) is changed while In Synchronization (InSync) is TRUE, the position of the Slave axis (Slave) operates matching the position with the offset amount added.

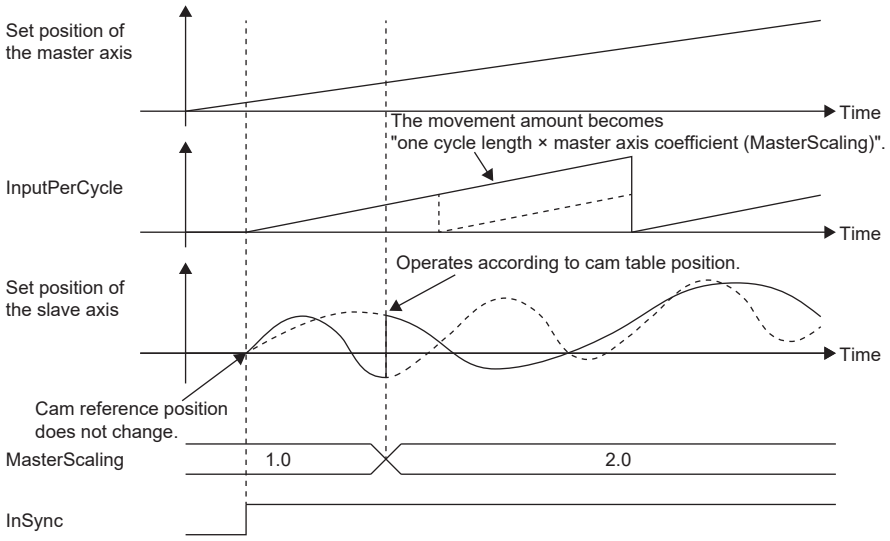
• When Start Mode (StartMode) is "0: Immediate (mcImmediate)"



■Master axis scaling (MasterScaling)

Master axis scaling (MasterScaling) expands/reduces the cam table length per cycle.

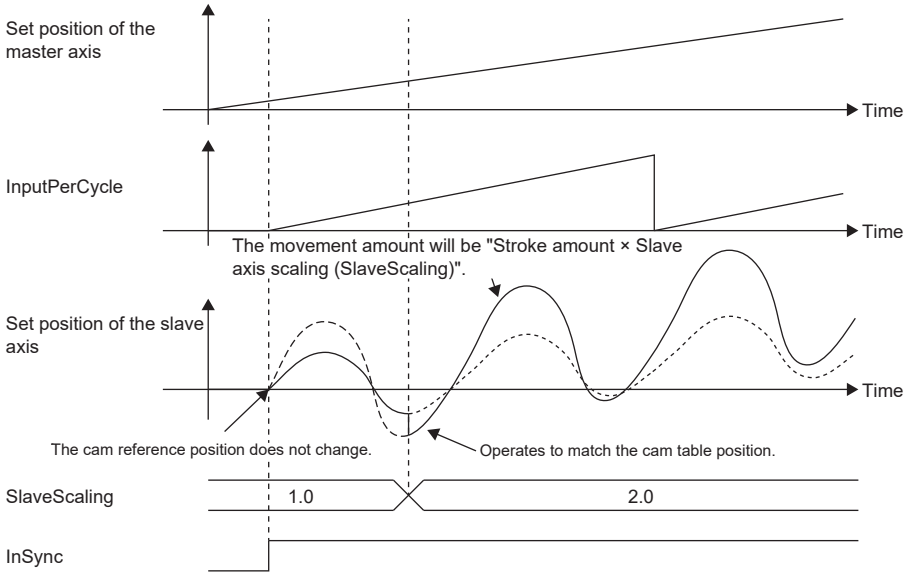
When changing during synchronization, a command equal to the movement amount is output to adjust the position of Slave axis (Slave) to the position of the cam table.



■Slave axis scaling (SlaveScaling)

Slave axis scaling (SlaveScaling) expands/reduces the stroke amount of the cam table.

When changing during synchronization, a command equal to the movement amount is output to adjust the position of Slave axis (Slave) to the position of the cam table.



■Start mode (StartMode)

The synchronization timing is set for cam operation and reflection timing is set for the cam control data.

For the synchronization timing of cam operation, refer to the following.

☞ Page 1758 Starting cam operation

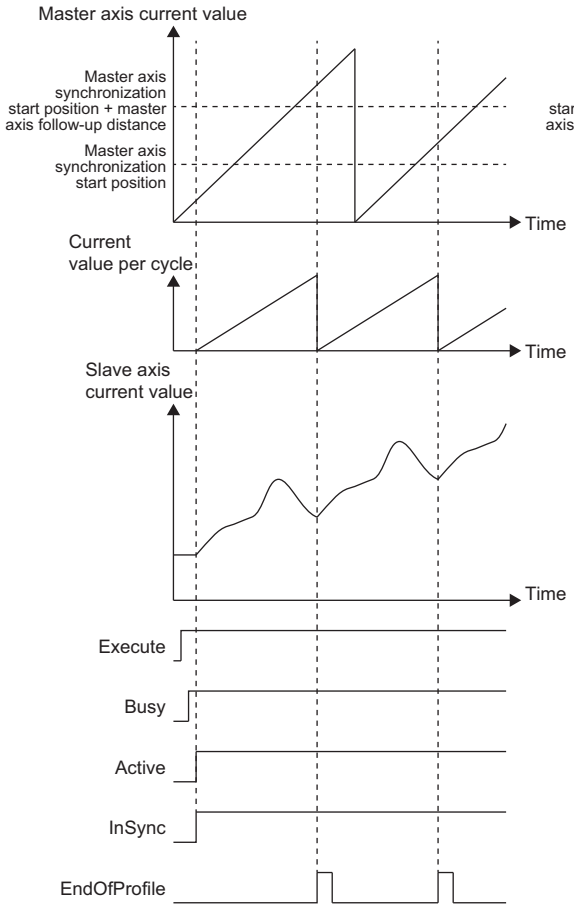
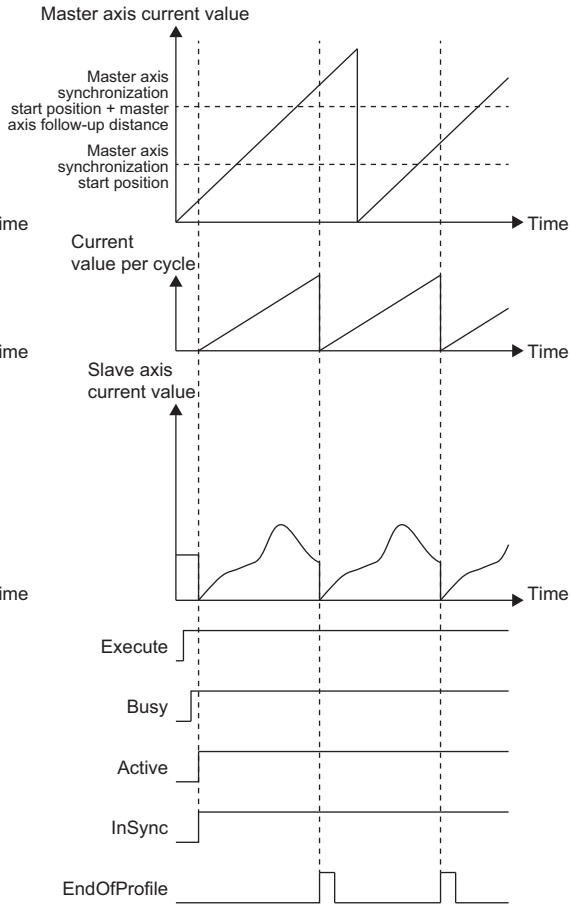
If a value outside the range is set, Out of Start Mode Range (error code: 1AB5H) occurs.

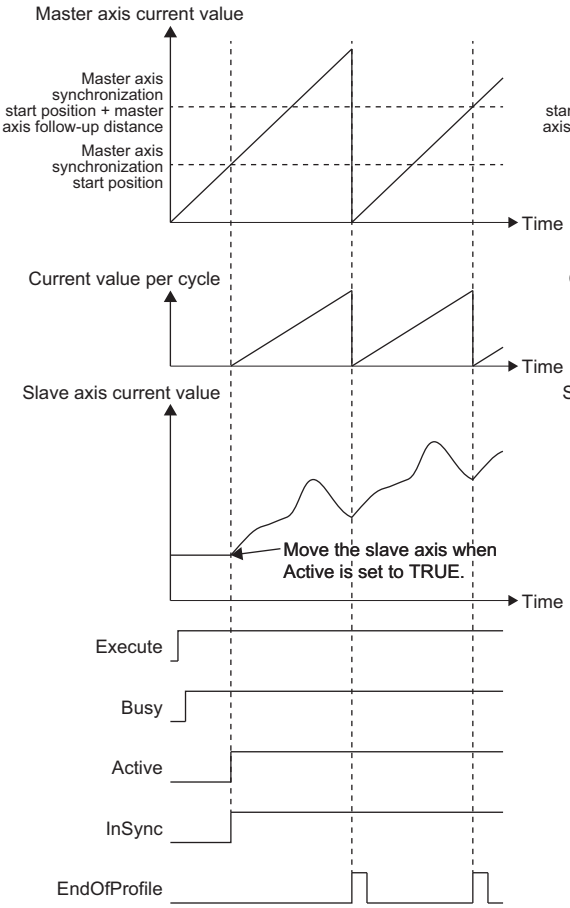
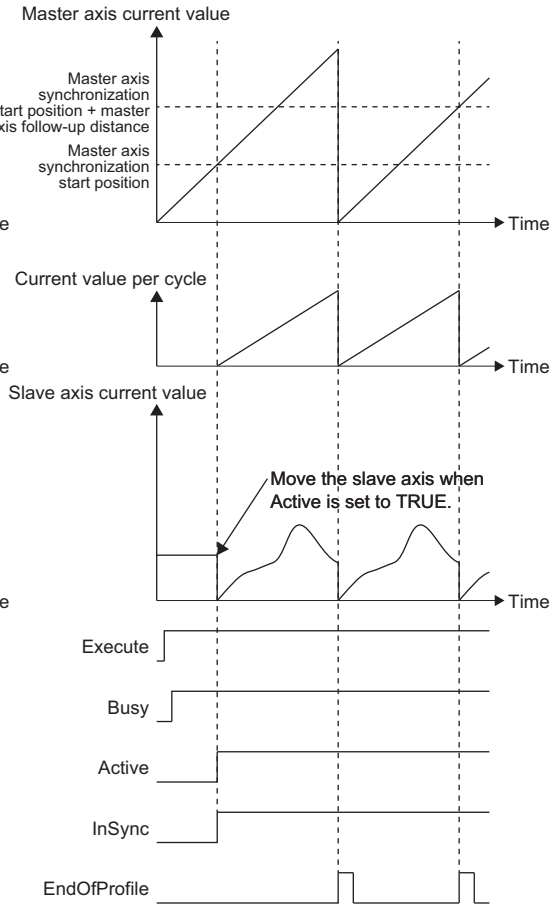
After Cam Operation Start (MC_CamIn) is executed, Controlling (Active) becomes TRUE if Current value per cycle (InputPerCycle) is synchronized, and In synchronization (InSync) becomes TRUE if Output value (OutputData) is synchronized.

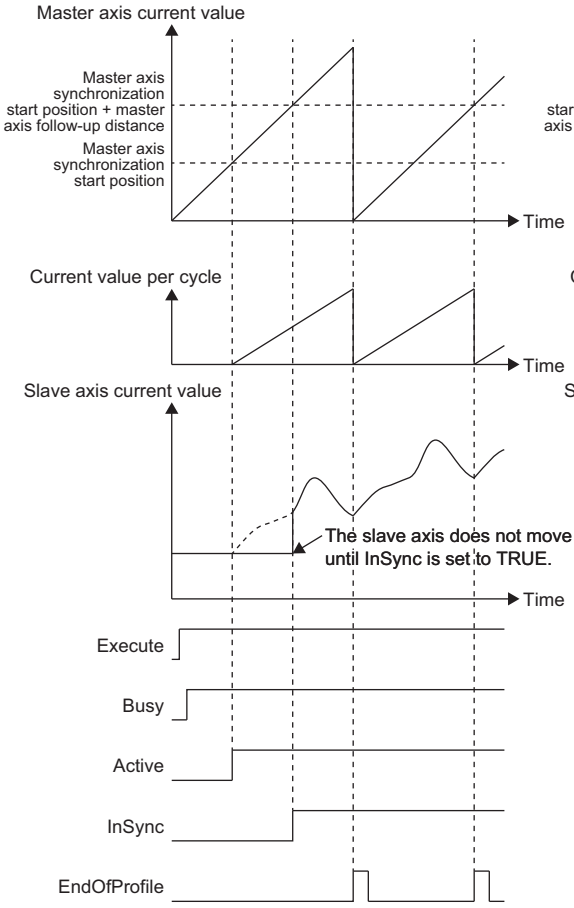
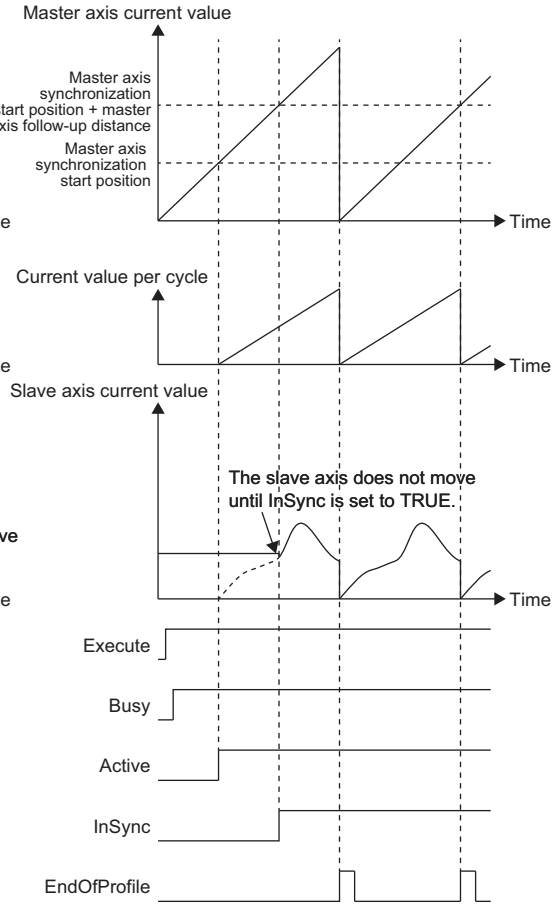
For examples of how each setting value is used, refer to "Cam operation" in the following manual.

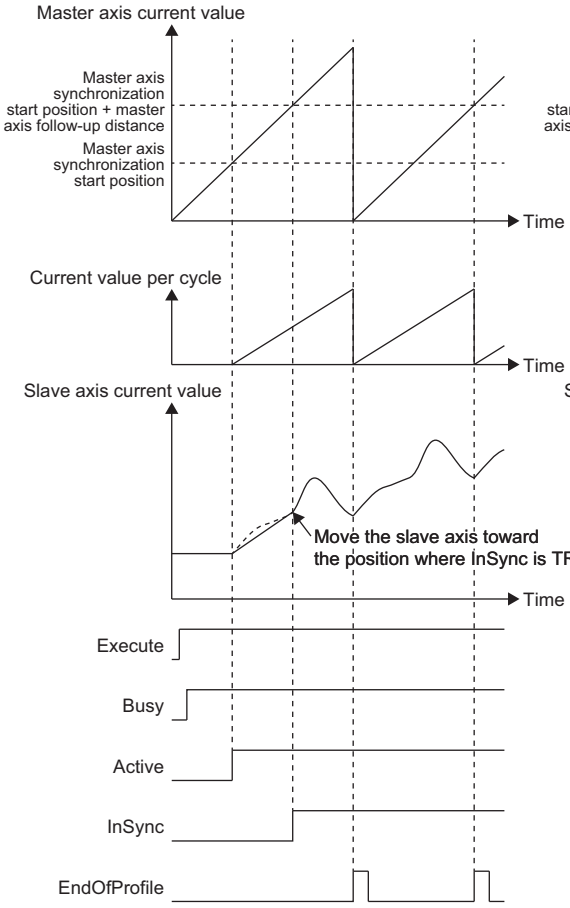
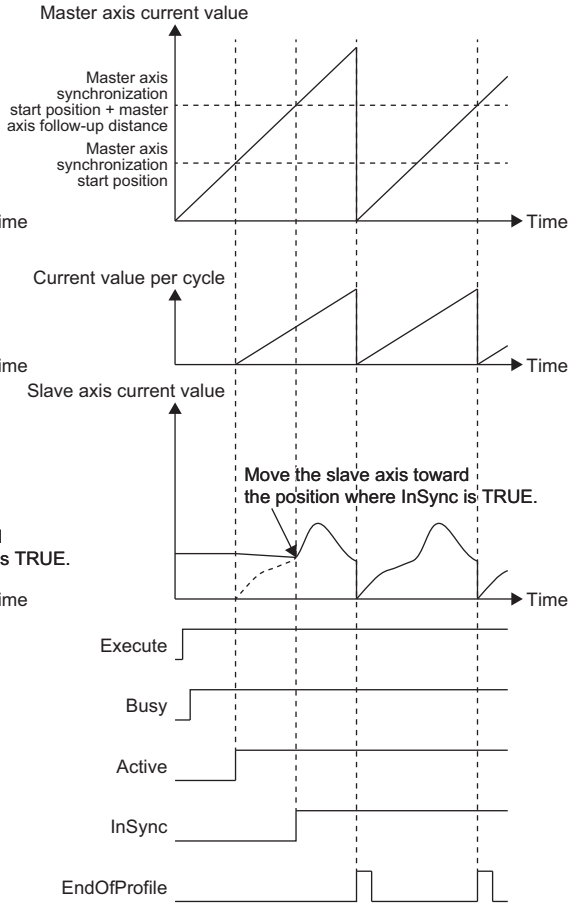
📖User's manual for the controller used

Depending on the setting of Slave axis absolute coordinate (SlaveAbsolute), the operation is as shown below.

Setting value	Description
0: Immediate (mcImmediate)	The slave axis moves after Execution command (Execute) is started. ■When slave axis absolute coordinate (SlaveAbsolute) is set to "FALSE: Relative coordinate"  ■When slave axis absolute coordinate (SlaveAbsolute) is set to "TRUE: Absolute coordinate" 

Setting value	Description
1: Absolute (mcAbsolute)	The slave axis moves after Execution command (Execute) is started and the master axis cumulative current position passes Master axis synchronization start position (MasterSyncPosition). Checking of whether the Master axis cumulative current position has passed the Master axis synchronization start position (MasterSyncPosition) begins at the point when the Slave axis status becomes "7: During synchronous operation (SynchronizedMotion)". ■When slave axis absolute coordinate (SlaveAbsolute) is set to "FALSE: Relative coordinate"  ■When slave axis absolute coordinate (SlaveAbsolute) is set to "TRUE: Absolute coordinate" 

Setting value	Description
2: Relative (mcRelative)	After the Execution command (Execute) is issued and the master axis cumulative current position passes the Master axis synchronization start position (MasterSyncPosition), the current value per cycle (InputPerCycle) is updated. Then, after the master axis cumulative current position moves by the Master axis follow-up distance (MasterStartDistance), the slave axis starts moving according to the cam data. Checking of whether the Master axis cumulative current position has passed the Master axis synchronization start position (MasterSyncPosition) begins at the point when the Slave axis status becomes "7: During synchronous operation (SynchronizedMotion)". ■When slave axis absolute coordinate (SlaveAbsolute) is set to "FALSE: Relative coordinate"  ■When slave axis absolute coordinate (SlaveAbsolute) is set to "TRUE: Absolute coordinate" 

Setting value	Description
3: Ramp-in (mcRampIn)	After the Execution command (Execute) is issued and the master axis cumulative current position passes the Master axis synchronization start position (MasterSyncPosition), the current value per cycle (InputPerCycle) is updated, and at the start of synchronization, the slave axis starts moving toward the slave axis position. Then, after the master axis cumulative current position moves by the Master axis follow-up distance (MasterStartDistance), the slave axis starts moving according to the cam data. Checking of whether the Master axis cumulative current position has passed the Master axis synchronization start position (MasterSyncPosition) begins at the point when the Slave axis status becomes "7: During synchronous operation (SynchronizedMotion)". ■When slave axis absolute coordinate (SlaveAbsolute) is set to "FALSE: Relative coordinate"  ■When slave axis absolute coordinate (SlaveAbsolute) is set to "TRUE: Absolute coordinate" 

■Master axis data source selection (MasterValueSource)

Slave axis (Slave) sets the position type of Master axis (Master) that will execute single axis synchronous control.
The operation may differ depending on the execution order of the FBs or if the operation cycles of Master axis (Master) and Slave axis (Slave) are different.
Operation is performed with the same value as the set value when specifying Actual value in Master axis data source selection (MasterValueSource) while an axis type other than real axis is set in Master axis (Master).

Setting value	Description
1: Set value (mcSetValue)	Uses the commanded position of the master axis in the previous operation cycle.
2: Actual value (mcActualValue)	Uses the actual position of the master axis in the previous operation cycle.
101: Latest set value (mcLatestSetValue)	Uses the commanded position of the master axis in the current operation cycle.
102: Latest actual value (mcLatestActualValue)	Uses the actual position of the master axis in the current operation cycle.

Point

When Master axis data source selection (MasterValueSource) is set to "1: Set value (mcSetValue)" or "101: Latest set value (mcLatestSetValue)" and the master axis becomes servo OFF by a servo alarm or forced stop, the amount of the value change may be too large. This can be prevented by setting Master axis data source selection (MasterValueSource) to "2: Actual value (mcActualValue)" or "102: Latest actual value (mcLatestActualValue)".

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following buffer modes are settable in MC_CamIn (Cam Operation Start).

The velocity for MC_CamIn (Cam Operation Start) follows the synchronizing FB (the master axis) and this velocity changes immediately based on the master axis velocity and the set cam data when the FB is switched. The switching condition for each mode is shown below.

Setting value	Description	Switching condition
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.	Always
1: Buffered(mcBuffered)	Buffers the next FB on the under-control FB. If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after completion of the under-control FB.	When Cam cycle completion (EndOfProfile) is TRUE

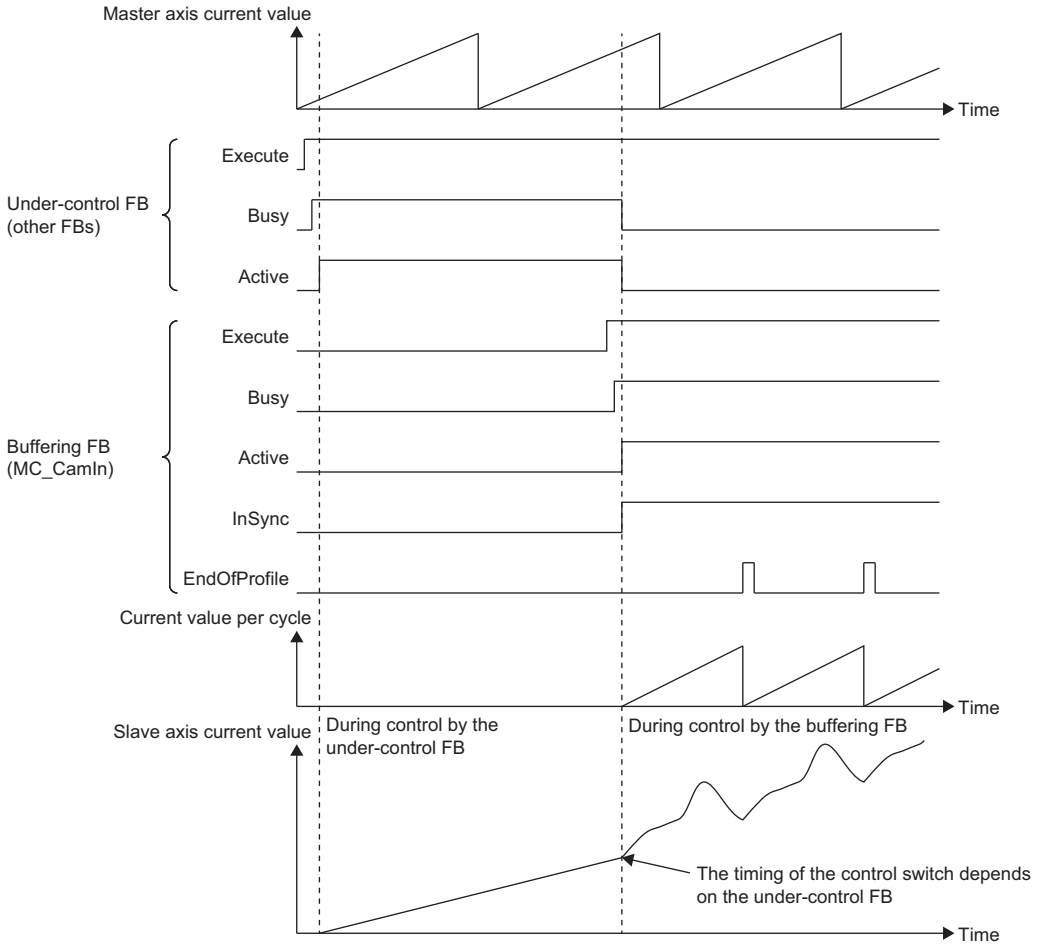
Point

For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

- Under-control FB and buffering FB (MC_CamIn (Cam Operation Start))

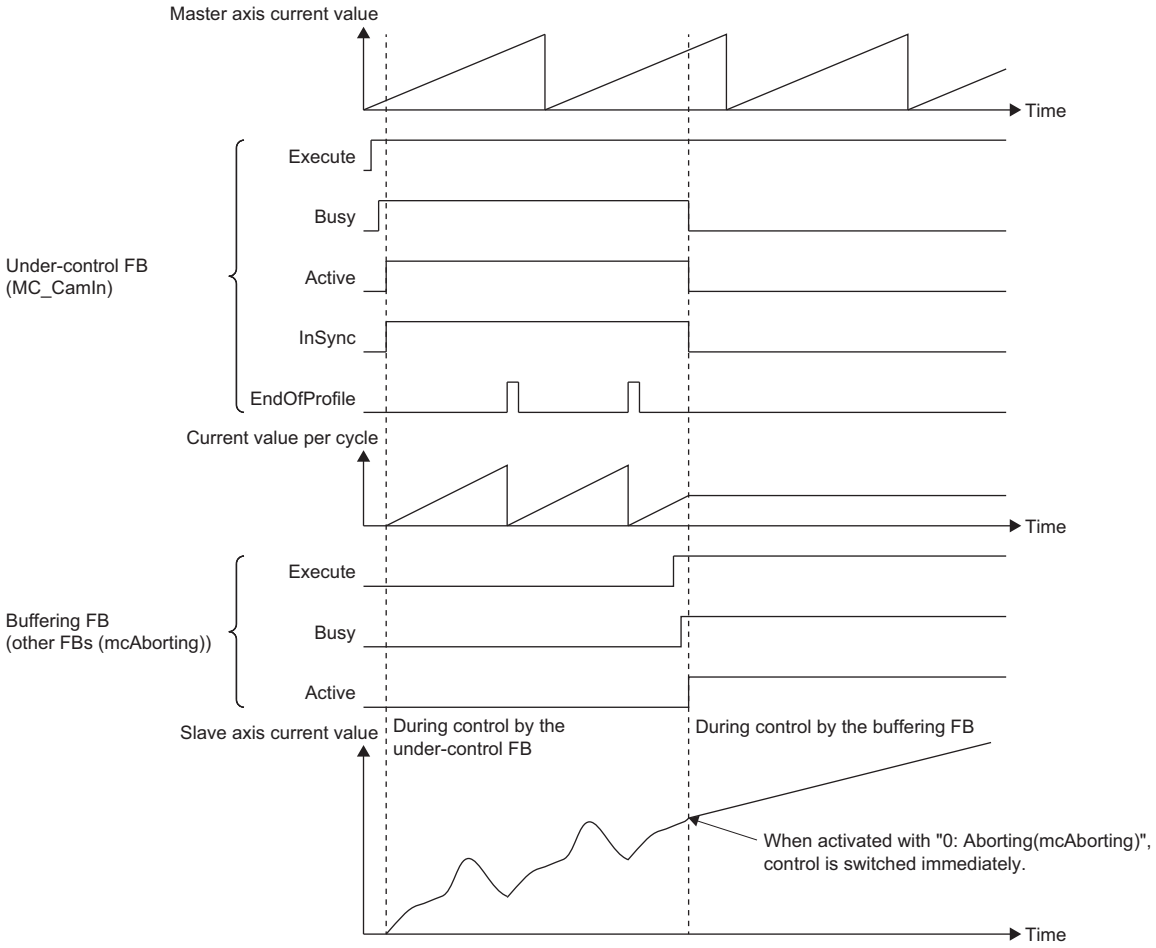
  If MC_CamIn (Cam Operation Start) is activated while the under-control FB is controlling the slave axis specified for MC_CamIn (Cam Operation Start), the control of the slave axis switches to MC_CamIn (Cam Operation Start). The timing of the control switch depends on the Buffer mode (BufferMode) set for the under-control FB and MC_CamIn (Cam Operation Start).



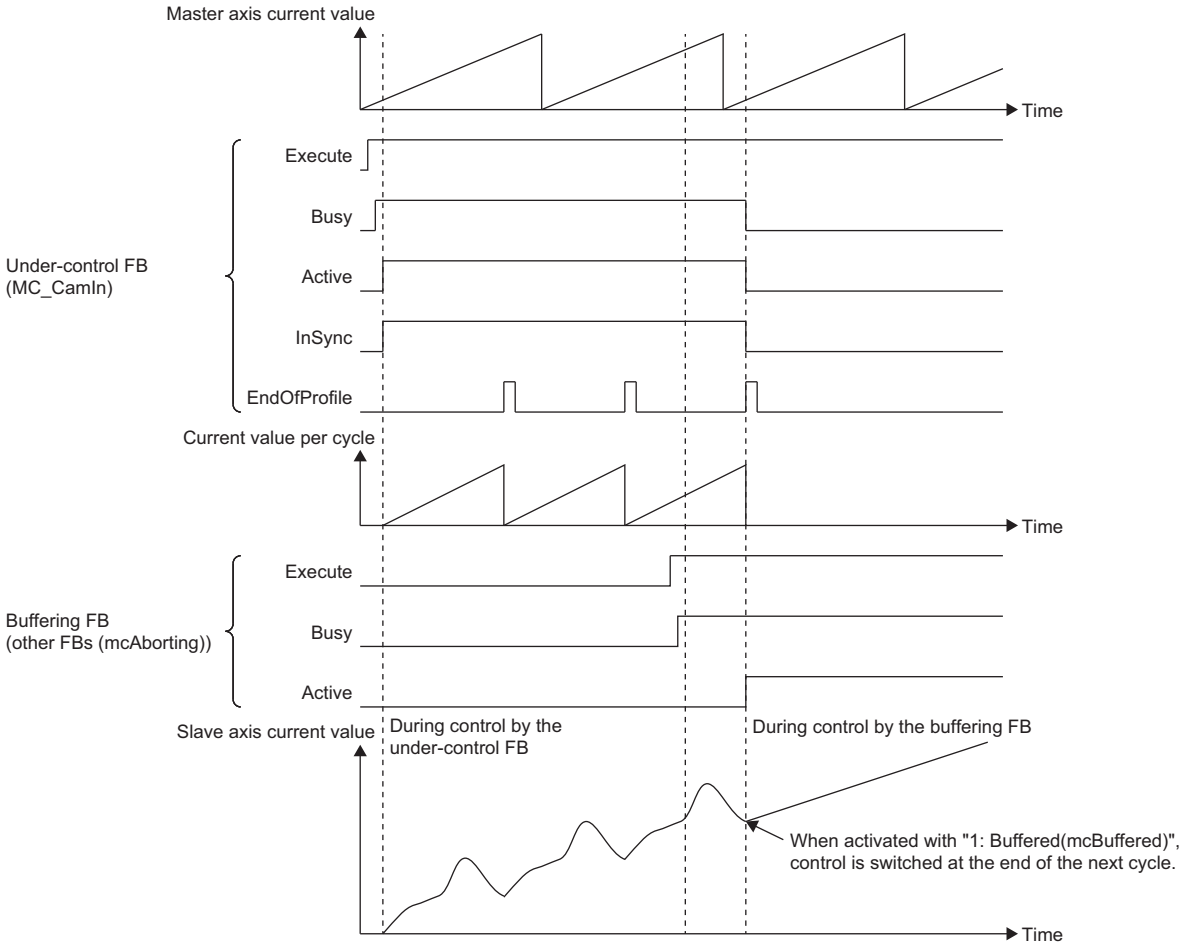
Point

The speed of a buffering FB (MC_CamIn (Cam Operation Start)) occurs after In synchronization (InSync) becomes TRUE.
When Start Mode (StartMode) is set to a value other than "0: Immediate (mcImmediate)", to prevent In synchronization (InSync) from becoming TRUE immediately after the FB starts, the velocity remains 0 until Executing (Busy), Controlling (Active), and In synchronization (InSync) all become TRUE.

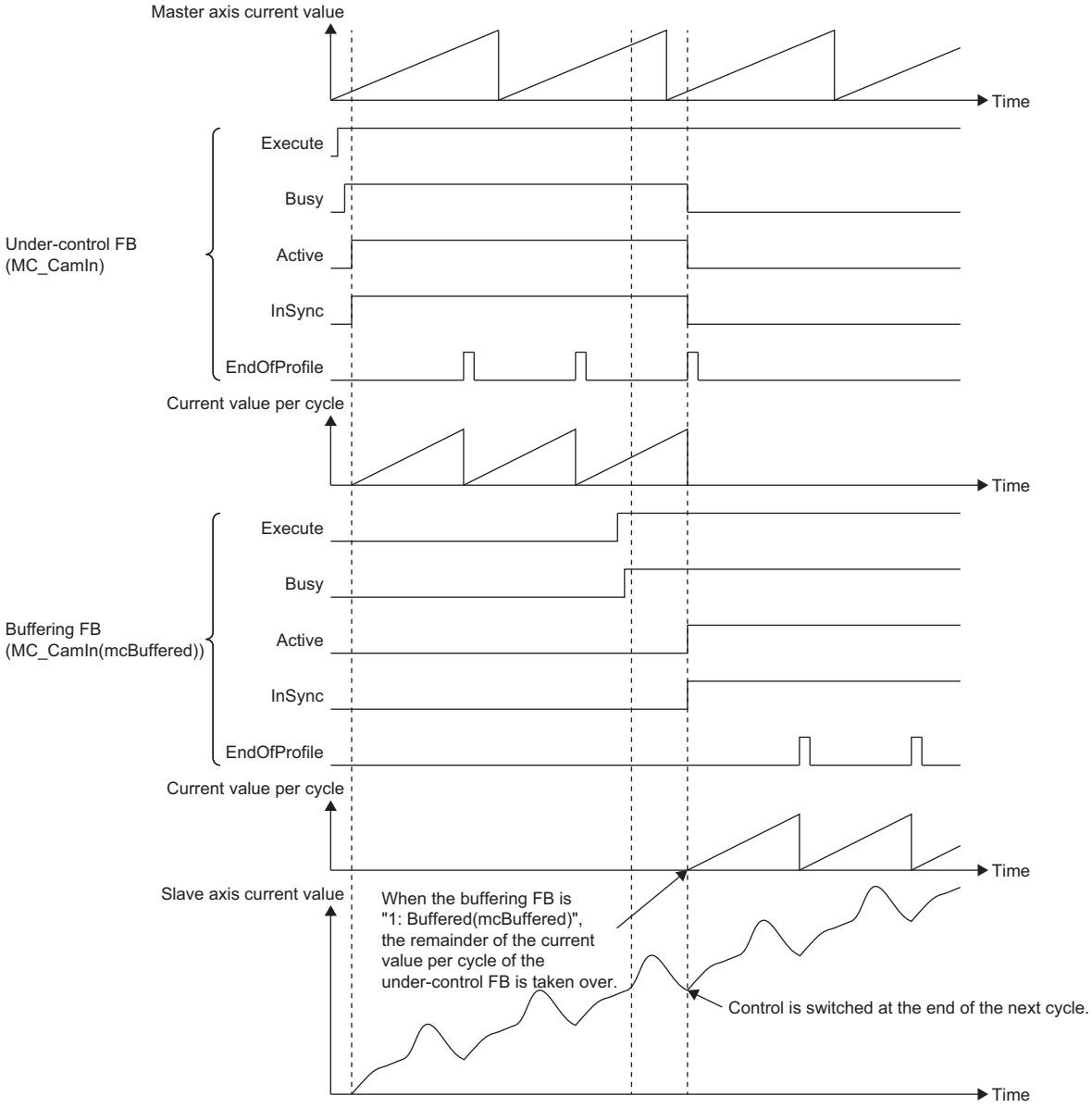
- Under-control FB (MC_CamIn (Cam Operation Start)) and buffering FB (Buffer mode (BufferMode)) set to "0: Aborting (mcAborting)"
  If another FB is activated for the slave axis specified for MC_CamIn (Cam Operation Start) while MC_CamIn (Cam Operation Start) is controlling the slave axis, control of the slave axis switches to the other FB. The timing of the control switch depends on the Buffer mode (BufferMode) of the other FB, and when "0: Aborting (mcAborting)" is specified, the control of the slave axis switches immediately.



- Under-control FB (MC_CamIn (Cam Operation Start)) and buffering FB (Buffer mode (BufferMode)) set to "1: Buffered (mcBuffered)"
  If another FB is activated for the slave axis specified for MC_CamIn (Cam Operation Start) while MC_CamIn (Cam Operation Start) is controlling the slave axis, control of the slave axis switches to the other FB. The timing of the control switch depends on the Buffer mode (BufferMode) of the other FB, and when "1: Buffered (mcBuffered)" is specified, the control of the slave axis switches at the end of the next cycle.



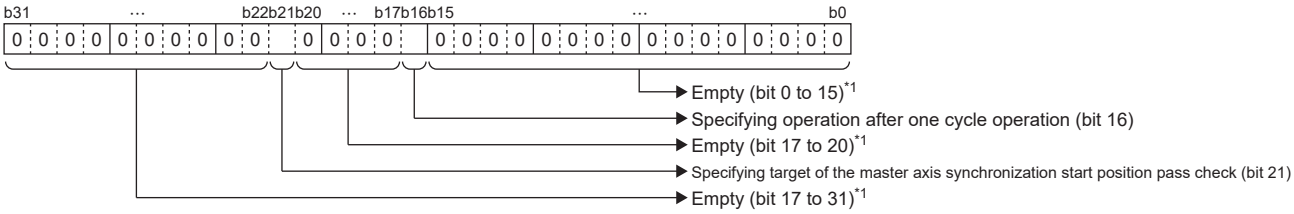
- Under-control FB (MC_CamIn (Cam Operation Start)) and buffering FB (MC_CamIn (Cam Operation Start))
  If another MC_CamIn (Cam Operation Start) is activated while MC_CamIn (Cam Operation Start) is controlling the slave axis, control of the slave axis switches. When Buffer mode (BufferMode) of the other MC_CamIn (Cam Operation Start) is set to "1: Buffered (mcBuffered)", adjust the starting point position so that the end and start points of the cam data are the same.



■Options (Options)

The function options for MC_CamIn (Cam Operation Start) are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.

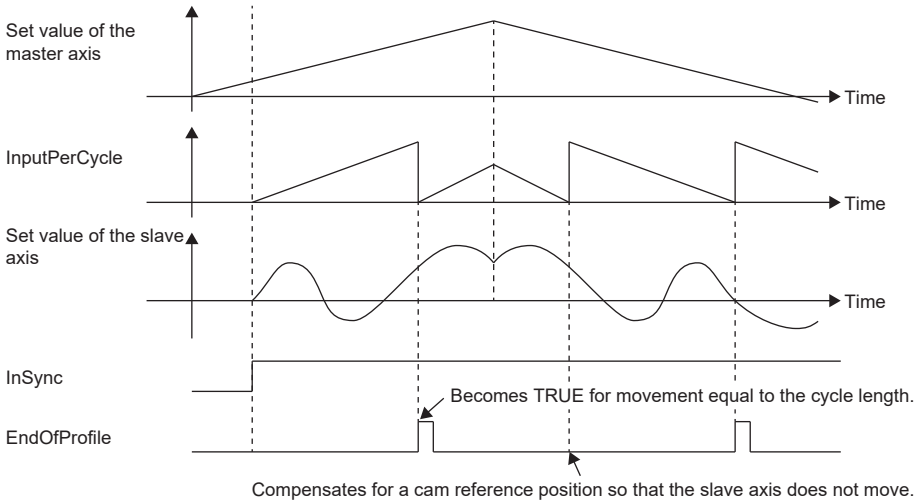


*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
16	Specifying operation after one cycle operation	The operation set is the one after one cycle operation when Periodic (Periodic) of MC_CamTableSelect (Cam Table Selection) is set to "FALSE: Non periodic". • 0: End • 1: Wait for restart
21	Specifying target of Master axis synchronization start position pass check	The target of the Master axis synchronization start position (MasterSyncPosition) pass check is set. • 0: Master axis cumulative current position, or where the feedback position has been converted to the cumulative position. • 1: Master axis set position, or the feedback position

■Cam cycle completion (EndOfProfile)

After Controlling (Active) becomes TRUE, this variable becomes TRUE only for one execution cycle of the POU (Program Organization Unit) that calls FB each time movement equal to the cycle length occurs.



■Current value per cycle (InputPerCycle)

The initial value is 0.0.

After Controlling (Active) becomes TRUE, the movement amount of Master axis (Master) is reflected as shown below.

- Current value per cycle = (Master axis cumulative movement amount - Master axis offset + Current value per cycle change value) MOD length per cycle.

The monitor value range is the following range of the length per cycle set in the cam table.

- 0 ≤ Current value per cycle (InputPerCycle) of MC_CamIn (Cam Operation Start) < (Length per cycle × Master axis scaling).

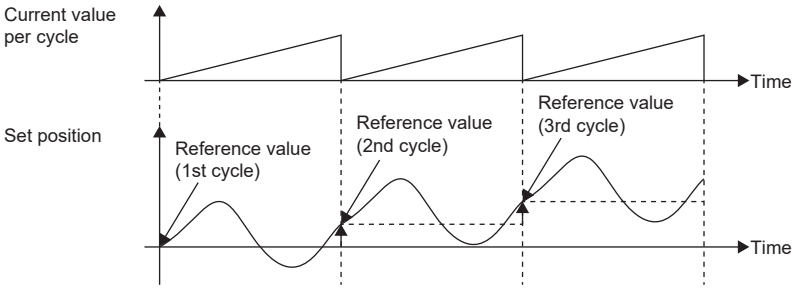
■Reference value (Reference)

When MC_CamIn (Cam Operation Start) is executed, the reference value is calculated from the cam position start (cam current value per cycle) based on the set position where In synchronization (InSync) becomes TRUE.

• For subtracting the initial stroke amount in the cam table, the formula for initial values are shown below.

Slave axis absolute coordinate (SlaveAbsolute)	Formula
FALSE (Relative coordinate)	Reference value = Slave axis position when synchronization starts - (Stroke value based on cam position (current value per cycle) when In synchronization (InSync) changes to TRUE + Initial stroke amount) × Slave axis scaling
TRUE (Absolute coordinate)	Reference value = The cumulative current position corresponding to the set position being 0 when In synchronization (InSync) changes to TRUE.

• The reference values used in two-way cam operation and feed cam operation are shown below.

Operation	Description
When Two-way cam operation, or Slave absolute coordinate (SlaveAbsolute) is "TRUE (Absolute Coordinate)"	The reference value is not updated.
When Feed cam operation as well as Slave axis absolute coordinate (SlaveAbsolute) are "FALSE (relative coordinate)"	The reference value is calculated as shown below. • Reference value = (Original reference value + (Stroke value at the last point - Stroke value at the 1st point) × Slave axis scaling) 

• The reference values are updated with the following timing.

Update timing	Reference value calculation formula
When Current value per cycle (InputPerCycle) going in the address increasing direction passes the last point of the cam table	Original reference value + ((Stroke value at the last point - Stroke value at the 1st point) × Slave axis scaling)
When Current value per cycle (InputPerCycle) going in the address decreasing direction passes the 1st point of the cam table	Original reference value - ((Stroke value at the last point - Stroke value at the 1st point) × Slave axis scaling)
When the current value per cycle is changed (Completion (Done) is TRUE for MCv_ChangeCycle (Current Value Change per Cycle))	Original reference value - (Stroke value corresponding to current value per cycle after change × Slave axis scaling)

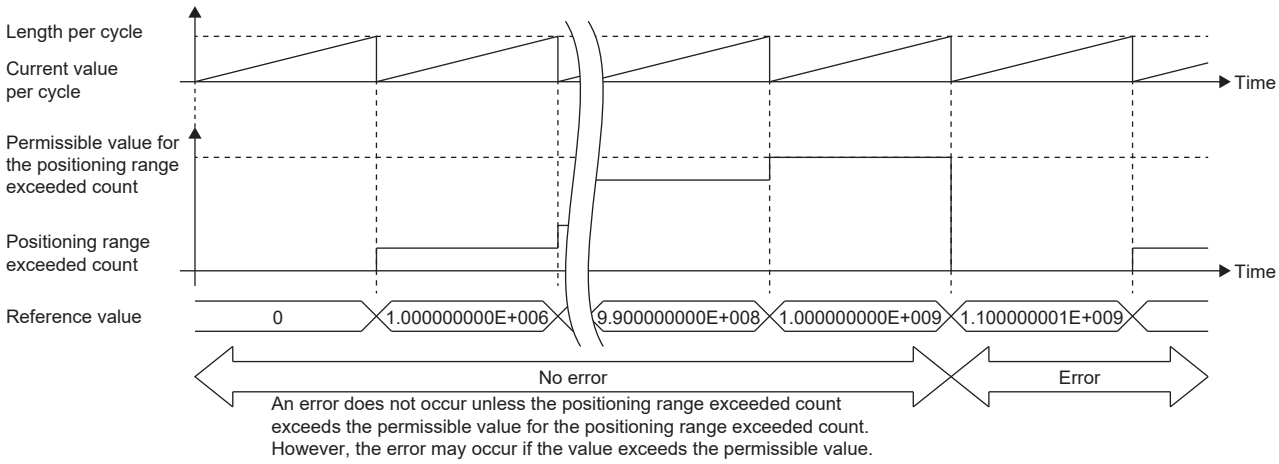
Restriction

<Reference value errors>

Performing any of the following operations may cause a reference value error to occur.

- If the number of TRUE outputs by Cam cycle completion (EndOfProfile) exceeds the permissible value after the FeedCam as well as In synchronization (InSync) become TRUE. *1
- If the FeedCam as well as the setting for the following control parameters contains a decimal.
  - (Stroke value at the last point - Stroke value at the first point)
  - Slave axis scaling
- If current value per cycle change is repeated with relative selection.

*1: While an error does not occur if the positioning range exceeded count is less than the permissible value for the positioning range exceeded count, an error may occur if the permissible value is exceeded. (Operation is continued regardless of whether or not a truncation error has occurred.)



If the reference value error becomes too large, clear the synchronization status or reset it by executing a current value change with absolute position specification.

The permissible value for the positioning range exceeded count is a variable value calculated with the following formula.

Permissible value for the positioning range exceeded count = Limit value for the positioning range exceeded count / ((Output value at the last point - Output value at the 1st point) /slave scaling)

*: The limit value for the positioning range exceeded count is the upper limit value of the positioning range, "10000000000".

(Example)

When the output value at the 1st point is "0", the output value at the last point is "1000000", and the output coefficient is "10", the permissible value for the positioning range exceeded count becomes "100000".

The monitor value range is shown below.

- Ring counter lower limit value of the slave axis ≤ Reference value < Ring counter upper limit value of the slave axis

■Output value (OutputData)

The default value is the initial stroke amount in the cam table.

The monitor value range is Lower limit value of the slave axis ring counter ≤ Output value < Upper limit value of the slave axis ring counter.

The update timing and calculation methods for Output value (OutputData) are shown below.

Update timing	Calculation method for Output value
While Controlling (Active) is TRUE and In synchronization (InSync) is FALSE	Reference value + (Initial stroke amount in the cam table + Slave axis offset) × Slave axis scaling When Start mode (StartMode) is set to Ramp-in (mcRampIn), the output value is calculated from the above position toward the slave axis position at the start of synchronization. For details, refer to the following. ☞ Page 1771 Start mode (StartMode)
While In synchronization (InSync) is TRUE	■When the interpolation method is specified to Specifying for each section or Spline interpolation Reference value + ((Stroke amount × Stroke ratio corresponding to current value per cycle) + Slave axis offset) × Slave axis scaling ■When the cam data format is Linear interpolation Reference value + ((Output value corresponding to current value per cycle) + Slave axis offset) × Slave axis scaling

Precautions

- When Master axis offset (MasterOffset), Slave axis offset (SlaveOffset), Master axis scaling (MasterScaling), Slave axis scaling (SlaveScaling), or Cam table ID (CamTableID) are changed, the slave axis may move suddenly at the start of control or a control change. This sudden movement may cause a shock to the equipment, so it is important to carefully check the setting values and the change timing.
- When monitoring the axis being cam controlled with the engineering tool, the position, velocity, or others are based on the unit setting of the axis. Units for the length per cycle and stroke amount set in the operation profile data are not used.

Program example

The following is a program example. The cam operation start command (bCamInStart) is set to TRUE, Axis 2 (Axis0002) is set as the master axis to be started in JOG operation, and cam operation of the slave axis (Axis0001) is performed according to the operation profile data (cam data 1).

• Setting

Item		Setting value
Data setting for JOG operation	Velocity	1000.0
	Acceleration	5000.0
	Deceleration	5000.0
	Jerk	0.0

■Axes

Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1
2	Axis0002	AXIS_REF	Axis 2

■Operation profile data

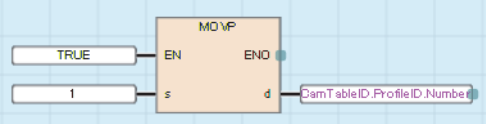
Profile ID	Label name	Data type	Comment
1	ProfileData0001	MC_CAM_REF	Cam data 1

■Labels used

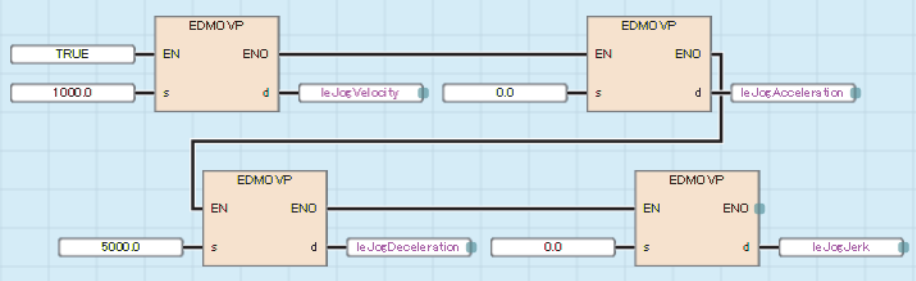
Label name	Data type	Comment
MC_CamIn_1	MC_CamIn	Cam Operation Start FB
bCamInStart	Bit	Cam operation start command
CamTableID	MC_CAM_ID	Cam table ID
bInSync	Bit	In synchronization
bCamBusy	Bit	Executing
bCamActive	Bit	Controlling
bCamCommandAborted	Bit	Abortion of execution
bCamError	Bit	Error
uwCamErrorID	Word [unsigned]/bit string [16-bit]	Error code
bEndOfProfile	Bit	Cam cycle completion
MCv_Jog_1	MCv_Jog	JOG FB
leJogVelocity	Double-precision real number	JOG velocity
leJogAcceleration	Double-precision real number	JOG acceleration
leJogDeceleration	Double-precision real number	JOG deceleration
leJogJerk	Double-precision real number	JOG jerk
bJogDone	Bit	Execution completion
bJogBusy	Bit	Executing
bJogActive	Bit	Controlling
bJogCommandAborted	Bit	Abortion of execution
bJogError	Bit	Error
uwJogErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

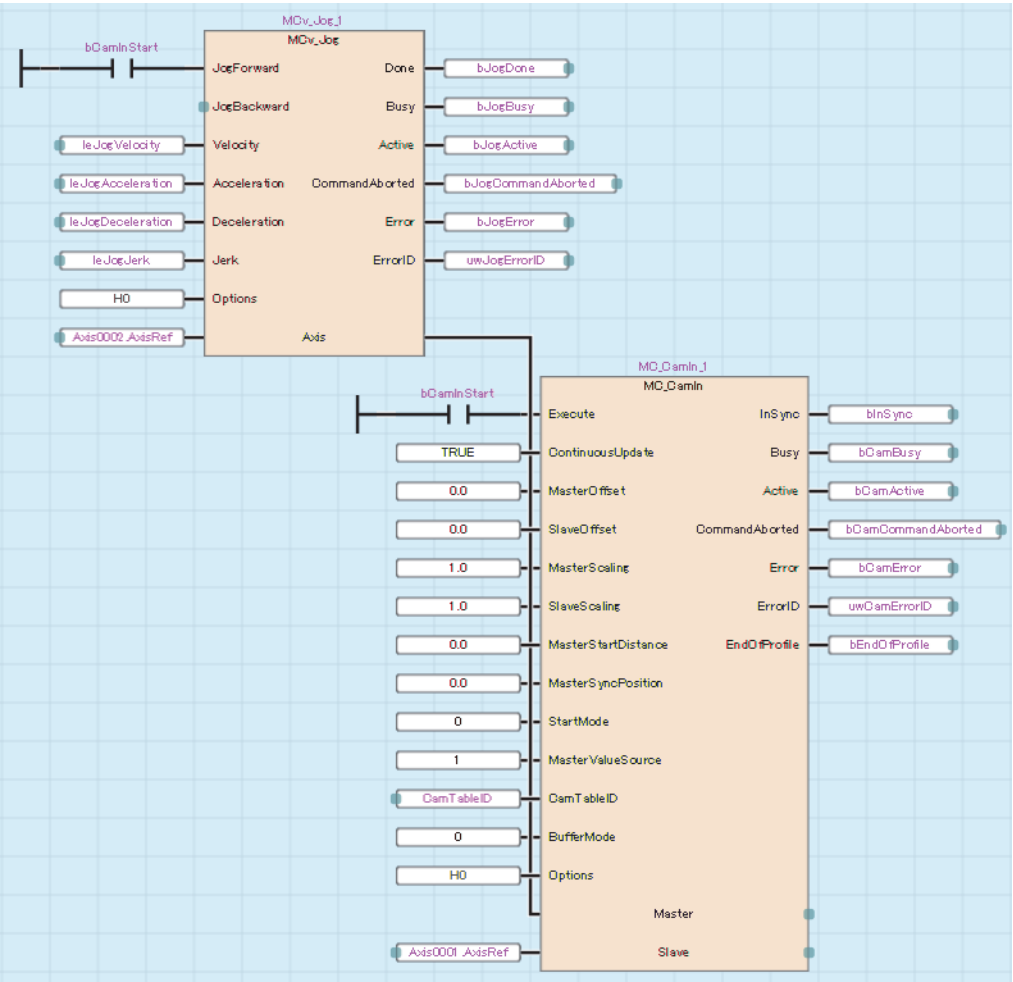
- Operation profile data (cam data) settings



- Data settings for JOG operation



- JOG operation/Cam operation start



■ST program

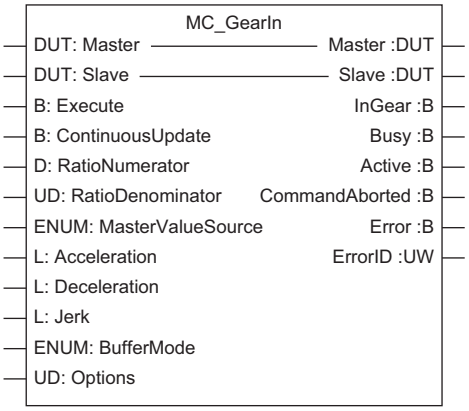
```
//-----Operation profile data (cam data) settings-----
CamTableID.ProfileID.Number:= 1;

//-----Data settings for JOG operation-----
leJogVelocity:= 1000.0;
leJogAcceleration:= 5000.0;
leJogDeceleration:= 5000.0;
leJogJerk:= 0.0;

//-----JOG-----
MCv_Jog_1(
   Axis:= Axis0002.AxisRef ,
   JogForward:= bCamInStart ,
   Velocity:= leJogVelocity ,
   Acceleration:= leJogAcceleration ,
   Deceleration:= leJogDeceleration ,
   Jerk:= leJogJerk ,
   Options:= H00000000 ,
   Done=> bJogDone ,
   Busy=> bJogBusy ,
   Active=> BJogActive ,
   CommandAborted=> bJogCommandAborted ,
   Error=> bJogError ,
   ErrorID=> uwJogErrorID
);

//-----Cam Operation Start-----
MC_CamIn_1(
   Master:= Axis0002.AxisRef ,
   Slave:= Axis0001.AxisRef ,
   Execute:= bCamInStart ,
   ContinuousUpdate:= TRUE ,
   MasterOffset:= 0.0 ,
   SlaveOffset:= 0.0 ,
   MasterScaling:= 1.0 ,
   SlaveScaling:= 1.0 ,
   MasterStartDistance:= 0.0 ,
   MasterSyncPosition:= 0.0 ,
   StartMode:= MC_START_MODE__mcImmediate ,
   MasterValueSource:= MC_SOURCE__mcSetValue ,
   CamTableID:= CamTableID ,
   BufferMode:= MC_BUFFER_MODE__mcAborting ,
   Options:= H00000000 ,
   InSync=> bInSync ,
   Busy=> bCamBusy ,
   Active=> bCamActive ,
   CommandAborted=> bCamCommandAborted ,
   Error=> bCamError ,
   ErrorID=> uwCamErrorID ,
   EndOfProfile=> bEndOfProfile
);
```

42.16 Gear Operation Start

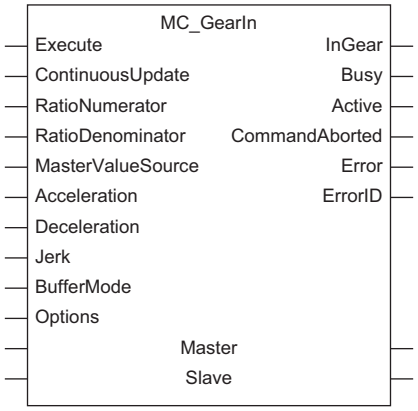
MC_GearIn

This FB starts gear operation according to the specified gear ratio.

Ladder



FBD/LD



ST

```
MC_GearIn(
  Master:= ?AXIS_REF? ,
  Slave:= ?AXIS_REF? ,
  Execute:= ?BOOL? ,
  ContinuousUpdate:= ?BOOL? ,
  RatioNumerator:= ?DINT? ,
  RatioDenominator:= ?DWORD? ,
  MasterValueSource:= ?INT? ,
  Acceleration:= ?LREAL? ,
  Deceleration:= ?LREAL? ,
  Jerk:= ?LREAL? ,
  BufferMode:= ?INT? ,
  Options:= ?DWORD? ,
  InGear=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Master	Master axis	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)
Slave	Slave axis	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_GearIn (Gear Operation Start) is executed.
Continuous Update	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	Whether to enable or disable continuous change of Gear ratio numerator (RatioNumerator), Gear ratio denominator (RatioDenominator), Acceleration (Acceleration), and Deceleration (Deceleration) is set. • FALSE: Disabled • TRUE: Enabled
RatioNumerator	Gear ratio numerator	DINT	At start/Retrigger possible/ Continuous update possible	-2147483648 to 2147483647	1	The numerator value of Master axis (Master) is set. For details, refer to the following. ☞ Page 1792 Gear ratio numerator (RatioNumerator)/Gear ratio denominator (RatioDenominator)
RatioDenominator	Gear ratio denominator	DWORD(UDINT)	At start/Retrigger possible/ Continuous update possible	1 to 2147483647	1	The denominator value of Master axis (Master) is set. For details, refer to the following. ☞ Page 1792 Gear ratio numerator (RatioNumerator)/Gear ratio denominator (RatioDenominator)

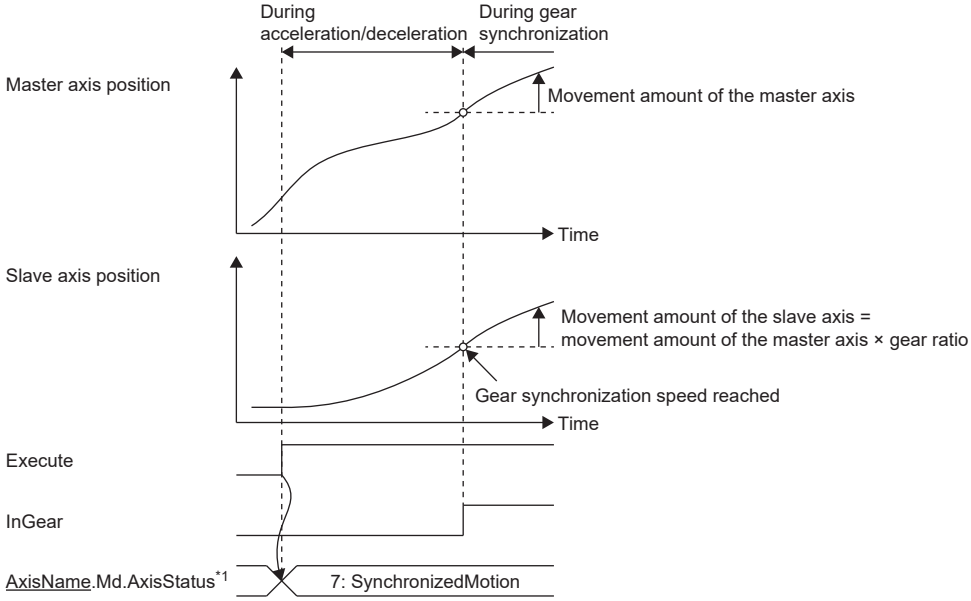
Input variable	Name	Data type	Input	Valid range	Default value	Description
MasterValue Source	Master axis data source selection	INT (MC_SOURCE)	At start	1, 2, 101, 102	1	The data source of Master axis (Master) is set. • 1: Set value (mcSetValue) • 2: Actual value (mcActualValue) • 101: Latest set value (mcLatestSetValue) • 102: Latest actual value (mcLatestActualValue) For details, refer to the following. ☞ Page 1793 Master axis data source selection (MasterValueSource)
Acceleration	Acceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The acceleration used for accelerating is set. When the target velocity is reached, Gear ratio reached (InGear) becomes TRUE and Slave axis (Slave) is controlled using a velocity equal to the velocity of Master axis (Master) converted by the gear ratio.
Deceleration	Deceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration used for decelerating is set. When the target velocity is reached, Gear ratio reached (InGear) becomes TRUE and Slave axis (Slave) is controlled using a velocity equal to the velocity of Master axis (Master) converted by the gear ratio.
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	The jerk at acceleration/deceleration start is set.
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0, 1	0	The buffer mode is set. • 0: Aborting(mcAborting) • 1: Buffered(mcBuffered) For details, refer to the following. ☞ Page 1793 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	Set "00000000H". *: If a value other than "00000000H" is set, Out of Options Range (error code: 1ABBH) occurs.

■Output variables

Output variable	Name	Data type	Default value	Description
InGear	Gear ratio reached	BOOL	FALSE	This variable becomes TRUE when the gear synchronization speed is reached.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_GearIn (Gear Operation Start) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE when Slave axis (Slave) is being controlled.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MC_GearIn (Gear Operation Start) is aborted. The variable becomes FALSE when Execution command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

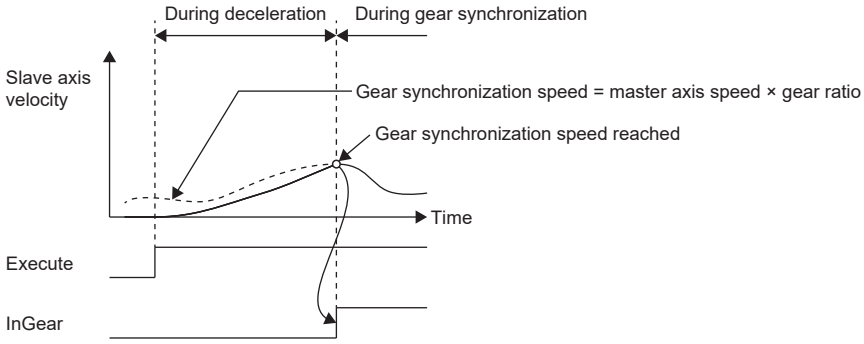
Processing details

- This FB sets Gear ratio numerator (RatioNumerator), Gear ratio denominator (RatioDenominator), Master axis data source selection (MasterValueSource), Acceleration (Acceleration), Deceleration (Deceleration), Jerk (Jerk), and Buffer mode (BufferMode), and starts the gear operation.
- To stop the operation, perform MC_Stop (Forced Stop).
- After executing this FB, Slave axis (Slave) performs acceleration/deceleration until reaching the target velocity, the value of which is the velocity of Master axis (Master) converted by the gear ratio. After reaching the target velocity, Gear ratio reached (InGear) becomes TRUE, and Slave axis (Slave) is controlled using a velocity equal to the velocity of Master axis (Master) converted by the gear ratio.



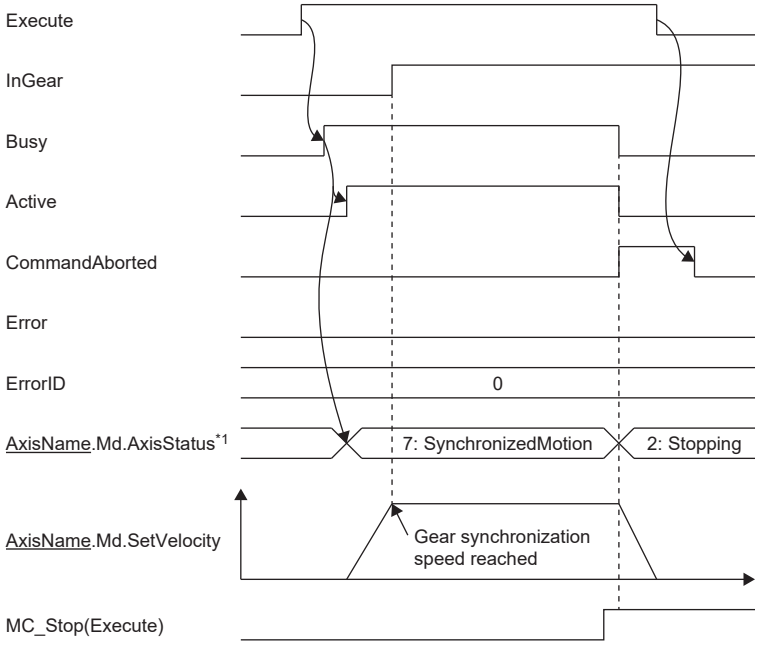
*1 Axis status of Slave axis (Slave)

- When the velocity of Master axis (Master) is changed during acceleration/deceleration, the gear synchronization speed is also updated.



■Timing chart

• When the processing is completed successfully



*1 Axis status of Slave axis (Slave)

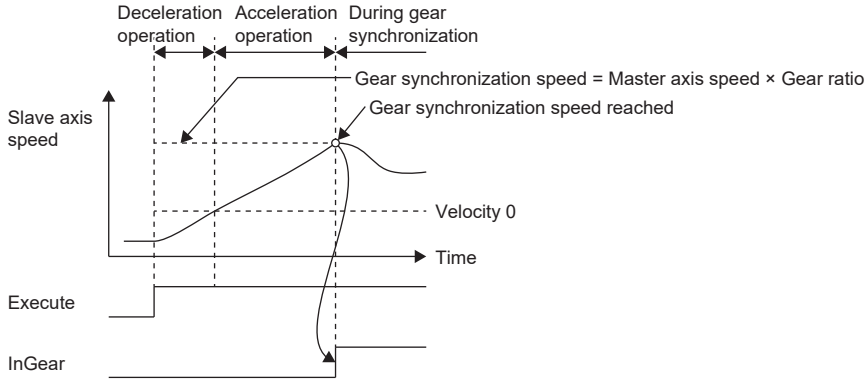
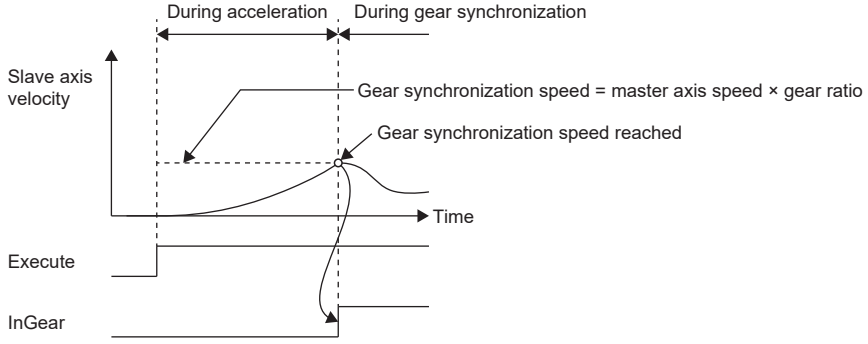
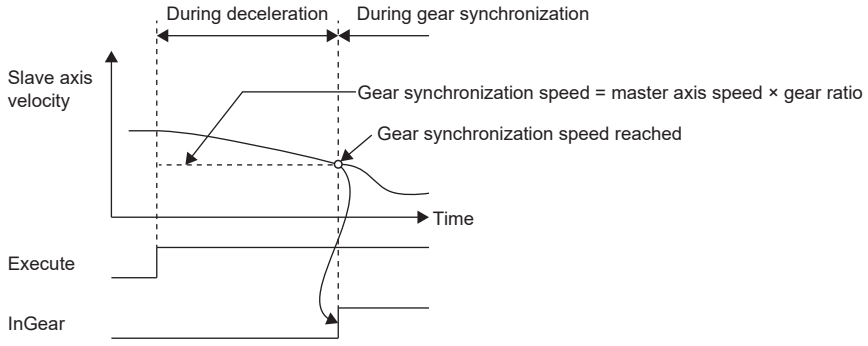
• When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Acceleration/deceleration until reaching the gear synchronization speed

- The acceleration/deceleration method for MC_GearIn (Gear Operation Start) is the acceleration/deceleration specification method.
- Depending on the velocity and target velocity of the slave axis (Slave) at operation start, one of the following 4 types of acceleration/deceleration is performed. The type of acceleration/deceleration will not change even if the target velocity is updated during the acceleration/deceleration.

Slave axis (Slave) velocity and target velocity	Acceleration/deceleration
Slave axis (Slave) velocity < 0 < Gear synchronization speed Alternatively, Slave axis (Slave) velocity > 0 > Gear synchronization speed	Deceleration is performed to velocity 0, then acceleration is performed from velocity 0 to the gear synchronization speed. 
Slave axis (Slave) velocity (Absolute value) < Gear synchronization speed (Absolute value)	Acceleration is performed until reaching the gear synchronization speed. 
Slave axis (Slave) velocity (Absolute value) > Gear synchronization speed (Absolute value)	Acceleration is performed until reaching the gear synchronization speed. 
Slave axis (Slave) velocity = Gear synchronization speed	Acceleration/deceleration is not performed. After operation is started, Gear ratio reached (InGear) becomes TRUE immediately and Slave axis (Slave) is controlled using a velocity equal to the velocity of Master axis (Master) converted by the gear ratio.

■Gear ratio numerator (RatioNumerator)/Gear ratio denominator (RatioDenominator)

Gear ratio numerator (RatioNumerator) and Gear ratio denominator (RatioDenominator) are set together.

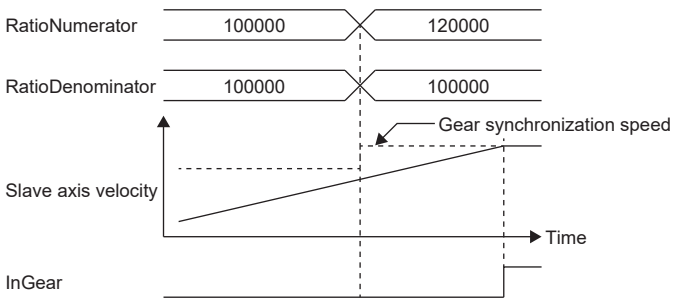
Gear ratio numerator (RatioNumerator) sets the numerator value for converting and transmitting the velocity of Master axis (Master).

Gear ratio denominator (RatioDenominator) sets the denominator value for converting and composing the movement amount of Master axis (Master).

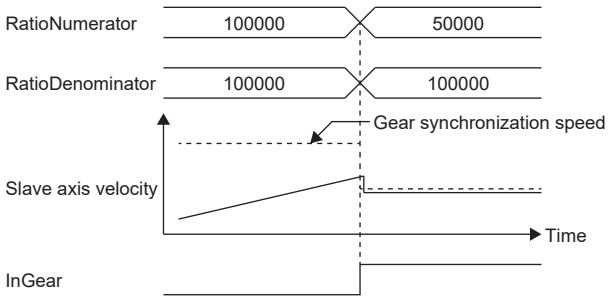
The velocity is converted as shown below.

$$\text{Master axis (Master) velocity after conversion} = \text{master axis (Master) velocity before conversion} \times \frac{\text{Gear ratio numerator (RatioNumerator)}}{\text{Gear ratio denominator (RatioDenominator)}}$$

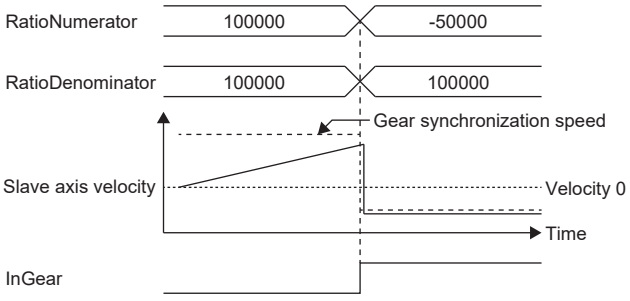
- Set a positive value in Gear ratio denominator (RatioDenominator).
- The velocity direction can be reversed and transmitted by setting a negative value in Ratio numerator (RatioNumerator). The gear synchronization speed is changed if the gear ratio is changed during acceleration/deceleration. Therefore, the velocity of Slave axis (Slave) is changed directly and the gear operation may start because of the change of the gear synchronization speed.
- The velocity of Master axis (Master) after conversion can be set to "0" by setting the value of Gear ratio numerator (RatioNumerator) to "0".
- The target velocity is changed if the gear ratio is changed during acceleration/deceleration. Therefore, changing the target velocity may directly change the velocity of Slave axis (Slave) and start the gear operation.
  - Changing the gear ratio during acceleration/deceleration
    - Gear ratio increase (set to a value higher than the velocity of Slave axis (Slave))



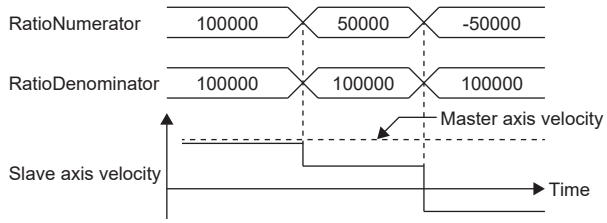
    - Gear ratio decrease (set to a value higher than the velocity of Slave axis (Slave))



    - Gear ratio direction reverse (set the gear synchronization speed to a value lower than "0")



- When the gear ratio is changed during gear synchronization, the velocity of Slave axis (Slave) is changed directly.
  - Gear ratio change during gear synchronization



■Master axis data source selection (MasterValueSource)

Slave axis (Slave) sets the position type of Master axis (Master) that will execute single axis synchronous control.
The operation may differ depending on the execution order of the FBs or if the operation cycles of Master axis (Master) and Slave axis (Slave) are different.
Operation is performed with the same value as the set value when specifying Actual value in Master axis data source selection (MasterValueSource) while an axis type other than real axis is set in Master axis (Master).

Setting value	Description
1: Set value (mcSetValue)	Uses the commanded position of the master axis in the previous operation cycle.
2: Actual value (mcActualValue)	Uses the actual position of the master axis in the previous operation cycle.
101: Latest set value (mcLatestSetValue)	Uses the commanded position of the master axis in the current operation cycle.
102: Latest actual value (mcLatestActualValue)	Uses the actual position of the master axis in the current operation cycle.

Point

When Master axis data source selection (MasterValueSource) is set to "1: Set value (mcSetValue)" or "101: Latest set value (mcLatestSetValue)" and the master axis becomes servo OFF by a servo alarm or forced stop, the amount of the value change may be too large. This can be prevented by setting Master axis data source selection (MasterValueSource) to "2: Actual value (mcActualValue)" or "102: Latest actual value (mcLatestActualValue)".

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.
The following buffer modes are settable in MC_GearIn (Gear Operation Start).

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.
1: Buffered(mcBuffered)	Buffers the next FB on the under-control FB. If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after completion of the under-control FB.

Point

For details on multiple start (buffer mode), refer to the following.
User's manual for the controller used

Precautions

- When the gear ratio is changed, the velocity of the slave axis is changed directly. To smooth the speed change, use the buffer mode together with MCv_SmoothingFilter (Smoothing Filter).
- When the velocity of Master axis (Master) after conversion is over the acceleration/deceleration time upper limit value during acceleration/deceleration operation, Acceleration Time Limit Over Warning (event code: 00D04H) or Deceleration Time Limit Over Warning (event code: 00D05H) will occur, the acceleration/deceleration operation is stopped, and the operation is restarted and continued at the velocity in the warning detection. If the warnings shown above occur, adjust the velocity of Master axis (Master) after conversion, Acceleration (Acceleration), and Deceleration (Deceleration) so that the setting for each does not exceed the acceleration/deceleration upper limit value.

Program example

The following is a program example. The gear operation start command (bGearInStart) is set to TRUE, Axis 1 (Axis0001) to be started in JOG operation is set as the master axis, and gear operation of the slave axis (Axis0002) is performed according to the specified gear ratio.

• Setting

Item		Setting value
Data settings for gear operation	Acceleration	5000.0
	Deceleration	5000.0
	Jerk	0.0
Data settings for JOG operation	Velocity	1000.0
	Acceleration	5000.0
	Deceleration	5000.0
	Jerk	0.0

■Axes

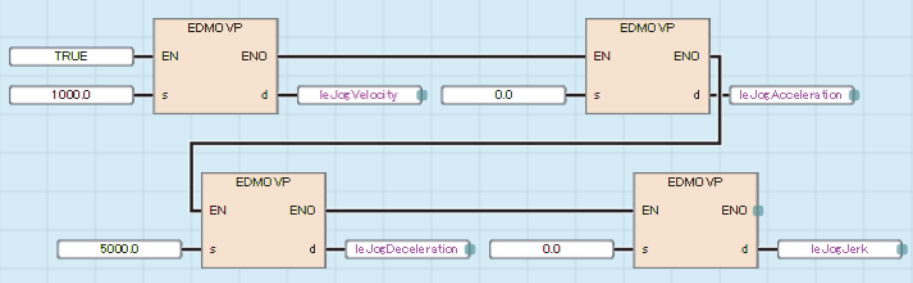
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1
2	Axis0002	AXIS_REF	Axis 2

■Labels used

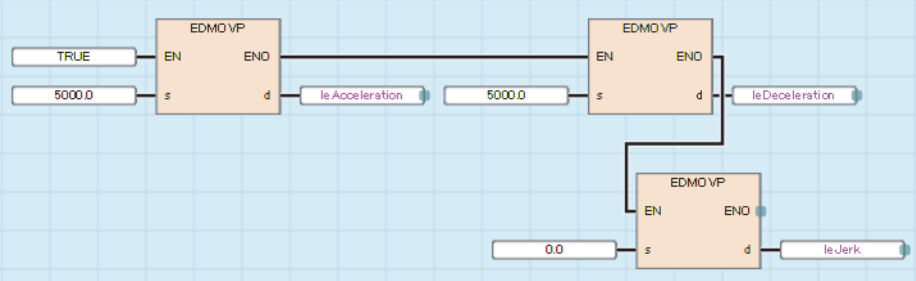
Label name	Data type	Comment
MC_GearIn_1	MC_GearIn	Gear Operation Start FB
bGearInStart	Bit	Gear operation start command
leAcceleration	Double-precision real number	Acceleration
leDeceleration	Double-precision real number	Deceleration
leJerk	Double-precision real number	Jerk
bInGear	Bit	In synchronization
bGearBusy	Bit	Executing
bGearActive	Bit	Controlling
bGearCommandAborted	Bit	Abortion of execution
bGearError	Bit	Error
uwGearErrorID	Word [unsigned]/bit string [16-bit]	Error code
MCv_Jog_1	MCv_Jog	JOG FB
leJogVelocity	Double-precision real number	JOG velocity
leJogAcceleration	Double-precision real number	JOG acceleration
leJogDeceleration	Double-precision real number	JOG deceleration
leJogJerk	Double-precision real number	JOG jerk
bJogDone	Bit	Execution completion
bJogActive	Bit	Controlling
bJogCommandAborted	Bit	Abortion of execution
bJogError	Bit	Error
uwJogErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

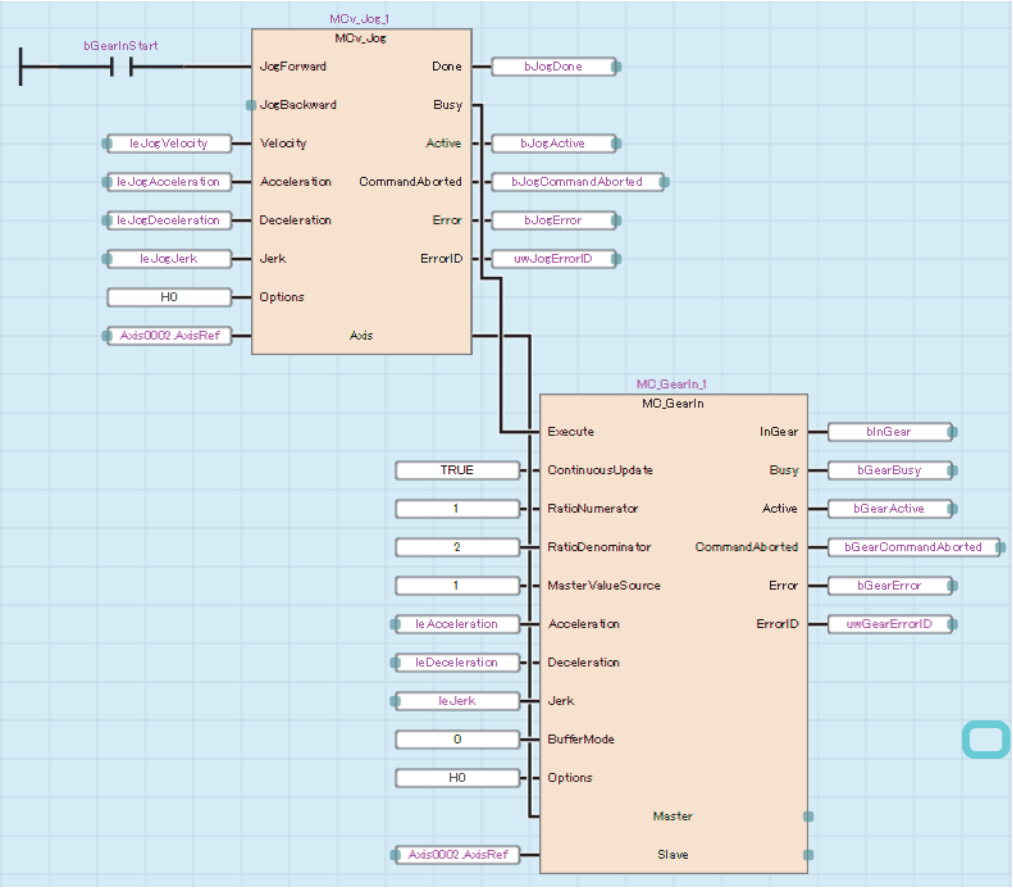
• Data settings for JOG operation



• Data settings for gear operation



• JOG/Gear Operation Start



■ST program

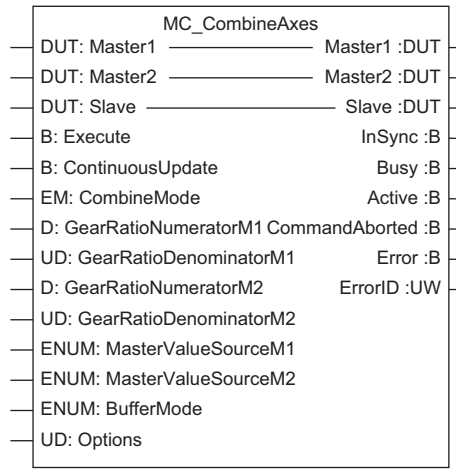
```
//-----Data settings for JOG operation-----
leJogVelocity:= 1000.0;
leJogAcceleration:= 5000.0;
leJogDeceleration:= 5000.0;
leJogJerk:= 0.0;

//-----Data settings for gear operation-----
leAcceleration:= 5000.0;
leDeceleration:= 5000.0;
leJerk:= 0.0;

//-----JOG-----
MCv_Jog_1(
   Axis:= Axis0001.AxisRef ,
   JogForward:= bGearInstart ,
   Velocity:= leJogVelocity ,
   Acceleration:= leJogAcceleration ,
   Deceleration:= leJogDeceleration ,
   Jerk:= leJogJerk ,
   Options:= H00000000 ,
   Done=> bJogDone ,
   Busy=> bJogBusy ,
   Active=> BJogActive ,
   CommandAborted=> bJogCommandAborted ,
   Error=> bJogError ,
   ErrorID=> uwJogErrorID
);

//-----Gear Operation Start-----
   MC_GearIn_1(
   Master:= Axis0001.AxisRef ,
   Slave:= Axis0002.AxisRef ,
   Execute:= bJogBusy ,
   ContinuousUpdate:= TRUE ,
   RatioNumerator:= 1 ,
   RatioDenominator:= 2 ,
   MasterValueSource:= MC_SOURCE__mcSetValue ,
   Acceleration:= leAcceleration ,
   Deceleration:= leDeceleration ,
   Jerk:= leJerk ,
   BufferMode:= MC_BUFFER_MODE__mcAborting ,
   Options:= H00000000 ,
   InGear=> bInGear ,
   Busy=> bGearBusy ,
   Active=> bGearActive ,
   CommandAborted=> bGearCommandAborted ,
   Error=> bGearError ,
   ErrorID=> uwGearErrorID
);
```

42.17 Addition/Subtraction Positioning

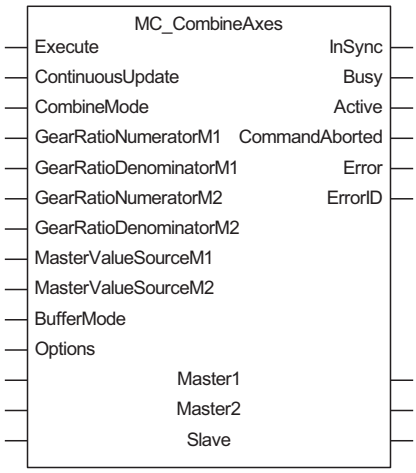
MC_CombineAxes

This FB adds or subtracts the movement amounts of the two specified master axes and performs positioning using said value as the commanded position.

Ladder



FBD/LD



ST

```
MC_CombineAxes(
  Master1:= ?AXIS_REF? ,
  Master2:= ?AXIS_REF? ,
  Slave:= ?AXIS_REF? ,
  Execute:= ?BOOL? ,
  ContinuousUpdate:= ?BOOL? ,
  CombineMode:= ?INT? ,
  GearRatioNumeratorM1:= ?DINT? ,
  GearRatioDenominatorM1:= ?DWORD? ,
  GearRatioNumeratorM2:= ?DINT? ,
  GearRatioDenominatorM2:= ?DWORD? ,
  MasterValueSourceM1:= ?INT? ,
  MasterValueSourceM2:= ?INT? ,
  BufferMode:= ?INT? ,
  Options:= ?DWORD? ,
  InSync=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Master1	Master axis 1	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)
Master2	Master axis 2	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)
Slave	Slave axis	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MC_CombineAxes (Addition/Subtraction Positioning) is executed.
Continuous Update	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	Whether to enable or disable continuous change of Addition/subtraction method selection (CombineMode), Master axis 1 gear ratio numerator (GearRatioNumeratorM1), Master axis 1 gear ratio denominator (GearRatioDenominatorM1), Master axis 2 gear ratio numerator (GearRatioNumeratorM2), and Master axis 2 gear ratio denominator (GearRatioDenominatorM2) is set. • FALSE: Disabled • TRUE: Enabled

Input variable	Name	Data type	Input	Valid range	Default value	Description
CombineMode	Addition/subtraction method selection	INT (MC_COMBINE_MODE)	At start/Retrigger possible/ Continuous update possible	0, 1	1	A method used to combine the movement amounts of Master axis 1 (Master1) and Master axis 2 (Master2) is set. • 0: Add positions of 2 input axes (mcAddAxes) • 1: Subtract positions of 2 input axes (mcSubAxes) For details, refer to the following. ☞ Page 1800 Addition/subtraction method selection (CombineMode)
GearRatioNumeratorM1	Master axis 1 gear ratio numerator	DINT	At start/Retrigger possible/ Continuous update possible	-2147483648 to 2147483647	1	The numerator value of Master axis 1 (Master1) is set. For details, refer to the following. ☞ Page 1801 Master axis 1 gear ratio numerator (GearRatioNumeratorM1)/Master axis 1 gear ratio denominator (GearRatioDenominatorM1)
GearRatioDenominatorM1	Master axis 1 gear ratio denominator	DWORD(UDINT)	At start/Retrigger possible/ Continuous update possible	1 to 2147483647	1	The denominator value of Master axis 1 (Master1) is set. For details, refer to the following. ☞ Page 1801 Master axis 1 gear ratio numerator (GearRatioNumeratorM1)/Master axis 1 gear ratio denominator (GearRatioDenominatorM1)
GearRatioNumeratorM2	Master axis 2 gear ratio numerator	DINT	At start/Retrigger possible/ Continuous update possible	-2147483648 to 2147483647	1	The numerator value of Master axis 2 (Master2) is set. For details, refer to the following. ☞ Page 1801 Master axis 2 gear ratio numerator (GearRatioNumeratorM2)/Master axis 2 gear ratio denominator (GearRatioDenominatorM2)
GearRatioDenominatorM2	Master axis 2 gear ratio denominator	DWORD(UDINT)	At start/Retrigger possible/ Continuous update possible	1 to 2147483647	1	The denominator value of Master axis 2 (Master2) is set. For details, refer to the following. ☞ Page 1801 Master axis 2 gear ratio numerator (GearRatioNumeratorM2)/Master axis 2 gear ratio denominator (GearRatioDenominatorM2)
MasterValueSourceM1	Master axis 1 data source selection	INT (MC_SOURCE)	At start	1, 2, 101, 102	1	The data source of Master axis 1 (Master1) is set. • 1: Set value (mcSetValue) • 2: Actual value (mcActualValue) • 101: Latest set value (mcLatestSetValue) • 102: Latest actual value (mcLatestActualValue) For details, refer to the following. ☞ Page 1802 Master axis 1 data source selection (MasterValueSourceM1)/Master axis 2 data source selection (MasterValueSourceM2)
MasterValueSourceM2	Master axis 2 data source selection	INT (MC_SOURCE)	At start	1, 2, 101, 102	1	The data source of Master axis 2 (Master2) is set. • 1: Set value (mcSetValue) • 2: Actual value (mcActualValue) • 101: Latest set value (mcLatestSetValue) • 102: Latest actual value (mcLatestActualValue) For details, refer to the following. ☞ Page 1802 Master axis 1 data source selection (MasterValueSourceM1)/Master axis 2 data source selection (MasterValueSourceM2)
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0, 1	0	The buffer mode is set. • 0: Aborting(mcAborting) • 1: Buffered(mcBuffered) For details, refer to the following. ☞ Page 1803 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	Set "00000000H". *: If a value other than "00000000H" is set, Out of Options Range (error code: 1ABBH) occurs.

■Output variables

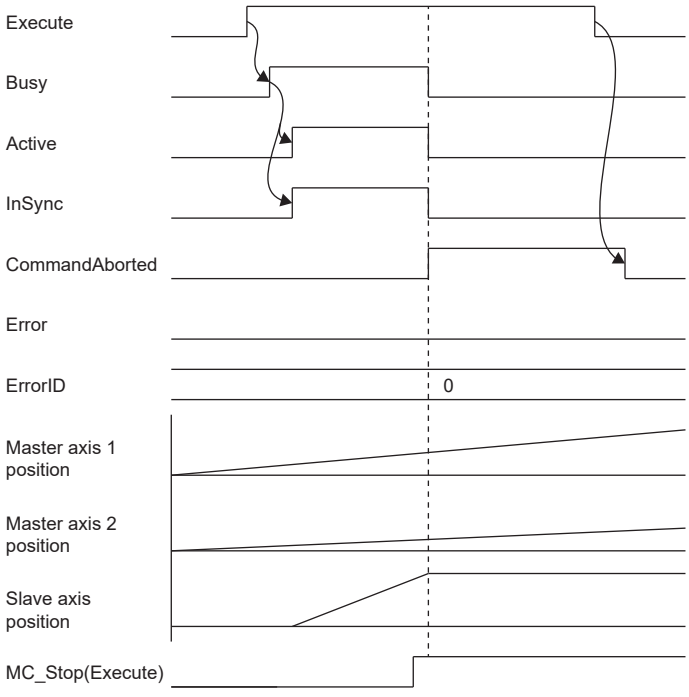
Output variable	Name	Data type	Default value	Description
InSync	In synchronization	BOOL	FALSE	This variable becomes TRUE when Slave axis (Slave) starts synchronization.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_CombineAxes (Addition/Subtraction Positioning) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE when Slave axis (Slave) is being controlled.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MC_CombineAxes (Addition/Subtraction Positioning) is aborted. The variable becomes FALSE when Execution command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

- This FB sets Addition/subtraction method selection (CombineMode), Master axis 1 gear ratio numerator (GearRatioNumeratorM1), Master axis 1 gear ratio denominator (GearRatioDenominatorM1), Master axis 2 gear ratio numerator (GearRatioNumeratorM2), Master axis 2 gear ratio denominator (GearRatioDenominatorM2), Master axis 1 data source selection (MasterValueSourceM1), Master axis 2 data source selection (MasterValueSourceM2), and Buffer mode (BufferMode), and performs the addition/subtraction positioning.
- To stop the operation, perform MC_Stop (Forced Stop).
- The movement amounts of Master axis 1 (Master1) and Master axis 2 (Master2) is combined. In addition, a gear ratio can be set for each master axis. Combining the movement amounts adds or subtracts the numerical values regardless of the units of Master axis 1 (Master1) and Master axis 2 (Master2).

■Timing chart

- When the processing is completed successfully



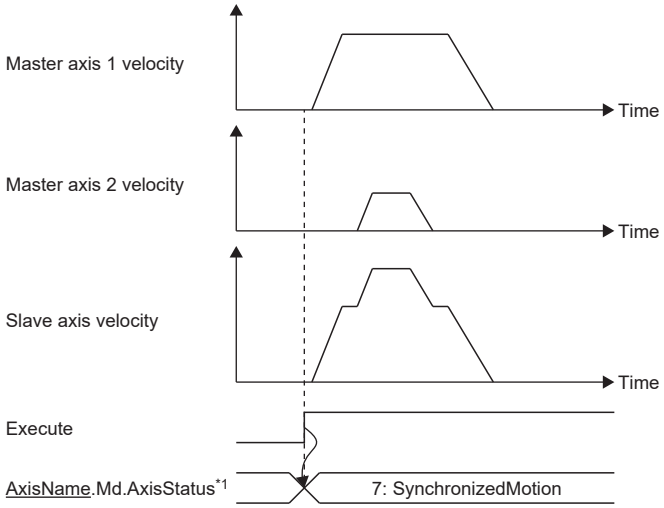
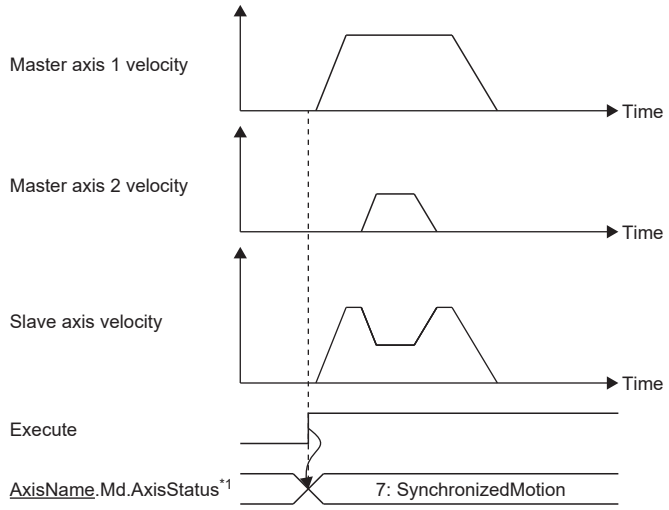
- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Addition/subtraction method selection (CombineMode)

The method used to combine the movement amounts of Master axis 1 (Master1) and Master axis 2 (Master2) is set.
Combining the movement amounts adds or subtracts the numerical values regardless of the units of Master axis 1 (Master1) and Master axis 2 (Master2).

Setting value	Description
0: Add positions of 2 input axes (mcAddAxes)	The movement amounts of Master axis 1 (Master1) and Master axis 2 (Master2) are added for output. • Movement amount of Slave axis (Slave) = Movement amount of Master axis 1 (Master1) + Master axis 2 (Master2) 
1: Subtract positions of 2 input axes (mcSubAxes)	The movement amount of Master axis 2 (Master2) is subtracted from that of Master axis 1 (Master1) for output. • Movement amount of Slave axis (Slave) = Movement amount of Master axis 1 (Master1) - Master axis 2 (Master2) 

*1 Axis status of Slave axis (Slave)

■Master axis 1 gear ratio numerator (GearRatioNumeratorM1)/Master axis 1 gear ratio denominator (GearRatioDenominatorM1)

Master axis 1 gear ratio numerator (GearRatioNumeratorM1) and Master axis 1 gear ratio denominator (GearRatioDenominatorM1) are set together.

Master axis 1 gear ratio numerator (GearRatioNumeratorM1) sets the numerator value used when converting and combining the movement amount of Master axis 1 (Master1).

Master axis 1 gear ratio denominator (GearRatioDenominatorM1) sets the denominator value used when converting and combining the movement amount of Master axis 1 (Master1).

The movement amount is converted as shown below.

$$\text{Movement amount of the master axis 1 (Master1) after conversion} = \text{movement amount of the master axis 1 (Master1) before conversion} \times \frac{\text{Master axis 1 gear ratio numerator (GearRatioNumeratorM1)}}{\text{Master axis 1 gear ratio denominator (GearRatioDenominatorM1)}}$$

- Set a positive value in Master axis 1 gear ratio denominator (GearRatioDenominatorM1).
- The movement amount can be reversed and combined by setting a negative value in Master axis 1 gear ratio numerator (GearRatioNumeratorM1).
- To combine and have the movement amount of Master axis 1 (Master1) be "0" after conversion, set "0" as the setting value in Master axis 1 gear ratio numerator (GearRatioNumeratorM1).

■Master axis 2 gear ratio numerator (GearRatioNumeratorM2)/Master axis 2 gear ratio denominator (GearRatioDenominatorM2)

Master axis 2 gear ratio numerator (GearRatioNumeratorM2) and Master axis 2 gear ratio denominator (GearRatioDenominatorM2) are set together.

Master axis 2 gear ratio numerator (GearRatioNumeratorM2) sets the numerator value used when converting and combining the movement amount of Master axis 2 (Master2).

Master axis 2 gear ratio denominator (GearRatioDenominatorM2) sets the denominator value used when converting and combining the movement amount of Master axis 2 (Master2).

The movement amount is converted as shown below.

$$\text{Movement amount of the master axis 2 (Master2) after conversion} = \text{movement amount of the master axis 2 (Master2) before conversion} \times \frac{\text{Master axis 2 gear ratio numerator (GearRatioNumeratorM2)}}{\text{Master axis 2 gear ratio denominator (GearRatioDenominatorM2)}}$$

- Set a positive value in Master axis 2 gear ratio denominator (GearRatioDenominatorM2).
- The movement amount can be reversed and combined by setting a negative value in Master axis 2 gear ratio numerator (GearRatioNumeratorM2).
- To combine and have the movement amount of Master axis 2 (Master2) be "0" after conversion, set "0" as the setting value in Master axis 2 gear ratio numerator (GearRatioNumeratorM2).

■Master axis 1 data source selection (MasterValueSourceM1)/Master axis 2 data source selection (MasterValueSourceM2)

Slave axis (Slave) sets the position type of Master axis 1 (Master1) or Master axis 2 (Master2) that will execute single axis synchronous control.

The operation may differ depending on the execution order of the FBs or if the operation cycles of Master axis 1 (Master1) or Master axis 2 (Master2) and Slave axis (Slave) are different.

Operation is performed with the same value as the set value when specifying Actual value in Master axis 1 data source selection (MasterValueSourceM1) or Master axis 2 data source selection (MasterValueSourceM2) while an axis type other than real axis is set in Master axis 1 (Master1) or Master axis 2 (Master2).

Setting value	Description
1: Set value (mcSetValue)	Uses the commanded position of the master axis in the previous operation cycle.
2: Actual value (mcActualValue)	Uses the actual position of the master axis in the previous operation cycle.
101: Latest set value (mcLatestSetValue)	Uses the commanded position of the master axis in the current operation cycle.
102: Latest actual value (mcLatestActualValue)	Uses the actual position of the master axis in the current operation cycle.

Point

When Master axis 1 data source selection (MasterValueSourceM1) or Master axis 2 data source selection (MasterValueSourceM2) are set to "1: Set value (mcSetValue)" or "101: Latest set value (mcLatestSetValue)" and the master axis becomes servo OFF by a servo alarm or forced stop, the amount of the value change may be too large. This can be prevented by setting Master axis 1 data source selection (MasterValueSourceM1) or Master axis 2 data source selection (MasterValueSourceM2) to "2: Actual value (mcActualValue)" or "102: Latest actual value (mcLatestActualValue)".

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following buffer modes are settable in MC_CombineAxes (Addition/Subtraction Positioning).

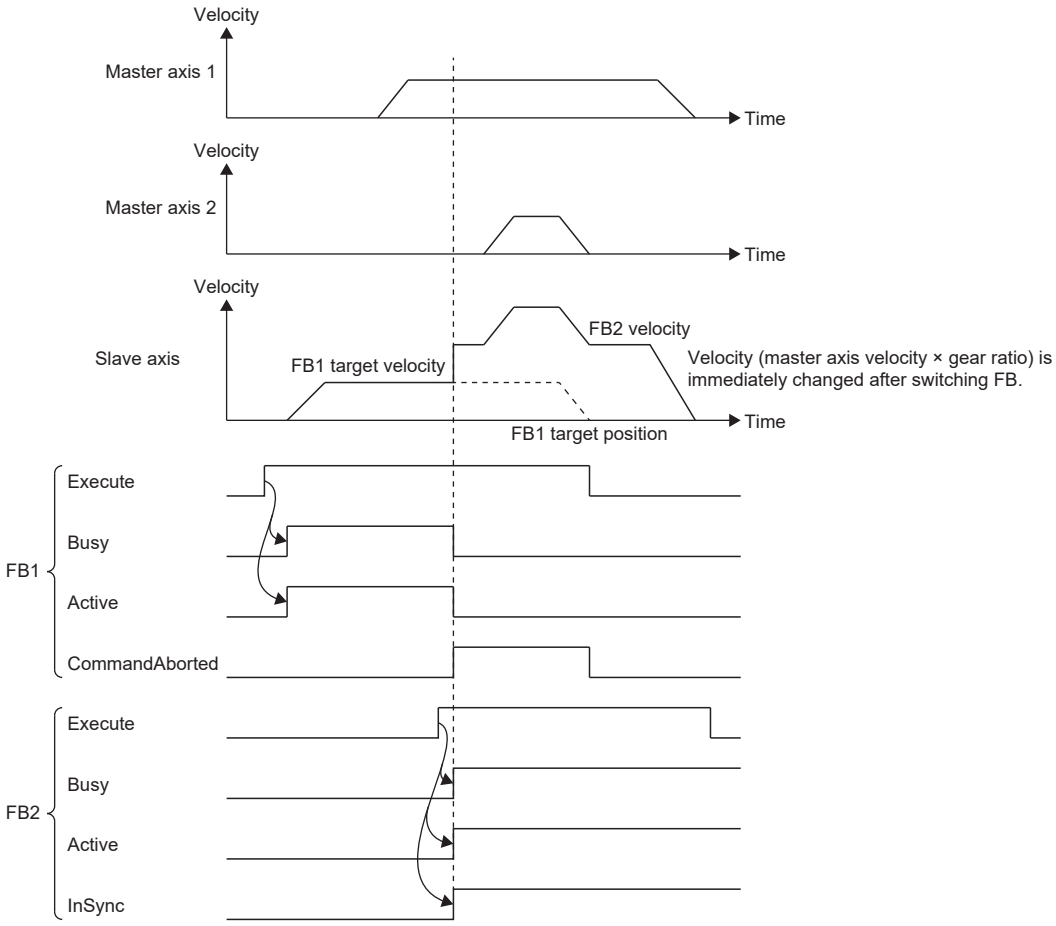
The velocity of MC_CombineAxes (Addition/Subtraction Positioning) follows the synchronizing FB (master axis) and changes immediately based on the master axis velocity and the set gear data when the FB is switched.

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.
1: Buffered(mcBuffered)	Buffers the next FB on the under-control FB. If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after completion of the under-control FB.

Ex.

When linked start FB2 (Slave axis (Slave): MC_CombineAxes (Addition/Subtraction Positioning)) has Buffer mode (BufferMode) set to "0: Aborting (mcAborting)" while FB1 is being executed

· When Addition/subtraction method selection (CombineMode) is set to "0: Add positions of 2 input axes (mcAddAxes)"



Velocity of FB2 occurs after In synchronization (InSync) becomes TRUE.

Point

For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

Program example

The following is a program example. The addition/subtraction positioning start command (bCombineAxesStart) is set to TRUE, and positioning of the slave axis (Axis0001) is performed using the added/subtracted movement amount of the two master axes, Axis 2 (Axis0002) of the JOG 1 FB and Axis 3 (Axis0003) of the JOG 1 FB.

• Settings

Item		Setting value	
		Data for JOG 1	Data for JOG 2
Data settings for JOG operation	Velocity	1000.0	500.0
	Acceleration	5000.0	5000.0
	Deceleration	5000.0	5000.0
	Jerk	0.0	0.0

■Axes

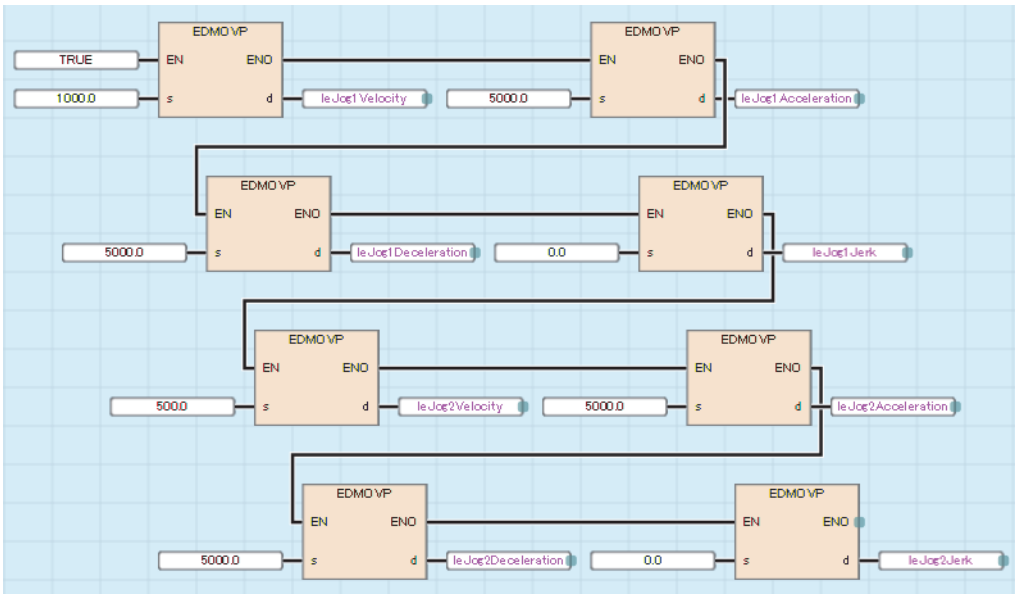
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1
2	Axis0002	AXIS_REF	Axis 2
3	Axis0003	AXIS_REF	Axis 3

■Labels used

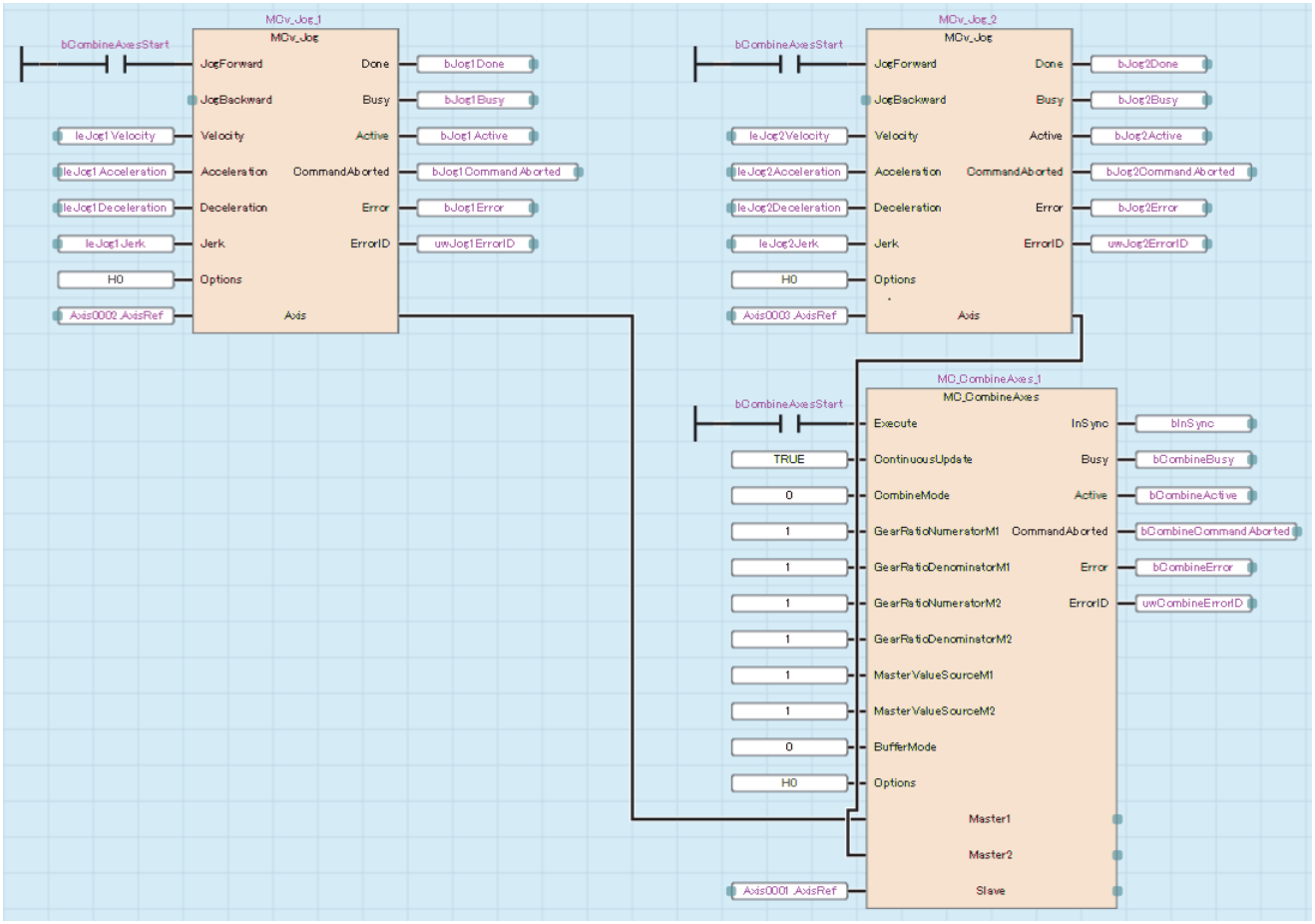
Label name	Data type	Comment
MC_CombineAxes_1	MC_CombineAxes	Addition/Subtraction Positioning FB
bCombineAxesStart	Bit	Addition/subtraction positioning start command
bInSync	Bit	In synchronization
bCombineBusy	Bit	Executing
bCombineActive	Bit	Controlling
bCombineCommandAborted	Bit	Abortion of execution
bCombineError	Bit	Error
uwCombineErrorID	Word [unsigned]/bit string [16-bit]	Error code
MCv_Jog_1	MCv_Jog	JOG 1 FB
leJog1Velocity	Double-precision real number	JOG velocity
leJog1Acceleration	Double-precision real number	JOG acceleration
leJog1Deceleration	Double-precision real number	JOG deceleration
leJog1Jerk	Double-precision real number	JOG jerk
bJog1Done	Bit	Execution completion
bJog1Busy	Bit	Executing
bJog1Active	Bit	Controlling
bJog1CommandAborted	Bit	Abortion of execution
bJog1Error	Bit	Error
uwJog1ErrorID	Word [unsigned]/bit string [16-bit]	Error code
MCv_Jog_2	MCv_Jog	JOG 2 FB
leJog2Velocity	Double-precision real number	JOG velocity
leJog2Acceleration	Double-precision real number	JOG acceleration
leJog2Deceleration	Double-precision real number	JOG deceleration
leJog2Jerk	Double-precision real number	JOG jerk
bJog2Done	Bit	Execution completion
bJog2Busy	Bit	Executing
bJog2Active	Bit	Controlling
bJog2CommandAborted	Bit	Abortion of execution
bJog2Error	Bit	Error
uwJog2ErrorID	Word [unsigned]/bit string [16-bit]	Error code

■FBD/LD program

• Data settings for JOG operation



• JOG 1/JOG 2/Addition/Subtraction Positioning



■ST program

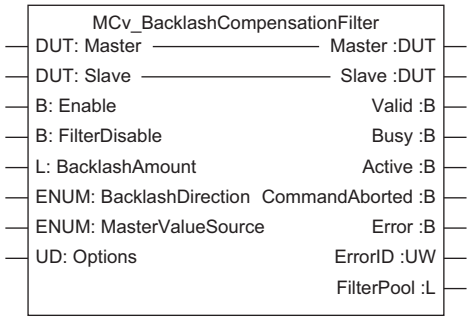
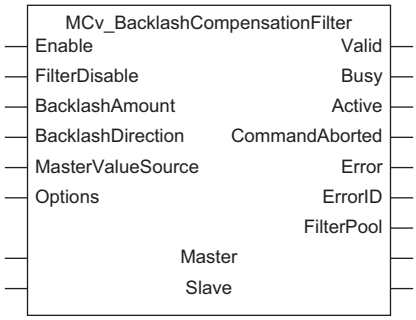
```
//-----Data settings for JOG operation-----
leJog1Velocity:= 1000.0;
leJog1Acceleration:= 5000.0;
leJog1Deceleration:= 5000.0;
leJog1Jerk:= 0.0;
leJog2Velocity:= 500.0;
leJog2Acceleration:= 5000.0;
leJog2Deceleration:= 5000.0;
leJog2Jerk:= 0.0;

//-----JOG 1-----
MCv_Jog_1(
   Axis:= Axis0002.AxisRef ,
   JogForward:= bCombineAxesStart ,
   Velocity:= leJog1Velocity ,
   Acceleration:= leJog1Acceleration ,
   Deceleration:= leJog1Deceleration ,
   Jerk:= leJog1Jerk ,
   Options:= H00000000 ,
   Done=> bJog1Done ,
   Busy=> bJog1Busy ,
   Active=> bJog1Active ,
   CommandAborted=> bJog1CommandAborted ,
   Error=> bJog1Error ,
   ErrorID=> uwJog1ErrorID
);

//-----JOG 2-----
MCv_Jog_2(
   Axis:= Axis0003.AxisRef ,
   JogForward:= bCombineAxesStart ,
   Velocity:= leJog2Velocity ,
   Acceleration:= leJog2Acceleration ,
   Deceleration:= leJog2Deceleration ,
   Jerk:= leJog2Jerk ,
   Options:= H00000000 ,
   Done=> bJog2Done ,
   Busy=> bJog2Busy ,
   Active=> bJog2Active ,
   CommandAborted=> bJog2CommandAborted ,
   Error=> bJog2Error ,
   ErrorID=> uwJog2ErrorID
);
```

```
//-----Addition/Subtraction Positioning-----
MC_CombineAxes_1(
    Master1:= Axis0002.AxisRef ,
    Master2:= Axis0003.AxisRef ,
    Slave:= Axis0001.AxisRef ,
    Execute:= bCombineAxesStart ,
    ContinuousUpdate:= TRUE ,
    CombineMode:= MC_COMBINE_MODE__mcAddAxes ,
    GearRatioNumeratorM1:= 1 ,
    GearRatioDenominatorM1:= 1 ,
    GearRatioNumeratorM2:= 1 ,
    GearRatioDenominatorM2:= 1 ,
    MasterValueSourceM1:= MC_SOURCE__mcSetValue ,
    MasterValueSourceM2:= MC_SOURCE__mcSetValue ,
    BufferMode:= MC_BUFFER_MODE__mcAborting ,
    Options:= H00000000 ,
    InSync=> bInSync ,
    Busy=> bCombineBusy ,
    Active=> bCombineActive ,
    CommandAborted=> bCombineCommandAborted ,
    Error=> bCombineError ,
    ErrorID=> uwCombineErrorID
);
```

42.18 Backlash Compensation Filter

MCv_BacklashCompensationFilter

This FB performs filter processing that compensates the backlash amount in the machine system according to the movement direction.

Ladder	FBD/LD	ST
		MCv_BacklashCompensationFilter(Master:= ?AXIS_REF? , Slave:= ?AXIS_REF? , Enable:= ?BOOL? , FilterDisable:= ?BOOL? , BacklashAmount:= ?LREAL? , BacklashDirection:= ?INT? , MasterValueSource:= ?INT? , Options:= ?DWORD? , Valid=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD? , FilterPool=> ?LREAL?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Master	Master axis	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)
Slave	Slave axis	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Enable	Enable	BOOL	Always	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_BacklashCompensationFilter (Backlash Compensation Filter) is executed.
FilterDisable	Filter disable	BOOL	Always	TRUE, FALSE	FALSE	The filter processing method is set. • FALSE: Operation is performed according to the filter setting. • TRUE: The filter is disabled and the Master axis (Master) input is transmitted to Slave axis (Slave) as it is. *: When this variable is set to TRUE, the value of Filter cumulated value (FilterPool) becomes "0" and the disposed value from Filter cumulated value (FilterPool) is stored in Disposed filter cumulated value (PurgedFilterPool).
BacklashAmount	Backlash amount	LREAL	At start	0.0 to 2500000000.0 [U]	0.0	The backlash amount used in the machine system is set.

Input variable	Name	Data type	Input	Valid range	Default value	Description
BacklashDirection	Backlash compensation direction	INT (MC_DIRECTION)	At start	1, 2	0	The backlash compensation direction is set. • 1: Positive direction (mcPositiveDirection) • 2: Negative direction (mcNegativeDirection) *: If this setting is omitted, Out of Backlash Compensation Direction Range (error code: 1BC0H) occurs. For details, refer to the following. ☞ Page 1813 Backlash compensation direction (BacklashDirection)
MasterValueSource	Master axis data source selection	INT (MC_SOURCE)	At start	1, 101	1	The data source of Master axis (Master) is set. • 1: Set value (mcSetValue) • 101: Latest set value (mcLatestSetValue) For details, refer to the following. ☞ Page 1813 Master axis data source selection (MasterValueSource)
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	The function options are set. For details, refer to the following. ☞ Page 1814 Options (Options)

■Output variables

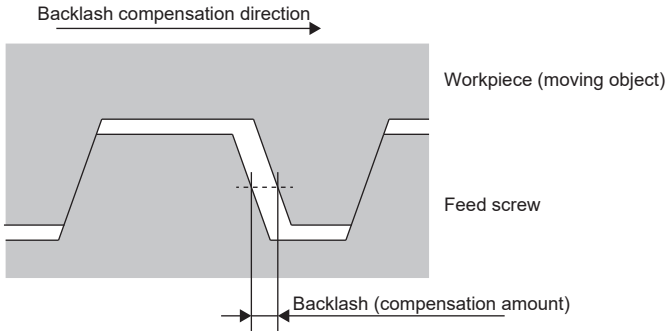
Output variable	Name	Data type	Default value	Description
Valid	Output value valid	BOOL	FALSE	This variable becomes TRUE when the output value is valid.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_BacklashCompensationFilter (Backlash Compensation Filter) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE when MCv_BacklashCompensationFilter (Backlash Compensation Filter) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MCv_BacklashCompensationFilter (Backlash Compensation Filter) is aborted.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used
FilterPool	Filter cumulated value	LREAL	0.0	This variable outputs the slave axis movement amount cumulated in filter processing. When the backlash filter is used, the compensation amount transmitted to the slave axis is the reversed positive/negative value in Filter cumulated value (FilterPool).

■Public variables

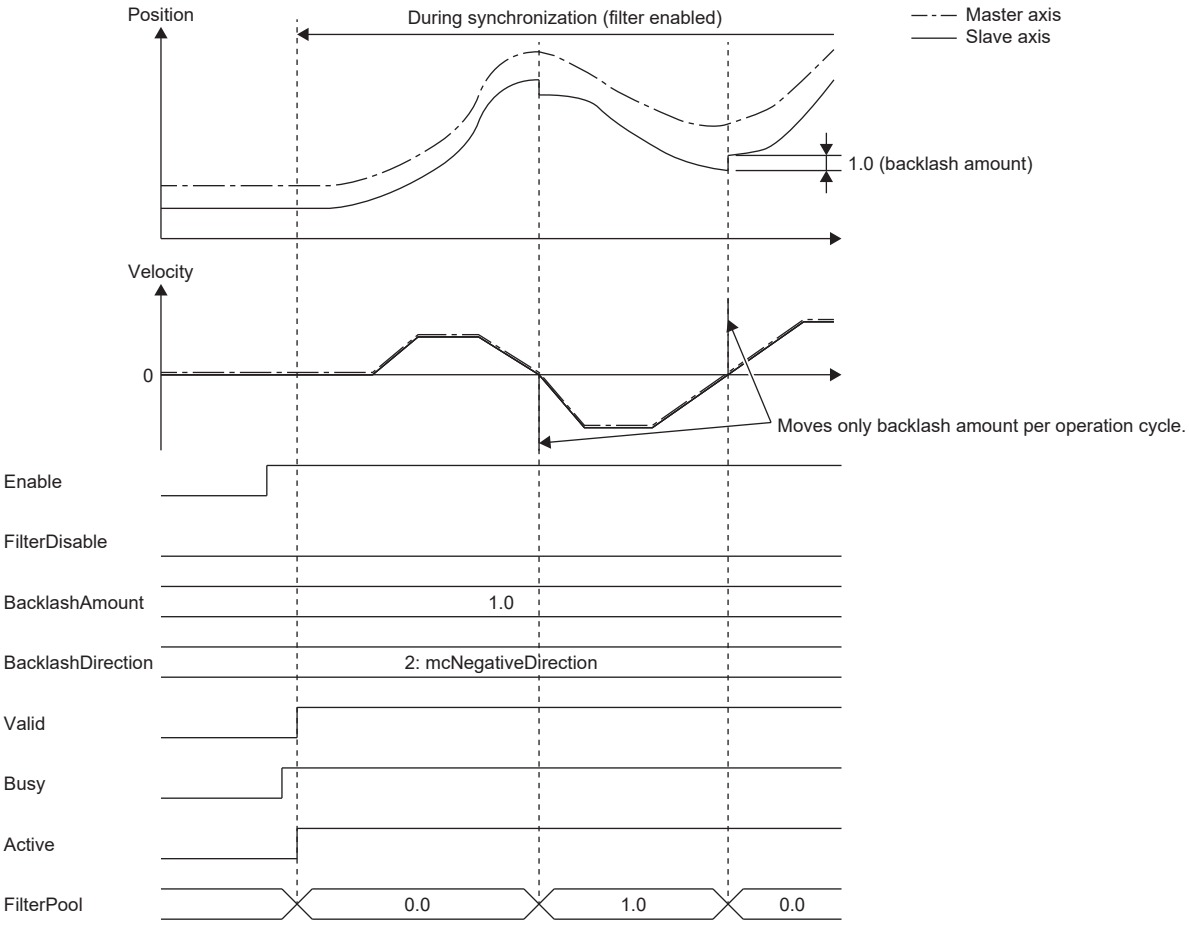
Public variable	Name	Data type	Default value	Description
PurgedFilterPool	Disposed filter cumulated value	LREAL	0.0	When Filter disable (FilterDisable) is set to TRUE, this variable stores the disposed value from Filter cumulated value (FilterPool). Disposed filter cumulated value (PurgedFilterPool) is not cumulative. The value is cleared to 0 at start of the FB.

Processing details

- This FB compensates for the backlash amount in the machine system.

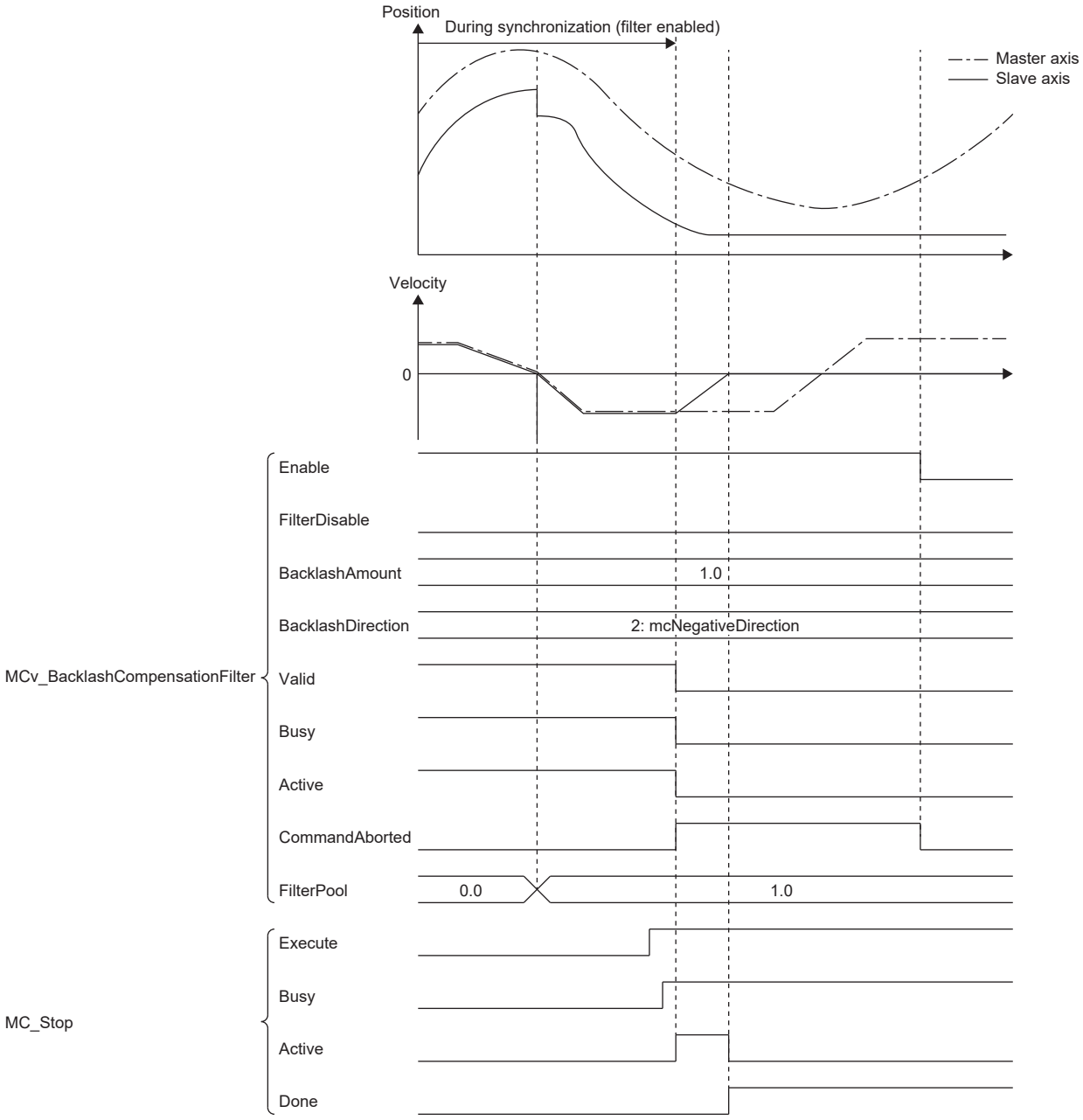


- When the movement direction of Master axis (Master) is the same as Backlash compensation direction (BacklashDirection), Slave axis (Slave) synchronizes to the position equivalent to the Master axis (Master) current position shifted in the Backlash direction (BacklashDirection) by the value of Backlash amount (BacklashAmount). When the movement direction of Master axis (Master) is the reverse of Backlash compensation direction (BacklashDirection), Slave axis (Slave) synchronizes to the Master axis (Master) current position.
  Whenever the Master axis (Master) movement direction changes, backlash compensation is performed with the Slave axis (Slave) extra movement amount limited to only the value of Backlash amount (BacklashAmount).



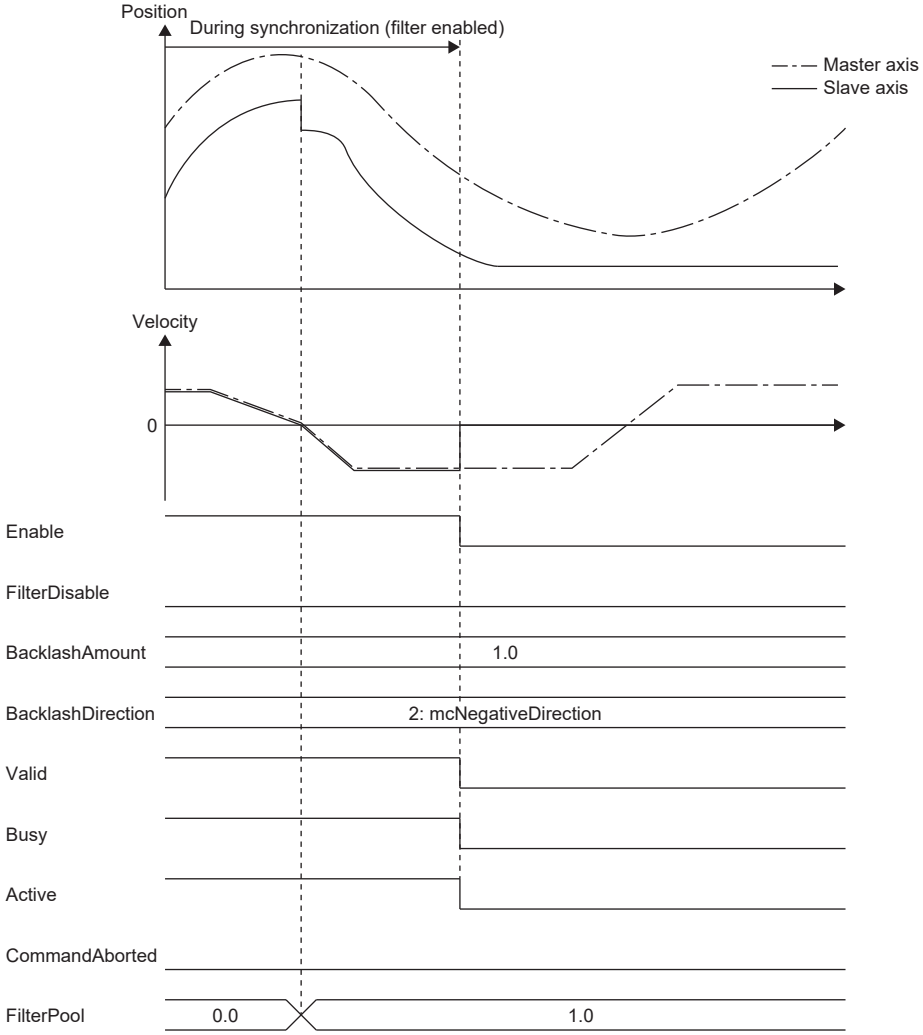
• This section describes the stop operation specific to the backlash compensation filter.

When the stop factor occurs on Slave axis (Slave), Slave axis (Slave) starts the stop operation from the synchronous position. The compensated movement amount (the value from Filter cumulated value (FilterPool)) will not be restored to its original value.



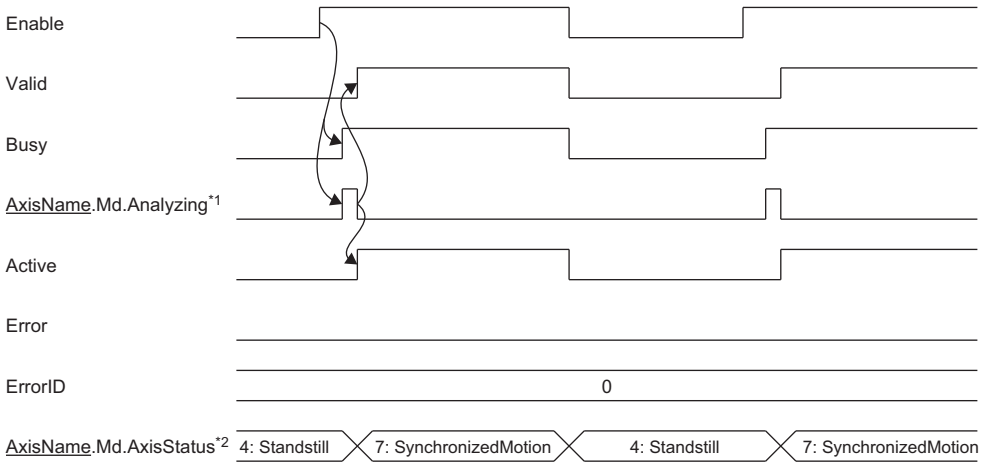
• Operation at filter disabled (at Enable (Enable) falling edge)

Slave axis (Slave) stops immediately at the synchronous position upon reaching the Enable (Enable) falling edge. The compensated movement amount (the value from Filter cumulated value (FilterPool)) will not be restored to its original value.



■Timing chart

• When the processing is completed successfully



*1 "Analyzing" of Slave axis (Slave).

*2 Axis status of Slave axis (Slave)

• When the processing is completed with an error

For the timing chart, refer to the following.

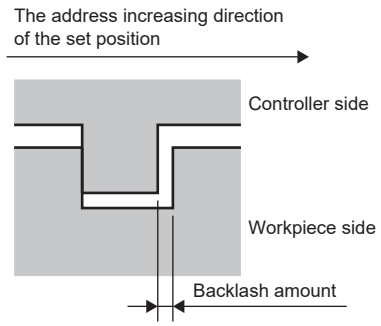
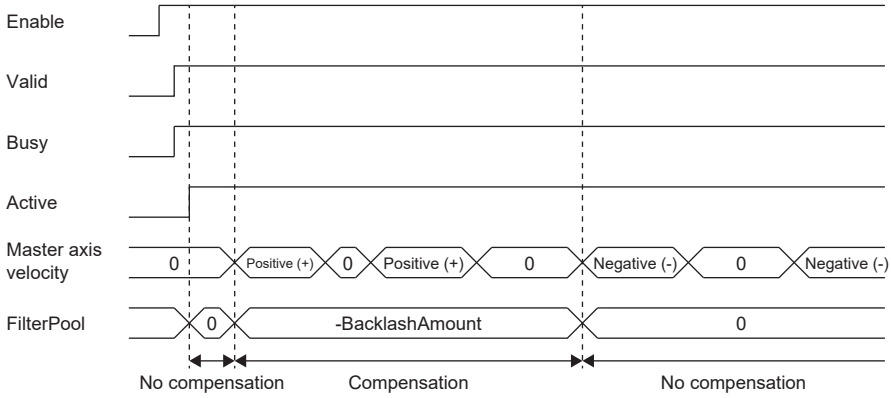
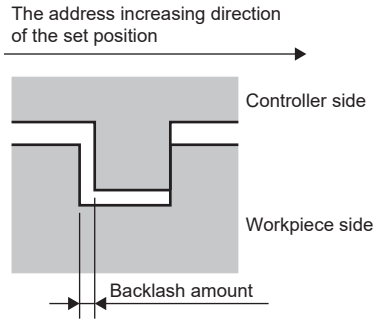
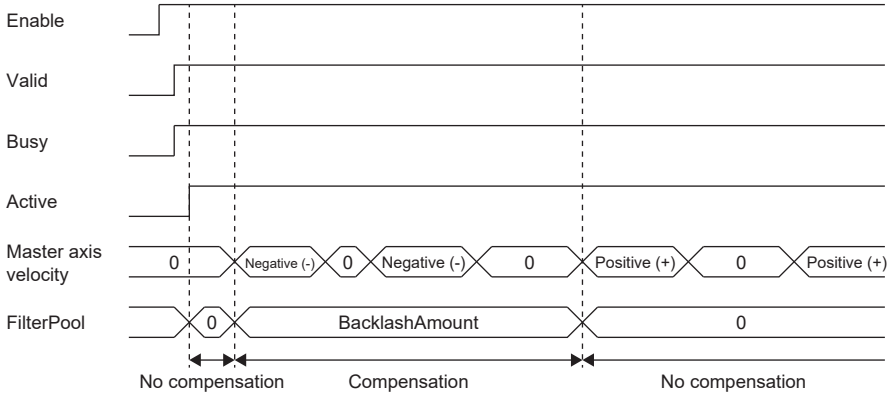
☞ Page 1390 Basic operation of Enable (Enable) type motion control FBs

■Backlash compensation direction (BacklashDirection)

The backlash compensation direction is set.

The compensation amount is output to Slave axis (Slave) when the Master axis (Master) movement direction is the value of Backlash compensation direction (BacklashDirection).

When the backlash compensation filter is enabled (when starting the FB with Filter disable (FilterDisable) set to FALSE, or when Filter disable (FilterDisable) reaches the trailing edge during execution), compensation is performed in the address increasing direction of the Master axis (Master) set position so that the backlash status of the machine system is in the following relationship.

Setting value	Description
1: Positive direction (mcPositiveDirection) 	When the movement direction of Master axis (Master) is the positive direction, Slave axis (Slave) synchronizes to the position equivalent to the Master axis (Master) current position shifted in the positive direction by the value of Backlash amount (BacklashAmount). 
2: Negative direction (mcNegativeDirection) 	When the movement direction of Master axis (Master) is the negative direction, Slave axis (Slave) synchronizes to the position equivalent to the Master axis (Master) current position shifted in the negative direction by the value of Backlash amount (BacklashAmount). 

■Master axis data source selection (MasterValueSource)

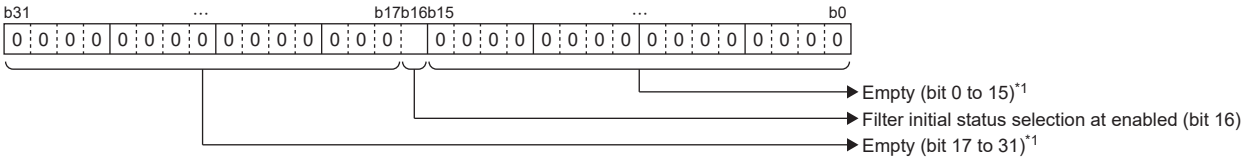
Slave axis (Slave) sets the position type of Master axis (Master) that will execute single axis synchronous control.

The operation may differ depending on the execution order of the FBs or if the operation cycles of Master axis (Master) and Slave axis (Slave) are different.

Setting value	Description
1: Set value (mcSetValue)	Uses the commanded position of the master axis in the previous operation cycle.
101: Latest set value (mcLatestSetValue)	Uses the commanded position of the master axis in the current operation cycle.

■Options (Options)

The function options for MCv_BacklashCompensationFilter (Backlash Compensation Filter) are set with bit specification.
The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
16	Enable filter initial status selection	Set the initial status for when the backlash compensation filter is enabled (when starting the FB with Filter disable (FilterDisable) set to FALSE, or when Filter disable (FilterDisable) reaches the trailing edge during execution). • 0: First compensation *:If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs and the function does not start.

• Enable filter initial status selection (Bit 16)

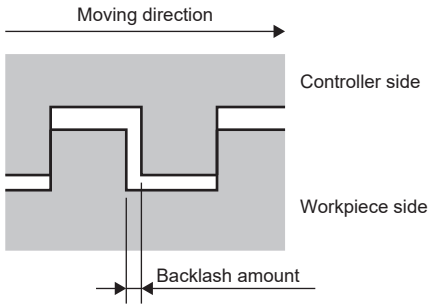
Setting value	Description
0: First compensation	Regardless of the Master axis (Master) current direction when the filter is enabled, starts filter operation assuming that the position relation of the Master axis (Master) current position and the Slave axis (Slave) current position do not have the backlash compensation added.

■Procedure for enabling the backlash compensation filter

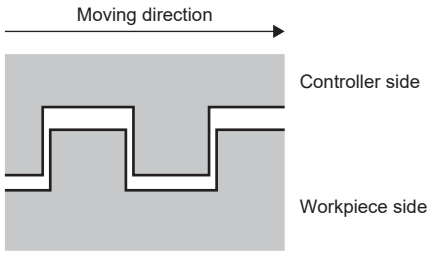
This section describes the procedure for enabling the backlash compensation filter and a usage example. Perform this procedure after the machine position is established by homing, or when restarting the backlash compensation filter after cancelling it midway.

1. Move the real drive axis by operations such as homing or JOG operation, and maintain the status so that the backlash of the machine system is generated in only one direction as shown in the correct example below.

■Correct example



■Incorrect example



2. Match the Master axis (Master) set position with the real drive axis set position by using the current position change or others.

3. Set Backlash compensation direction (BacklashDirection) so that it is the reverse direction of the direction that the real drive axis was moved to in Step 1., then start the backlash compensation filter. In addition, start any back part function blocks that are present.

Precautions

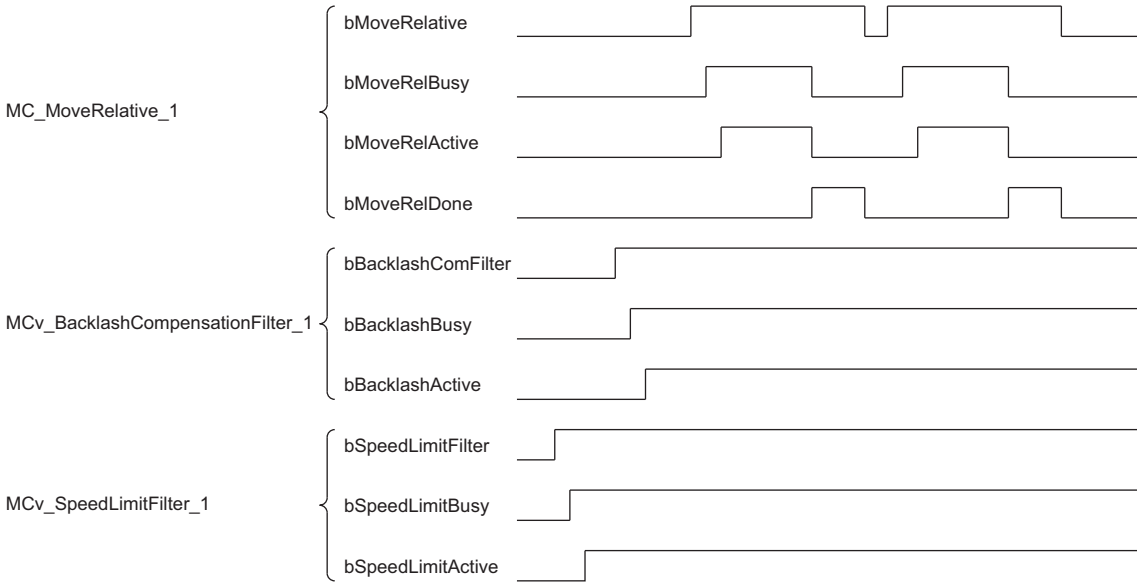
- The backlash compensation is a function which the command direction and the backlash status should not change from the external force and other factors. Correct compensation will not be performed when this function is used in the following cases.
  - A machine that has external force constantly applied in one direction, such as a vertical axis
  - A mechanism (such as rack and pinion mechanism) in which the backlash amount changes depending on the machine position
- When the Master axis (Master) movement direction changes, Slave axis (Slave) moves only the value of Backlash amount (BacklashAmount) for one operation cycle. Depending on the value, it may exceed the command frequency that the speed limit value and the driver unit can accept, so connect the speed limit filter or the smoothing filter to the backlash compensation filter back part.
- The Slave axis (Slave) current position (such as set position and machine feed position) will be the position to which the value of Backlash amount (BacklashAmount) compensation is added. Backlash amount (BacklashAmount) compensation will not be added to the Master axis (Master) current position.
- If Filter disable (FilterDisable) is switched during operation, the Slave axis (Slave) velocity may change rapidly.
- After enabling the command filter, when the Master axis (Master) command is not transmitted to Slave axis (Slave) or the back part synchronous control FB because the stop factor occurred on Slave axis (Slave), the synchronous position relation deviates. Before enabling the command filter again, execute the synchronous positioning as needed.
- When using the mixed operation cycles function, there are some restrictions if the control operation cycle setting of the master axis and slave axis are not the same. For details, refer to each function and the following.

📖 User's manual for the controller used

Program example

The following is a program example. The speed limit filter is connected to the output stage of the backlash compensation filter and the filter processing is performed.

• Timing chart



• Settings

Item		Setting value
Backlash amount		1.0
Data settings for single axis positioning	Movement amount	1000.0
	Velocity	20.0
	Acceleration	300.0
	Deceleration	300.0
	Jerk	0.0
Positive direction/negative direction limit value	Positive direction limit value	100.0
	Negative direction limit value	100.0

■Axes

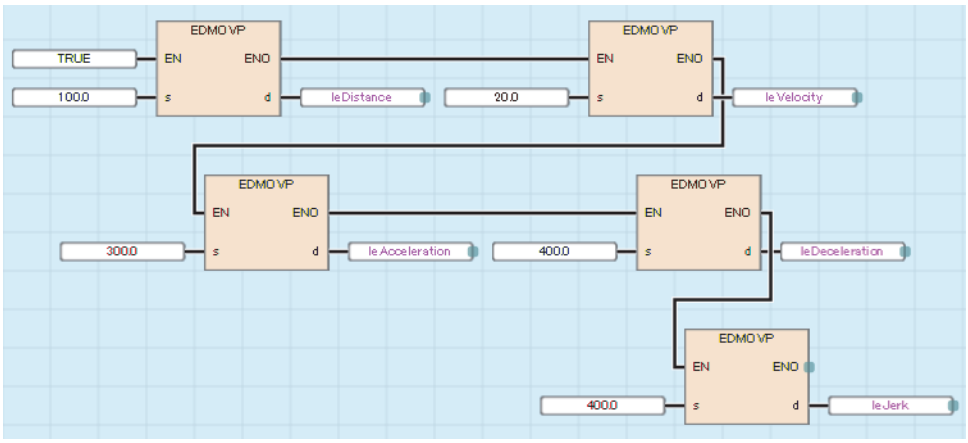
Axis No.	Label name	Data type	Comment
1	VirtualAxis0001	AXIS_REF	Axis 1 (Virtual drive axis)
2	VirtualAxis0002	AXIS_REF	Axis 2 (Virtual drive axis)
3	Axis0003	AXIS_REF	Axis 3 (Real drive axis)

■Labels used

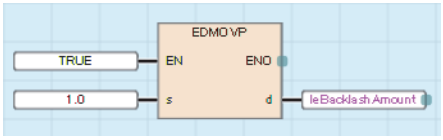
Label name	Data type	Comment
MC_MoveRelative_1	MC_MoveRelative	Relative Value Positioning FB
bMoveRelative	Bit	Single axis relative value positioning start
leDistance	Double-precision real number	Movement amount
leVelocity	Double-precision real number	Velocity
leAcceleration	Double-precision real number	Acceleration
leDeceleration	Double-precision real number	Deceleration
leJerk	Double-precision real number	Jerk
bMoveRelDone	Bit	Execution completion
bMoveRelBusy	Bit	Executing
bMoveRelActive	Bit	Controlling
bMoveRelCommandAborted	Bit	Abortion of execution
bMoveRelError	Bit	Error
uwMoveRelErrorID	Word [unsigned]/bit string [16-bit]	Error code
MCv_BacklashCompensationFilter_1	MCv_BacklashCompensationFilter	Backlash Compensation Filter FB
bBacklashComFilter	Bit	Backlash compensation filter command
leBacklashAmount	Double-precision real number	Backlash amount
bBacklashValid	Bit	Output value valid
bBacklashBusy	Bit	Executing
bBacklashActive	Bit	Controlling
bBacklashCommandAborted	Bit	Abortion of execution
bBacklashError	Bit	Error
uwBacklashErrorID	Word [unsigned]/bit string [16-bit]	Error code
leBacklashFilterPool	Double-precision real number	Filter cumulated value
MCv_SpeedLimitFilter_1	MCv_SpeedLimitFilter	Speed Limit Filter
bSpeedLimitFilter	Bit	Speed limit filter command
lePositiveLimit	Double-precision real number	Positive direction limit value
leNegativeLimit	Double-precision real number	Negative direction limit value
bSpeedLimitValid	Bit	Output value valid
bSpeedLimitBusy	Bit	Executing
bSpeedLimitActive	Bit	Controlling
bSpeedLimitCommandAborted	Bit	Abortion of execution
bSpeedLimitError	Bit	Error
uwSpeedLimitErrorID	Word [unsigned]/bit string [16-bit]	Error code
leFilterPool	Double-precision real number	Filter cumulated value

■FBD/LD program

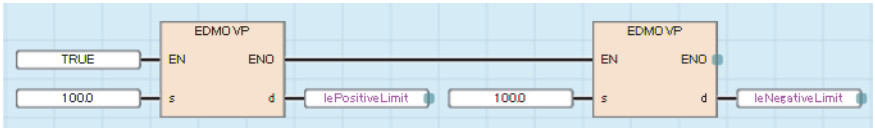
- Data settings for single axis positioning



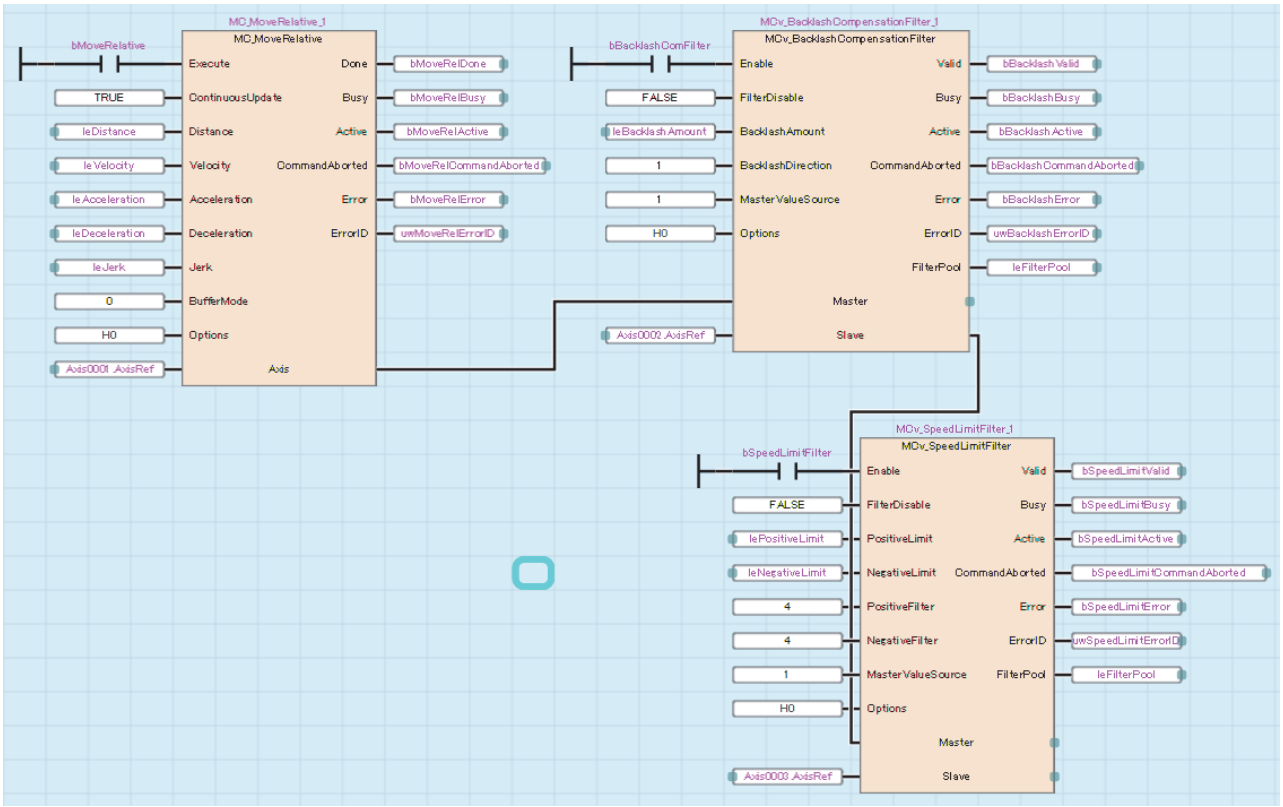
- Backlash amount



- Positive direction/negative direction limit value



- Relative Value Positioning/Backlash Compensation Filter/Speed Limit Filter



■ST program

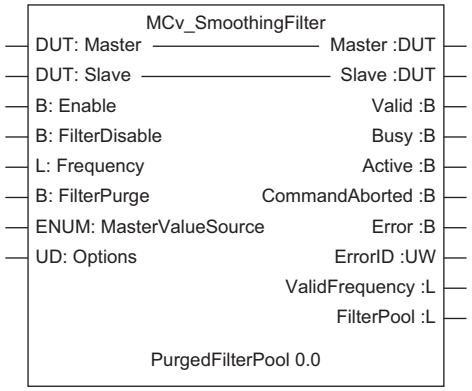
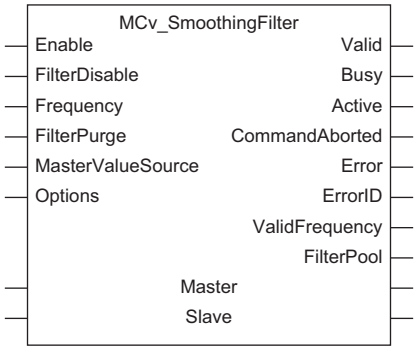
```
//-----Data settings for single axis positioning-----
leDistance:= 100.0;
leVelocity:= 20.0;
leAcceleration:= 300.0;
leDeceleration:= 400.0;
leJerk:= 0.0;

//-----Relative Value Positioning-----
MC_MoveRelative_1(
   Axis:= VirtualAxis0001.AxisRef ,
   Execute:= bMoveRelative ,
   ContinuousUpdate:= TRUE ,
   Distance:= leDistance ,
   Velocity:= leVelocity ,
   Acceleration:= leAcceleration ,
   Deceleration:= leDeceleration ,
   Jerk:= leJerk ,
   BufferMode:= MC_BUFFER_MODE__mcAborting ,
   Options:= H00000000 ,
   Done=> bMoveRelDone ,
   Busy=> bMoveRelBusy ,
   Active=> bMoveRelActive ,
   CommandAborted=> bMoveRelCommandAborted ,
   Error=> bMoveRelError ,
   ErrorID=> uwMoveRelErrorID
);

//-----Backlash amount-----
leBacklashAmount:= 1.0;

//-----Backlash Compensation Filter-----
MCv_BacklashCompensationFilter_1(
   Master:= MC_MoveRelative_1.Axis ,
   Slave:= VirtualAxis0002.AxisRef ,
   Enable:= bBacklashComFilter ,
   FilterDisable:= FALSE ,
   BacklashAmount:= leBacklashAmount ,
   BacklashDirection:= MC_DIRECTION__mcPositiveDirection ,
   MasterValueSource:= MC_SOURCE__mcSetValue ,
   Options:= H00000000 ,
   Valid=> bBacklashValid ,
   Busy=> bBacklashBusy ,
   Active=> bBacklashActive ,
   CommandAborted=> bBacklashCommandAborted ,
   Error=> bBacklashError ,
   ErrorID=> uwBacklashErrorID ,
   FilterPool=> leBacklashFilterPool
);
```

```
//-----Positive direction/negative direction limit value-----
lePositiveLimit:= 100.0;
leNegativeLimit:= 100.0;

//-----Speed Limit Filter-----
MCv_SpeedLimitFilter_1(
    Master:= MCv_BacklashCompensationFilter_1.Slave ,
    Slave:= Axis0003.AxisRef ,
    Enable:= bSpeedLimitFilter ,
    FilterDisable:= FALSE ,
    PositiveLimit:= lePositiveLimit ,
    NegativeLimit:= leNegativeLimit ,
    PositiveFilter:= MC_VELOCITY_LIMIT_MODE__ClampWithoutRamp ,
    NegativeFilter:= MC_VELOCITY_LIMIT_MODE__ClampWithoutRamp ,
    MasterValueSource:= MC_SOURCE__mcSetValue ,
    Options:= H00000000 ,
    Valid=> bSpeedLimitValid ,
    Busy=> bSpeedLimitBusy ,
    Active=> bSpeedLimitActive ,
    CommandAborted=> bSpeedLimitCommandAborted ,
    Error=> bSpeedLimitError ,
    ErrorID=> uwSpeedLimitErrorID ,
    FilterPool=> leFilterPool
);
```

42.19 Smoothing Filter

MCv_SmoothingFilter

This FB performs filter processing according to the specified frequency.

Ladder	FBD/LD	ST
		MCv_SmoothingFilter(Master:= ?AXIS_REF? , Slave:= ?AXIS_REF? , Enable:= ?BOOL? , FilterDisable:= ?BOOL? , Frequency:= ?LREAL? , FilterPurge:= ?BOOL? , MasterValueSource:= ?INT? , Options:= ?DWORD? , Valid=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD? , ValidFrequency=> ?LREAL? , FilterPool=> ?LREAL?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Master	Master axis	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)
Slave	Slave axis	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Enable	Enable	BOOL	Always	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_SmoothingFilter (Smoothing Filter) is executed.
FilterDisable	Filter disable	BOOL	Always	TRUE, FALSE	FALSE	The filter processing method is set. • FALSE: Operation is performed according to the filter setting. • TRUE: The filter is disabled and the Master axis (Master) input is transmitted to Slave axis (Slave) as it is. For details, refer to the following. ☞ Page 1824 Filter disable (FilterDisable)
Frequency	Frequency	LREAL	Always	0.20 to 250.00 [Hz]	10.00	The frequency to suppress vibration is set. For details, refer to the following. ☞ Page 1824 Frequency (Frequency)

Input variable	Name	Data type	Input	Valid range	Default value	Description
FilterPurge	Filter purge	BOOL	At start	TRUE, FALSE	FALSE	The end operation with the filter cumulative value is set. • FALSE: The slave axis stops immediately. • TRUE: The axis operates until the filter cumulative value becomes "0" and then stops. For details, refer to the following. ☞ Page 1825 Filter purge (FilterPurge)
MasterValueSource	Master axis data source selection	INT (MC_SOURCE)	At start	1, 2, 101, 102	1	The data source of Master axis (Master) is set. • 1: Set value (mcSetValue) • 2: Actual value (mcActualValue) • 101: Latest set value (mcLatestSetValue) • 102: Latest actual value (mcLatestActualValue) For details, refer to the following. ☞ Page 1826 Master axis data source selection (MasterValueSource)
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	Set "00000000H". *: If a value other than "00000000H" is set, Out of Options Range (error code: 1ABBH) occurs.

■Output variables

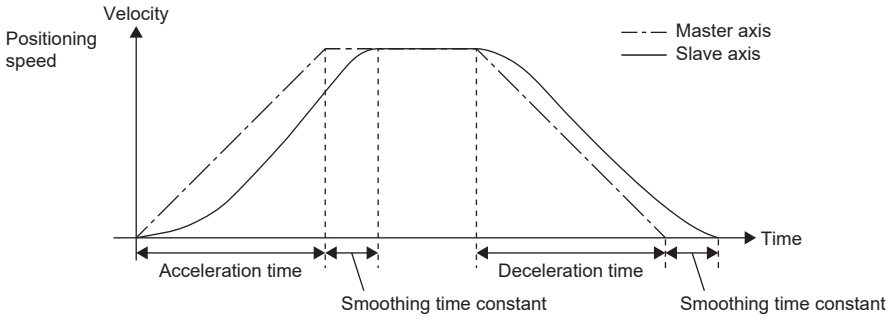
Output variable	Name	Data type	Default value	Description
Valid	Output value valid	BOOL	FALSE	This variable becomes TRUE when the output value is valid.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_SmoothingFilter (Smoothing Filter) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE when MCv_SmoothingFilter (Smoothing Filter) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MCv_SmoothingFilter (Smoothing Filter) is aborted.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used
ValidFrequency	Valid frequency	LREAL	0.00	This variable outputs the valid frequency.
FilterPool	Filter cumulated value	LREAL	0.0	This variable outputs the slave axis movement amount cumulated in filter processing. The held value is cleared when Enable (Enable) is set to TRUE. If the value of Filter cumulated value (FilterPool) exceeds the positioning range, Filter Cumulated Value Over Warning (event code: 00D13H) occurs. When this warning is detected, the error of Filter cumulated value (FilterPool) may increase.

■Public variables

Public variable	Name	Data type	Default value	Description
PurgedFilterPool	Disposed filter cumulated value	LREAL	0.0	When Filter disable (FilterDisable) is set to TRUE, this variable stores the disposed value from Filter cumulated value (FilterPool). Disposed filter cumulated value (PurgedFilterPool) is not cumulative. The value is cleared to 0 at start of the FB.

Processing details

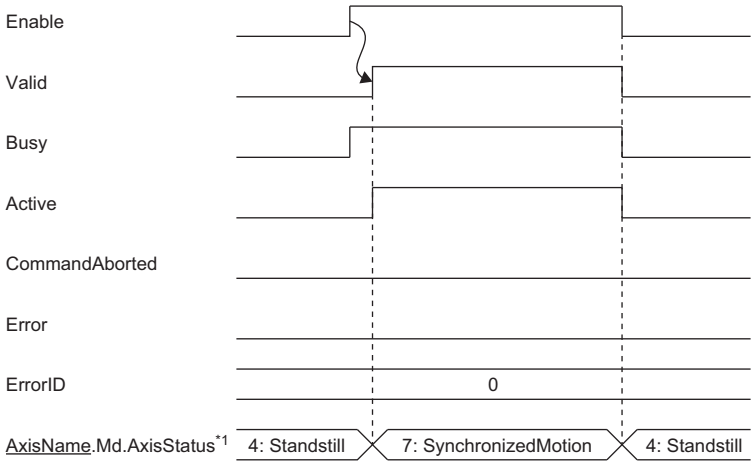
- This FB removes frequencies higher than the value set in Frequency (Frequency), and makes smooth acceleration/deceleration waveforms.
- Smoothing filter outputs the result of filter processing executed for the Master axis (Master) value set in Master axis data source selection (MasterValueSource) to Slave axis (Slave).
- Axis status (AxisName.Md.AxisStatus) of Slave axis (Slave) is "7: During synchronous operation (SynchronizedMotion)" during smoothing filter execution.
- The time constant of smoothing is "1/ Frequency (Frequency) [s]", and the acceleration time and deceleration time become longer by the amount of the smoothing time constant. When Frequency (Frequency) is 10 [Hz], the smoothing time constant is "0.1 [s] = 100 [ms]".



- If a smoothing filter is started while Slave axis (Slave) is in another operation FB, the start request will be ignored, and Start Not Possible (error code: 1AFCH) occurs. Start the smoothing filter when Axis status (AxisName.Md.AxisStatus) of Slave axis (Slave) is "4: Standby (Standstill)".
- MC_Stop (Forced Stop) is the only function block that can be multiple started during smoothing filter execution.

■Timing chart

- When the processing is completed successfully



*1 Axis status of Slave axis (Slave)

- When the processing is completed with an error

For the timing chart, refer to the following.

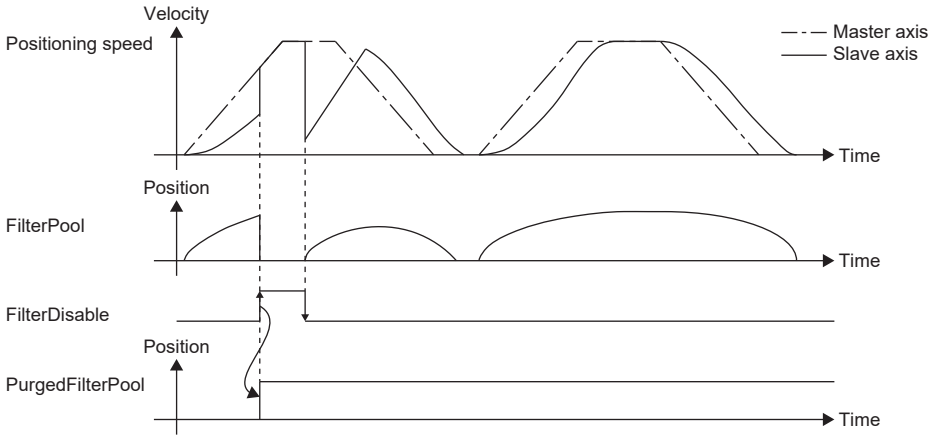
☞ Page 1390 Basic operation of Enable (Enable) type motion control FBs

■Filter disable (FilterDisable)

The method used for the filter processing is set.

Setting value	Description
FALSE	Operation is performed according to the filter setting.
TRUE	The filter is disabled and the Master axis (Master) input is transmitted to Slave axis (Slave) as it is.

- When Filter disable (FilterDisable) is set to TRUE, the value in Filter cumulated value (FilterPool) becomes "0.0" and the disposed value from Filter cumulated value (FilterPool) is stored in Disposed filter cumulated value (PurgedFilterPool).



■Frequency (Frequency)

The frequency to suppress vibration is set.

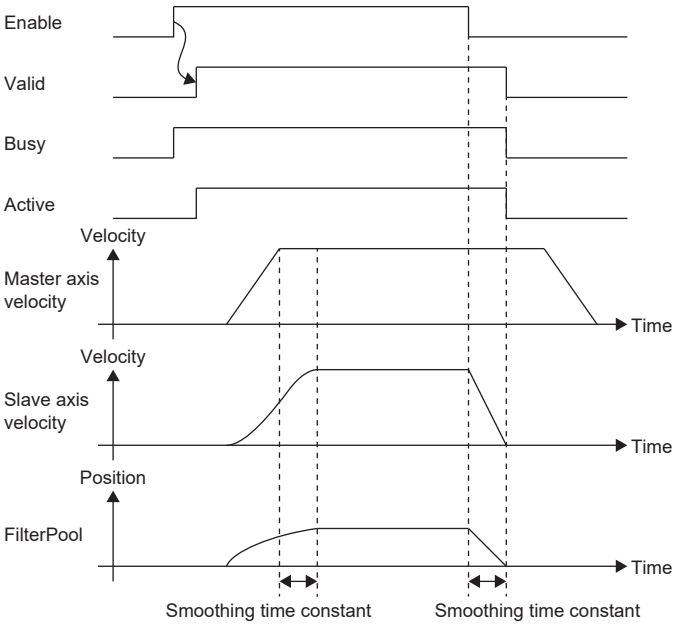
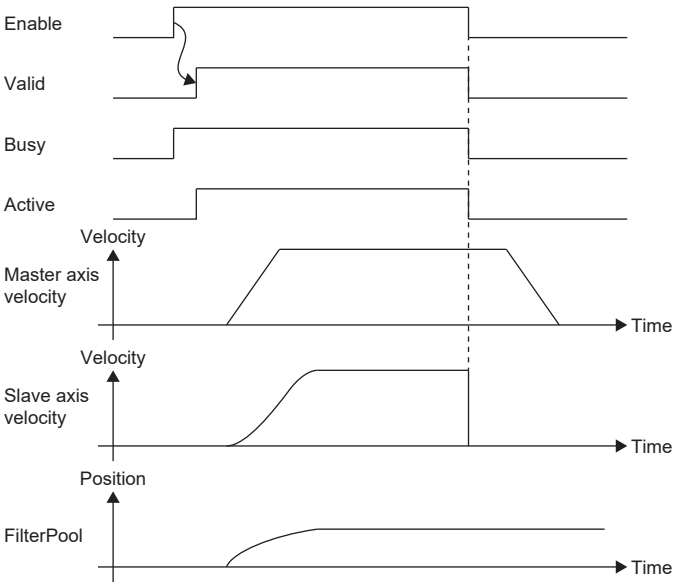
The following shows the settable range for Frequency (Frequency).

1/Slave axis operation cycle [s]	Setting range
250.00 [Hz] or more	0.20 to 250.00 [Hz]
Less than 250.00 [Hz]	0.20 to (1/Slave axis operation cycle [s]) [Hz]

- If a value outside the range is set in Frequency (Frequency), Out of Frequency Specification Range (error code: 1AE6H) occurs.
- Valid frequencies can be checked in Valid frequency (ValidFrequency).
- To change the frequency, change Frequency (Frequency) when Filter cumulated value (FilterPool) is "0.0". Even if it is changed in a status other than "0.0", the change will not reflected until Filter cumulated value (FilterPool) becomes "0.0".
- Frequency (Frequency) cannot be changed to a frequency lower than Valid frequency (ValidFrequency). If a lower frequency is set, Out of Frequency Specification Range (error code: 1AE6H) occurs.
- Filter cumulated value (FilterPool) may become "0.0" while the master axis operates with the operation pattern which repeats positive/reverse rotation. When the value of Frequency (Frequency) is changed while the filter operation is not settled, the filter operation will be stopped in the middle. Thus, it causes the deviation between the master axis and the slave axis. To change Frequency (Frequency), check the master axis stop and wait for the filter time constant before changing.

■Filter purge (FilterPurge)

Set the operation for when Enable (Enable) is set to FALSE and Filter cumulated value (FilterPool) is other than "0.0".

Setting value	Description
TRUE	When Enable (Enable) becomes FALSE, the axis operates until Filter cumulated value (FilterPool) reaches "0.0" and then stops. 
FALSE	When Enable (Enable) becomes FALSE, immediately stops the slave axis regardless of the Filter cumulated value (FilterPool) status. 

■Master axis data source selection (MasterValueSource)

Slave axis (Slave) sets the position type of Master axis (Master) that will execute single axis synchronous control.
The operation may differ depending on the execution order of the FBs or if the operation cycles of Master axis (Master) and Slave axis (Slave) are different.
Operation is performed with the same value as the set value when specifying Actual value in Master axis data source selection (MasterValueSource) while an axis type other than real axis is set in Master axis (Master).

Setting value	Description
1: Set value (mcSetValue)	Uses the commanded position of the master axis in the previous operation cycle.
2: Actual value (mcActualValue)	Uses the actual position of the master axis in the previous operation cycle.
101: Latest set value (mcLatestSetValue)	Uses the commanded position of the master axis in the current operation cycle.
102: Latest actual value (mcLatestActualValue)	Uses the actual position of the master axis in the current operation cycle.

Point

When Master axis data source selection (MasterValueSource) is set to "1: Set value (mcSetValue)" or "101: Latest set value (mcLatestSetValue)" and the master axis becomes servo OFF by a servo alarm or forced stop, the amount of the value change may be too large. This can be prevented by setting Master axis data source selection (MasterValueSource) to "2: Actual value (mcActualValue)" or "102: Latest actual value (mcLatestActualValue)".

Precautions

- For the smoothing filter, the operation load increases as the filter level increases. A rough standard for the operation load when the filter level is 5000 is as follows.

Module used	Operation load
MXR300-16 MXR300-32 MXR300-64	170μs
MXR500-128 MXR500-256	130μs

- The filter level is calculated with the following formula. Digits after the decimal point are rounded/up down to the nearest integer. When the result of rounding is "0", the operation is carried out as 1 level.

Filter level = (1 / frequency [Hz]) / operation cycle [s]

- The smoothing filter secures the memory to use for filter processing at start. The memory usage can be calculated with the following formula. If memory is insufficient, change Maximum RAM size (AddonSystem.PrConst.MT_MotionControl_AxisFilter.RamSizeMax) of Addon MotionControl_AxisFilter parameter (Addon_MotionControl_AxisFilter).

Memory usage = Filter level × 8 [Bytes]

- The operation error may increase when the movement amount of the master axis or the slave axis is large and the following case applies. Review the unit setting of the master axis and the slave axis and decrease the movement range. If the error increases, clear the synchronization status or execute the current value change of absolute position specification.
  - When the total sum of the movement amount (the following formula) equivalent to filter level at control exceeds the number of significant digits which can be represented by double-precision floating-point (approximately 15 digits).

$$\sum_{i=0}^{\text{Filter level - 1}} (\text{i Position of the master axis before the operation cycle - Position of the master axis at the start of control})$$

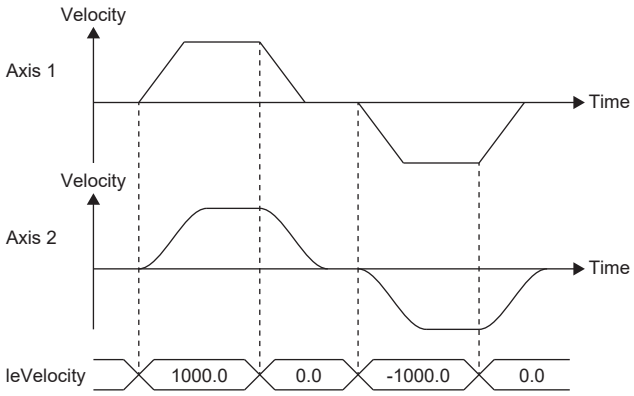
- If Filter disable (FilterDisable) is switched during operation, the Slave axis (Slave) velocity may change rapidly.
- After enabling the command filter, when the Master axis (Master) command is not transmitted to Slave axis (Slave) or the back part synchronous control FB because the stop factor occurred on Slave axis (Slave), the synchronous position relation deviates. Before enabling the command filter again, execute the synchronous positioning as needed.
- When using the mixed operation cycles function, there are some restrictions if the control operation cycle setting of the master axis and slave axis are not the same. For details, refer to each function and the following.

User's manual for the controller used

Program example

The following is a program example. The filter processing that changes the trapezoidal acceleration/deceleration to a smooth acceleration/deceleration waveform is performed.

• Operation



• Settings

Item		Setting value
Frequency		10.00
Data settings for speed control	Velocity	5000.0
	Acceleration	100.0
	Deceleration	100.0
	Jerk	0.0

■Axes

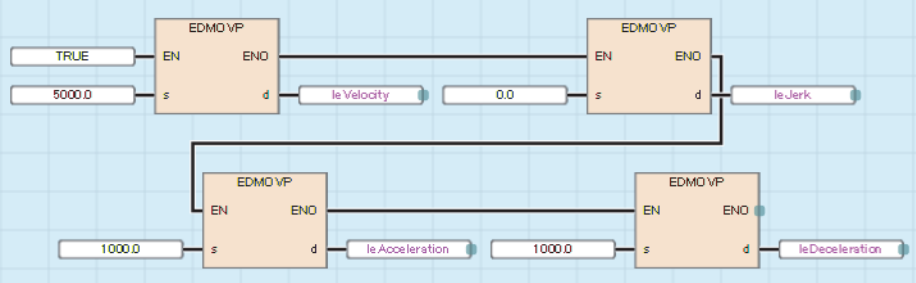
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1
2	Axis0002	AXIS_REF	Axis 2

■Labels used

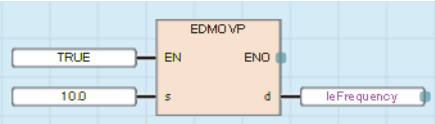
Label name	Data type	Comment
MCv_SpeedControl_1	MCv_SpeedControl	Speed Control (Including Position Loop) FB
bSpeedControl	Bit	Speed control command
leVelocity	Double-precision real number	Command velocity
leAcceleration	Double-precision real number	Acceleration
leDeceleration	Double-precision real number	Deceleration
leJerk	Double-precision real number	Jerk
bSpeedInVelocity	Bit	Target velocity reached
bSpeedBusy	Bit	Executing
bSpeedActive	Bit	Controlling
bSpeedCommandAborted	Bit	Abortion of execution
bSpeedError	Bit	Error
uwSpeedErrorID	Word [unsigned]/bit string [16-bit]	Error code
MCv_SmoothingFilter_1	MCv_SmoothingFilter	Smoothing Filter FB
bSmoothingFilter	Bit	Smoothing filter command
leFrequency	Double-precision real number	Frequency
bSmoothingValid	Bit	Output value valid
bSmoothingInVelocity	Bit	Target velocity reached
bSmoothingBusy	Bit	Executing
bSmoothingActive	Bit	Controlling
bSmoothingCommandAborted	Bit	Abortion of execution
bSmoothingError	Bit	Error
uwSmoothingErrorID	Word [unsigned]/bit string [16-bit]	Error code
leSmoothingValidFrequency	Double-precision real number	Valid frequency
leSmoothingFilterPool	Double-precision real number	Filter cumulated value

■FBD/LD program

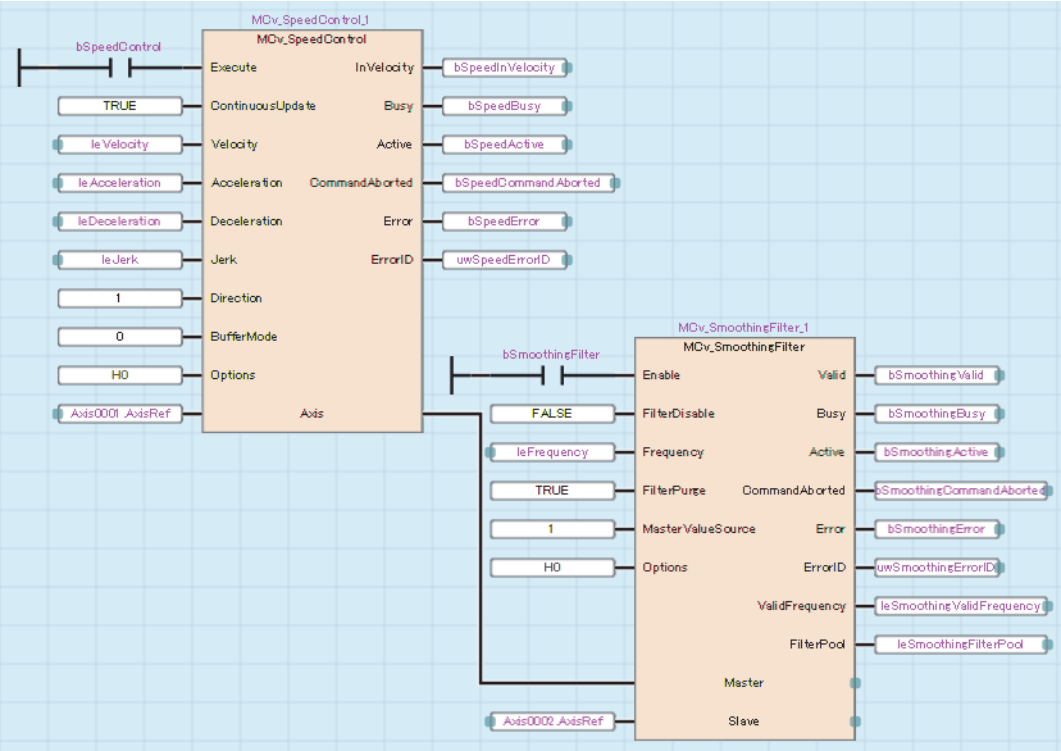
• Data settings for speed control



• Frequency



• Speed Control/Smoothing Filter



■ST program

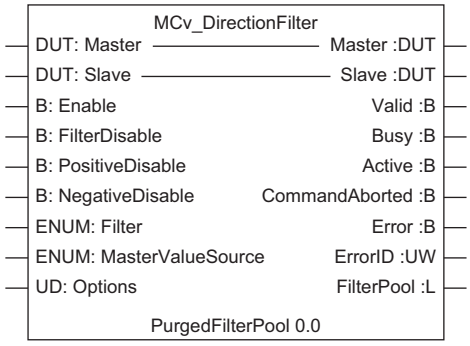
```
//-----Data settings for speed control-----
leVelocity:= 5000.0;
leAcceleration:= 1000.0;
leDeceleration:= 1000.0;
leJerk:= 0.0;

//-----Speed Control-----
MCv_SpeedControl_1(
    Axis:= Axis0001.AxisRef ,
    Execute:= bSpeedControl ,
    ContinuousUpdate:= TRUE ,
    Velocity:= leVelocity ,
    Acceleration:= leAcceleration ,
    Deceleration:= leDeceleration ,
    Jerk:= leJerk ,
    Direction:= MC_DIRECTION__mcPositiveDirection ,
    BufferMode:= MC_BUFFER_MODE__mcAborting ,
    Options:= H00000000 ,
    InVelocity=> bSpeedInVelocity ,
    Busy=> bSpeedBusy ,
    Active=> bSpeedActive ,
    CommandAborted=> bSpeedCommandAborted ,
    Error=> bSpeedError ,
    ErrorID=> uwSpeedErrorID
);

//-----Frequency-----
leFrequency:= 10.00;

//-----Smoothing Filter-----
MCv_SmoothingFilter_1(
    Master:= MCv_SpeedControl_1.Axis ,
    Slave:= Axis0002.AxisRef ,
    Enable:= bSmoothingFilter ,
    FilterDisable:= FALSE ,
    Frequency:= leFrequency ,
    FilterPurge:= TRUE ,
    MasterValueSource:= MC_SOURCE__mcSetValue ,
    Options:= H00000000 ,
    Valid=> bSmoothingValid ,
    Busy=> bSmoothingBusy ,
    Active=> bSmoothingActive ,
    CommandAborted=> bSmoothingCommandAborted ,
    Error=> bSmoothingError ,
    ErrorID=> uwSmoothingErrorID ,
    ValidFrequency=> leSmoothingValidFrequency ,
    FilterPool=> leSmoothingFilterPool
);
```

42.20 Moving Direction Restriction Filter

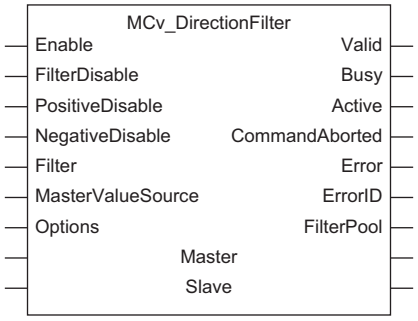
MCv_DirectionFilter

This FB performs filter processing to restrict traveling for the set movement direction.

Ladder



FBD/LD



ST

```
MCv_DirectionFilter(
  Master:= ?AXIS_REF? ,
  Slave:= ?AXIS_REF? ,
  Enable:= ?BOOL? ,
  FilterDisable:= ?BOOL? ,
  PositiveDisable:= ?BOOL? ,
  NegativeDisable:= ?BOOL? ,
  Filter:= ?INT? ,
  MasterValueSource:= ?INT? ,
  Options:= ?DWORD? ,
  Valid=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD? ,
  FilterPool=> ?LREAL?
);
```

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Master	Master axis	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)
Slave	Slave axis	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Enable	Enable	BOOL	Always	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_DirectionFilter (Moving Direction Restriction Filter) is executed.
FilterDisable	Filter disable	BOOL	Always	TRUE, FALSE	FALSE	The filter processing method is set. • FALSE: Operation is performed according to the filter setting. • TRUE: The filter is disabled and the Master axis (Master) input is transmitted to Slave axis (Slave) as it is. For details, refer to the following. ☞ Page 1834 Filter disable (FilterDisable)
PositiveDisable	Positive direction limit	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, the axis operation to the positive direction is limited. Slave axis (Slave) does not operate at Master axis (Master) forward rotation operation. When both Positive direction limit (PositiveDisable) and Negative direction limit (NegativeDisable) are set to TRUE, Slave axis (Slave) will not operate even if Master axis (Master) is operated. • FALSE: Not restricted • TRUE: Restricted

Input variable	Name	Data type	Input	Valid range	Default value	Description
NegativeDisable	Negative direction limit	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, the axis operation to the negative direction is limited. Slave axis (Slave) does not operate at Master axis (Master) reverse rotation operation. When both Positive direction limit (PositiveDisable) and Negative direction limit (NegativeDisable) are set to TRUE, Slave axis (Slave) will not operate even if Master axis (Master) is operated. • FALSE: Not restricted • TRUE: Restricted
Filter	Filter operation	INT (MC_VELOCITY_LIMIT_MODE)	At start	0, 2, 4	2	The filter operation is set. • 0: Ignore (Ignore) • 2: Truncate (Truncate) • 4: Clamp (Without Ramp at Deceleration) (ClampWithoutRamp) For details, refer to the following. ☞ Page 1835 Filter operation (Filter)
MasterValueSource	Master axis data source selection	INT (MC_SOURCE)	At start	1, 2, 101, 102	1	The data source of Master axis (Master) is set. • 1: Set value (mcSetValue) • 2: Actual value (mcActualValue) • 101: Latest set value (mcLatestSetValue) • 102: Latest actual value (mcLatestActualValue) For details, refer to the following. ☞ Page 1836 Master axis data source selection (MasterValueSource)
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	Set "00000000H". *: If a value other than "00000000H" is set, Out of Options Range (error code: 1ABBH) occurs.

■Output variables

Output variable	Name	Data type	Default value	Description
Valid	Output value valid	BOOL	FALSE	This variable becomes TRUE when the output value is valid.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_DirectionFilter (Moving Direction Restriction Filter) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE when MCv_DirectionFilter (Moving Direction Restriction Filter) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MCv_DirectionFilter (Moving Direction Restriction Filter) is aborted.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used
FilterPool	Filter cumulated value	LREAL	0.0	This variable outputs the slave axis movement amount cumulated in filter processing. When Enable (Enable) is set to FALSE while a value of this variable is other than "0.0", the value when Enable (Enable) was set to FALSE is held. The held value is cleared when Enable (Enable) is set to TRUE. If the value of Filter cumulated value (FilterPool) exceeds the positioning range, Filter Cumulated Value Over Warning (event code: 00D13H) occurs. When this warning is detected, the error of Filter cumulated value (FilterPool) may increase.

■Public variables

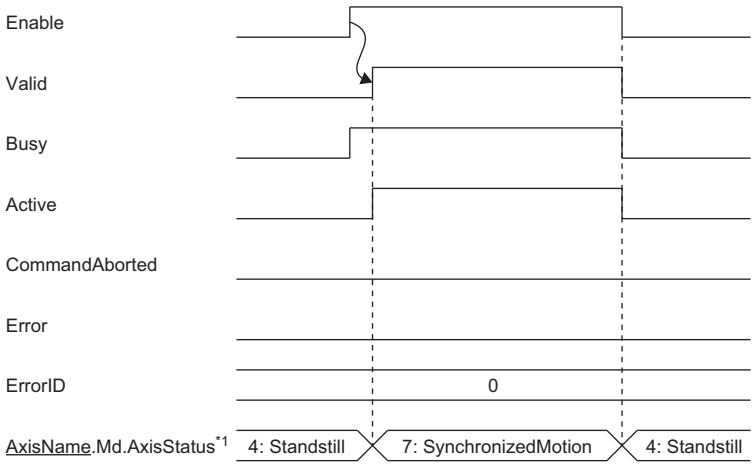
Public variable	Name	Data type	Default value	Description
PurgedFilterPool	Disposed filter cumulated value	LREAL	0.0	When Filter disable (FilterDisable) is set to TRUE, this variable stores the disposed value from Filter cumulated value (FilterPool). Disposed filter cumulated value (PurgedFilterPool) is not cumulative. The value is cleared to 0 at start of the FB.

Processing details

- This FB restricts the movement of the slave axis to one direction in relation to the master axis movement.
- Moving direction limit filter takes the result of filter processing executed for the Master axis (Master) value set in Master axis data source selection (MasterValueSource) and outputs it to Slave axis (Slave).
- Axis status (AxisName.Md.AxisStatus) of Slave axis (Slave) is "7: During synchronous operation (SynchronizedMotion)" during moving direction filter execution.
- If a moving direction limit filter is started while Slave axis (Slave) is in another operation FB, the start request will be ignored, and Start Not Possible (error code: 1AFCH) occurs. Start the moving direction limit filter when Axis status (AxisName.Md.AxisStatus) of Slave axis (Slave) is "4: Standby (Standstill)".
- MC_Stop (Forced Stop) is the only function block that can be multiple started during smoothing filter execution.
- When setting Enable (Enable) to FALSE, perform this function block while the slave axis is in stop status. If Enable (Enable) is set to FALSE while the slave axis is in operation, the slave axis will stop immediately.

■Timing chart

- When the processing is completed successfully



*1 Axis status of Slave axis (Slave)

- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1390 Basic operation of Enable (Enable) type motion control FBs

■Filter disable (FilterDisable)

The method used for the filter processing is set.

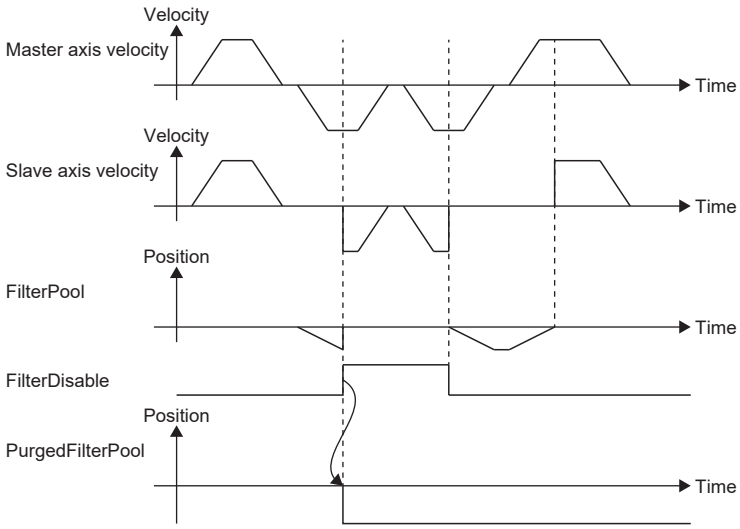
Setting value	Description
FALSE	Operation is performed according to the filter setting.
TRUE	The filter is disabled and the Master axis (Master) input is transmitted to Slave axis (Slave) as it is.

- When Filter disable (FilterDisable) is set to TRUE, the value in Filter cumulated value (FilterPool) becomes "0.0" and the disposed value from Filter cumulated value (FilterPool) is stored in Disposed filter cumulated value (PurgedFilterPool).

Ex.

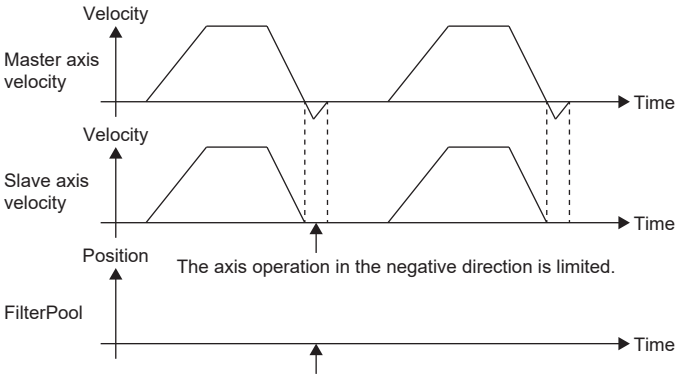
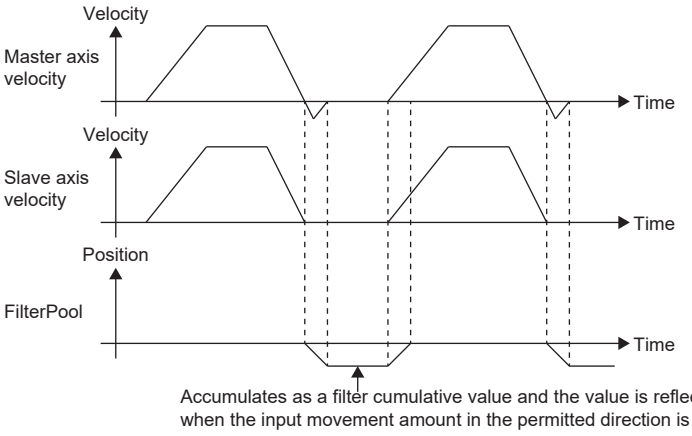
When set as shown below

Input pin	Setting value
Filter operation (Filter)	4: Clamp (Without Ramp at Deceleration) (ClampWithoutRamp)
Positive direction limit (PositiveDisable)	FALSE (Not limited)
Negative direction limit (NegativeDisable)	TRUE (Limited)



■Filter operation (Filter)

• The slave axis operation at control is set based on the setting of Filter operation (Filter).

Setting value	Description
0: Ignore (Ignore)	• The movement direction is not restricted. • Filter cumulated value (FilterPool) is not updated.
2: Truncate (Truncate)	• The movement amount to the restricted direction is ignored and is not added to Filter cumulated value (FilterPool). • The movement amount to the restricted direction will be truncated, and the position relation between the master axis and the slave axis will deviate by an amount equal to the movement amount to the restricted direction. Example: When Positive direction limit (PositiveDisable) is FALSE, and Negative direction limit (NegativeDisable) is TRUE 
4: Clamp (Without Ramp at Deceleration) (ClampWithoutRamp)	• The movement amount to the restricted direction is added to Filter cumulated value (FilterPool). • Filter cumulated value (FilterPool) reflects the reverse direction movement. Thus, even if operation to the restricted direction is repeated, the position relation between the master axis and the slave axis will not deviate from that when the status of Filter cumulated value (FilterPool) is "0.0". Example: When Positive direction limit (PositiveDisable) is FALSE (Not limited), and Negative direction limit (NegativeDisable) is TRUE (Limited) 

• If a value outside the range is set in Filter operation (Filter), Out of Filter Operation Specification Range (error code: 1AE7H) occurs.

■Master axis data source selection (MasterValueSource)

Slave axis (Slave) sets the position type of Master axis (Master) that will execute single axis synchronous control.
The operation may differ depending on the execution order of the FBs or if the operation cycles of Master axis (Master) and Slave axis (Slave) are different.
Operation is performed with the same value as the set value when specifying Actual value in Master axis data source selection (MasterValueSource) while an axis type other than real axis is set in Master axis (Master).

Setting value	Description
1: Set value (mcSetValue)	Uses the commanded position of the master axis in the previous operation cycle.
2: Actual value (mcActualValue)	Uses the actual position of the master axis in the previous operation cycle.
101: Latest set value (mcLatestSetValue)	Uses the commanded position of the master axis in the current operation cycle.
102: Latest actual value (mcLatestActualValue)	Uses the actual position of the master axis in the current operation cycle.

Point

When Master axis data source selection (MasterValueSource) is set to "1: Set value (mcSetValue)" or "101: Latest set value (mcLatestSetValue)" and the master axis becomes servo OFF by a servo alarm or forced stop, the amount of the value change may be too large. This can be prevented by setting Master axis data source selection (MasterValueSource) to "2: Actual value (mcActualValue)" or "102: Latest actual value (mcLatestActualValue)".

Precautions

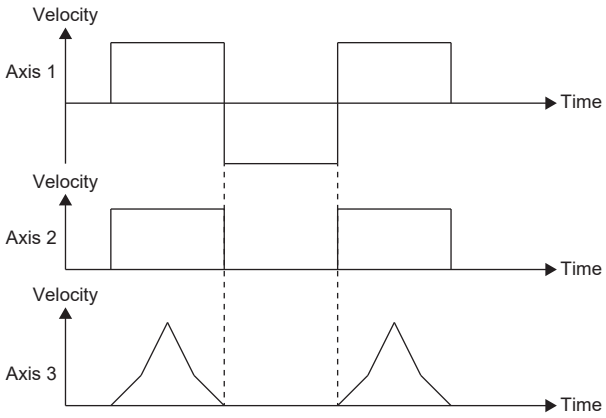
- If Filter disable (FilterDisable) is switched during operation, the Slave axis (Slave) velocity may change rapidly.
- After enabling the command filter, when the Master axis (Master) command is not transmitted to Slave axis (Slave) or the back part synchronous control FB because the stop factor occurred on Slave axis (Slave), the synchronous position relation deviates. Before enabling the command filter again, execute the synchronous positioning as needed.
- When using the mixed operation cycles function, there are some restrictions if the control operation cycle setting of the master axis and slave axis are not the same. For details, refer to each function and the following.

User's manual for the controller used

Program example

The following is a program example. The filter processing that restricts the direction of the cam operation is performed.

- Operation



- Settings

Item	Setting value
Cam ID setting	1

■Axes

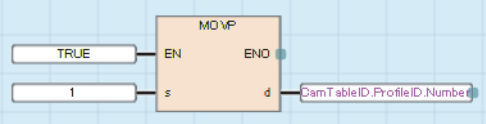
Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REF	Axis 1
2	Axis0002	AXIS_REF	Axis 2
3	Axis0003	AXIS_REF	Axis 3

■Labels used

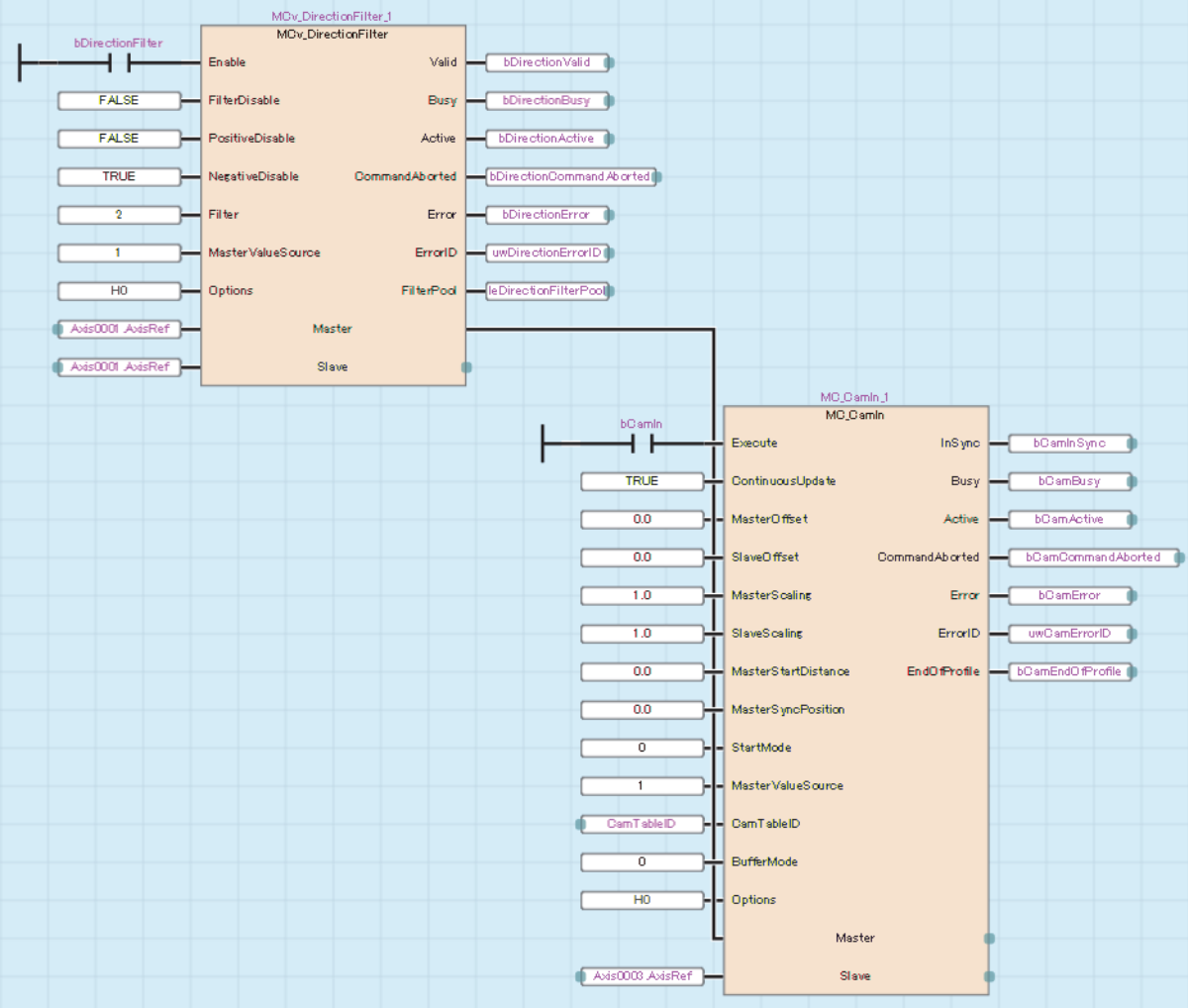
Label name	Data type	Comment
MCv_DirectionFilter_1	MCv_DirectionFilter	Moving Direction Restriction Filter FB
bDirectionFilter	Bit	Moving direction restriction filter command
bDirectionValid	Bit	Output value valid
bDirectionBusy	Bit	Executing
bDirectionActive	Bit	Controlling
bDirectionCommandAborted	Bit	Abortion of execution
bDirectionError	Bit	Error
uwDirectionErrorID	Word [unsigned]/bit string [16-bit]	Error code
leDirectionFilterPool	Double-precision real number	Disposed filter cumulated value
MC_CamIn_1	MC_CamIn	Cam Operation Start FB
bCamIn	Bit	Cam operation command
CamTableID	MC_CAM_ID	Cam ID
bCamInSync	Bit	In synchronization
bCamBusy	Bit	Executing
bCamActive	Bit	Controlling
bCamCommandAborted	Bit	Abortion of execution
bCamError	Bit	Error
uwCamErrorID	Word [unsigned]/bit string [16-bit]	Error code
bCamEndOfProfile	Bit	Cam cycle completion

■FBD/LD program

• Cam ID setting



• Moving Direction Restriction Filter/cam operation



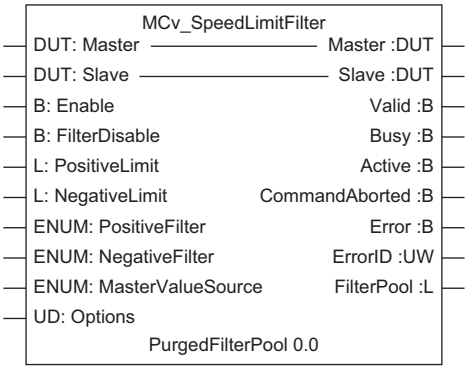
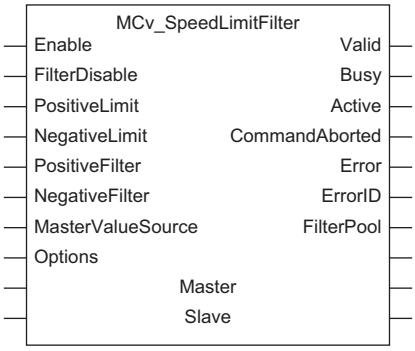
■ST program

```
//-----Moving Direction Restriction Filter-----
MCv_DirectionFilter_1(
    Master:= Axis0001.AxisRef ,
    Slave:= Axis0002.AxisRef ,
    Enable:= bDirectionFilter ,
    FilterDisable:= FALSE ,
    PositiveDisable:= FALSE ,
    NegativeDisable:= TRUE ,
    Filter:= MC_VELOCITY_LIMIT_MODE__Truncate ,
    MasterValueSource:= MC_SOURCE__mcSetValue ,
    Options:= H00000000 ,
    Valid=> bDirectionValid ,
    Busy=> bDirectionBusy ,
    Active=> bDirectionActive ,
    CommandAborted=> bDirectionCommandAborted ,
    Error=> bDirectionError ,
    ErrorID=> uwDirectionErrorID ,
    FilterPool=> leDirectionFilterPool
);

//Cam ID settings
CamTableID.ProfileID.Number:= 1;

//Cam operation
MC_CamIn_1(
    Master:= MCv_DirectionFilter_1.Slave ,
    Slave:= Axis0003.AxisRef ,
    Execute:= bCamIn ,
    ContinuousUpdate:= TRUE ,
    MasterOffset:= 0.0 ,
    SlaveOffset:= 0.0 ,
    MasterScaling:= 1.0 ,
    SlaveScaling:= 1.0 ,
    MasterStartDistance:= 0.0 ,
    MasterSyncPosition:= 0.0 ,
    StartMode:= MC_START_MODE__mcImmediate ,
    MasterValueSource:= MC_SOURCE__mcSetValue ,
    CamTableID:= CamTableID ,
    BufferMode:= MC_BUFFER_MODE__mcAborting ,
    Options:= H00000000 ,
    InSync=> bCamInSync ,
    Busy=> bCamBusy ,
    Active=> bCamActive ,
    CommandAborted=> bCamCommandAborted ,
    Error=> bCamError ,
    ErrorID=> uwCamErrorID ,
    EndOfProfile=> bCamEndOfProfile
);
```

42.21 Speed Limit Filter

MCv_SpeedLimitFilter

This FB performs filter processing to restrict to the velocity of the set limit value.

Ladder	FBD/LD	ST
		MCv_SpeedLimitFilter(Master:= ?AXIS_REF? , Slave:= ?AXIS_REF? , Enable:= ?BOOL? , FilterDisable:= ?BOOL? , PositiveLimit:= ?LREAL? , NegativeLimit:= ?LREAL? , PositiveFilter:= ?INT? , NegativeFilter:= ?INT? , MasterValueSource:= ?INT? , Options:= ?DWORD? , Valid=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD? , FilterPool=> ?LREAL?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Master	Master axis	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)
Slave	Slave axis	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Enable	Enable	BOOL	Always	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_SpeedLimitFilter (Speed Limit Filter) is executed.
FilterDisable	Filter disable	BOOL	Always	TRUE, FALSE	FALSE	The filter processing method is set. • FALSE: Operation is performed according to the filter setting. • TRUE: The filter is disabled and the Master axis (Master) input is transmitted to Slave axis (Slave) as it is. For details, refer to the following. ☞ Page 1843 Filter disable (FilterDisable)
PositiveLimit	Positive direction limit value	LREAL	At start	0.0, 0.0001 to 2500000000.0	0.0	The speed limit value of the positive direction is set. The upper limit value of the velocity output to Slave axis (Slave) is set in relation to the Master axis (Master) positive direction input velocity. When "0.0" is set, Slave axis does not operate to the positive direction. If a value outside the range is set, Out of Direction Limit Value Specification Range (error code: 1AE8H) occurs.

Input variable	Name	Data type	Input	Valid range	Default value	Description
NegativeLimit	Negative direction limit value	LREAL	At start	0.0, 0.0001 to 2500000000.0	0.0	The speed limit value of the negative direction is set. The upper limit value of the velocity output to Slave axis (Slave) is set in relation to the Master axis (Master) negative direction input velocity. When "0.0" is set, Slave axis does not operate to the negative direction. If a value outside the range is set, Out of Direction Limit Value Specification Range (error code: 1AE8H) occurs.
PositiveFilter	Positive direction filter operation	INT (MC_VELOCITY_LIMIT_MODE)	At start	0, 2, 4	2	The filter operation when the positive direction limit is exceeded is set. • 0: Ignore (Ignore) • 2: Truncate (Truncate) • 4: Clamp (Without Ramp at Deceleration) (ClampWithoutRamp) For details, refer to the following. ☞ Page 1843 Positive direction filter operation (PositiveFilter)/ Negative direction filter operation (NegativeFilter)
NegativeFilter	Negative direction filter operation	INT (MC_VELOCITY_LIMIT_MODE)	At start	0, 2, 4	2	The filter operation when the negative direction limit is exceeded is set. • 0: Ignore (Ignore) • 2: Truncate (Truncate) • 4: Clamp (Without Ramp at Deceleration) (ClampWithoutRamp) For details, refer to the following. ☞ Page 1843 Positive direction filter operation (PositiveFilter)/ Negative direction filter operation (NegativeFilter)
MasterValueSource	Master axis data source selection	INT (MC_SOURCE)	At start	1, 2, 101, 102	1	The data source of Master axis (Master) is set. • 1: Set value (mcSetValue) • 2: Actual value (mcActualValue) • 101: Latest set value (mcLatestSetValue) • 102: Latest actual value (mcLatestActualValue) For details, refer to the following. ☞ Page 1844 Master axis data source selection (MasterValueSource)
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	Set "00000000H". *: If a value other than "00000000H" is set, Out of Options Range (error code: 1ABBH) occurs.

■Output variables

Output variable	Name	Data type	Default value	Description
Valid	Output value valid	BOOL	FALSE	This variable becomes TRUE when the output value is valid.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_SpeedLimitFilter (Speed Limit Filter) is executed.
Active	Controlling	BOOL	FALSE	This variable indicates that the FB is controlling the axis. The variable becomes TRUE when MCv_SpeedLimitFilter (Speed Limit Filter) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MCv_SpeedLimitFilter (Speed Limit Filter) is aborted.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used
FilterPool	Filter cumulated value	LREAL	0.0	This variable outputs the slave axis movement amount cumulated in filter processing. When Enable (Enable) is set to FALSE while a value of this variable is other than "0.0", the value when Enable (Enable) was set to FALSE is held. The held value is cleared when Enable (Enable) is set to TRUE. If the value of Filter cumulated value (FilterPool) exceeds the positioning range, Filter Cumulated Value Over Warning (event code: 00D13H) occurs. When this warning is detected, the error of Filter cumulated value (FilterPool) may increase.

■Public variables

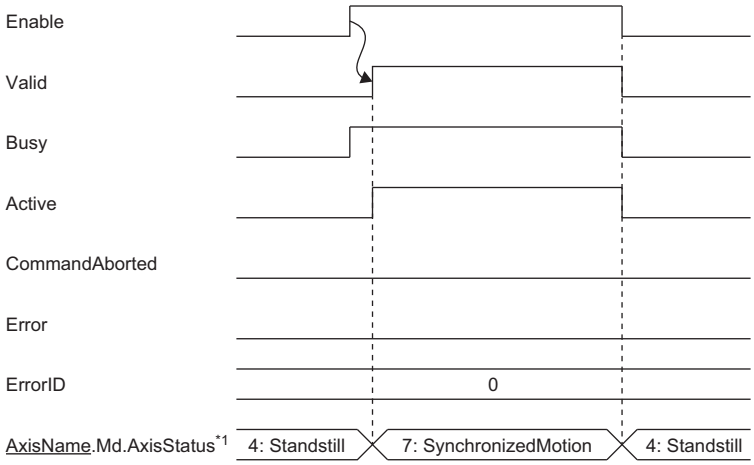
Public variable	Name	Data type	Default value	Description
PurgedFilterPool	Disposed filter cumulated value	LREAL	0.0	When Filter disable (FilterDisable) is set to TRUE, this variable stores the disposed value from Filter cumulated value (FilterPool). Disposed filter cumulated value (PurgedFilterPool) is not cumulative. The value is cleared to 0 at start of the FB.

Processing details

- This FB sets a specified control value to the input velocity of the master axis, and outputs the velocity of the set control value to the slave axis.
- Speed limit filter outputs the result of filter processing executed for the master axis (Master) value set in Master axis data source selection (MasterValueSource) to the slave axis (Slave).
- Axis status (AxisName.Md.AxisStatus) of Slave axis (Slave) is "7: During synchronous operation (SynchronizedMotion)" during speed limit filter execution.
- If a speed limit filter is started while Slave axis (Slave) is in another operation FB, the start request will be ignored, and Start Not Possible (error code: 1AFCH) occurs. Start the speed limit filter when Axis status (AxisName.Md.AxisStatus) of Slave axis (Slave) is "4: Standby (Standstill)".
- MC_Stop (Forced Stop) is the only function block that can be multiple started during speed limit filter execution.
- When setting Enable (Enable) to FALSE, perform this function block while the slave axis is in stop status. If Enable (Enable) is set to FALSE while the slave axis is in operation, the slave axis will stop immediately.

■Timing chart

- When the processing is completed successfully



*1 Axis status of Slave axis (Slave)

- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1390 Basic operation of Enable (Enable) type motion control FBs

■Filter disable (FilterDisable)

The method used for the filter processing is set.

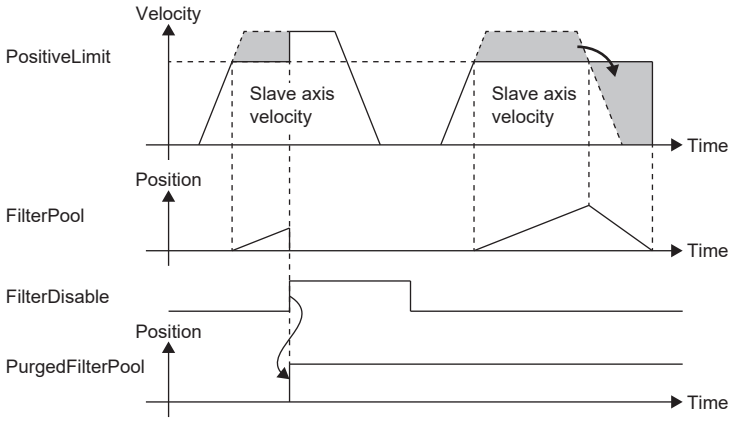
Setting value	Description
FALSE	Operation is performed according to the filter setting.
TRUE	The filter is disabled and the Master axis (Master) input is transmitted to Slave axis (Slave) as it is.

- When Filter disable (FilterDisable) is set to TRUE, the value in Filter cumulated value (FilterPool) becomes "0.0" and the disposed value from Filter cumulated value (FilterPool) is stored in Disposed filter cumulated value (PurgedFilterPool).

Ex.

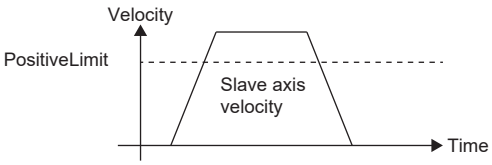
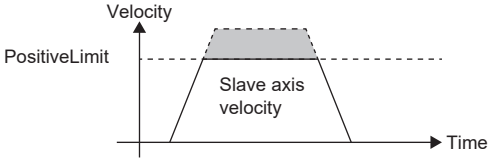
When set as shown below

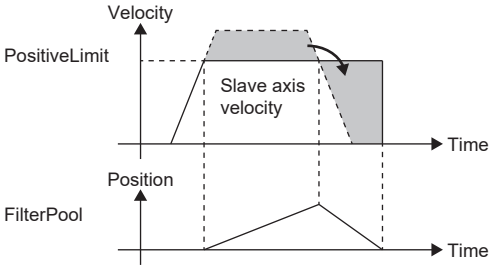
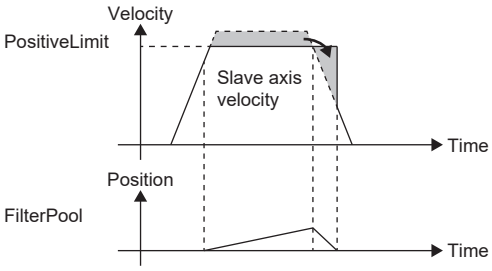
Input pin	Setting value
Positive direction filter operation (PositiveFilter)	4: Clamp (Without Ramp at Deceleration) (ClampWithoutRamp)



■Positive direction filter operation (PositiveFilter)/Negative direction filter operation (NegativeFilter)

- The slave axis operation at control is set based on the setting of Positive direction filter operation (PositiveFilter) and Negative direction filter operation (NegativeFilter).

Setting value	Description
0: Ignore (Ignore)	• The velocity is not limited. • Filter cumulated value (FilterPool) is not updated. 
2: Truncate (Truncate)	• The amount exceeding the control value is not output to Slave axis (Slave). • The restricted movement amount will be truncated, and the position relation between the master axis and the slave axis will deviate by the truncated amount. 

Setting value	Description
4: Clamp (Without Ramp at Deceleration) (ClampWithoutRamp)	• The amount exceeding the control value is cumulated as Filter cumulated value (FilterPool) and output later. The Filter cumulated value (FilterPool) output does not operate deceleration. • When "0.0" is set as the control value, all the master axis input to the applicable direction will be cumulated as Filter cumulated value (FilterPool) and will not be output later to the slave axis. The cumulated amount will be subtracted by the reverse direction master axis input, and the slave axis will not operate until Filter cumulated value (FilterPool) is "0.0". ■When Filter cumulated value is greater than the decelerated distance  ■When Filter cumulated value is smaller than the decelerated distance 

- If a value outside the range is set in Filter operation (Filter), Out of Filter Operation Specification Range (error code: 1AE7H) occurs.

■Master axis data source selection (MasterValueSource)

Slave axis (Slave) sets the position type of Master axis (Master) that will execute single axis synchronous control.
The operation may differ depending on the execution order of the FBs or if the operation cycles of Master axis (Master) and Slave axis (Slave) are different.
Operation is performed with the same value as the set value when specifying Actual value in Master axis data source selection (MasterValueSource) while an axis type other than real axis is set in Master axis (Master).

Setting value	Description
1: Set value (mcSetValue)	Uses the commanded position of the master axis in the previous operation cycle.
2: Actual value (mcActualValue)	Uses the actual position of the master axis in the previous operation cycle.
101: Latest set value (mcLatestSetValue)	Uses the commanded position of the master axis in the current operation cycle.
102: Latest actual value (mcLatestActualValue)	Uses the actual position of the master axis in the current operation cycle.

Point

When Master axis data source selection (MasterValueSource) is set to "1: Set value (mcSetValue)" or "101: Latest set value (mcLatestSetValue)" and the master axis becomes servo OFF by a servo alarm or forced stop, the amount of the value change may be too large. This can be prevented by setting Master axis data source selection (MasterValueSource) to "2: Actual value (mcActualValue)" or "102: Latest actual value (mcLatestActualValue)".

Precautions

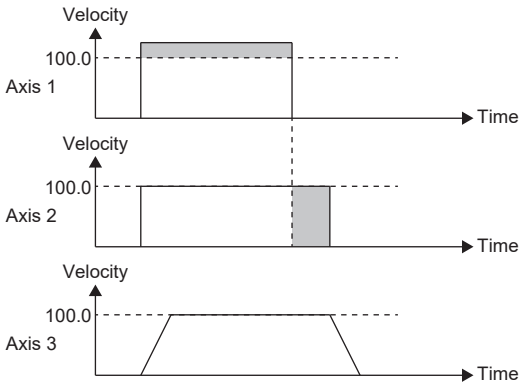
- If Filter disable (FilterDisable) is switched during operation, the Slave axis (Slave) velocity may change rapidly.
- After enabling the command filter, when the Master axis (Master) command is not transmitted to Slave axis (Slave) or the back part synchronous control FB because the stop factor occurred on Slave axis (Slave), the synchronous position relation deviates. Before enabling the command filter again, execute the synchronous positioning as needed.
- When using the mixed operation cycles function, there are some restrictions if the control operation cycle setting of the master axis and slave axis are not the same. For details, refer to each function and the following.

User's manual for the controller used

Program example

The following is a program example. The filter processing that restricts the velocity of the synchronous encoder is performed.

- Operation



- Settings

Item		Setting value
Positive direction/negative direction limit value	Positive direction limit value	100.0
	Negative direction limit value	0.0
Backlash amount		1.0

■Axes

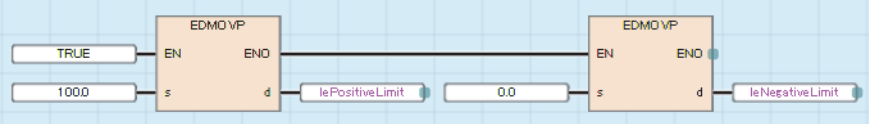
Axis No.	Label name	Data type	Comment
1	EncoderAxis0001	AXIS_REF	Axis 1
2	Axis0002	AXIS_REF	Axis 2
3	Axis0003	AXIS_REF	Axis 3

■Labels used

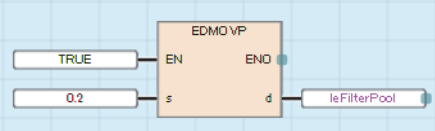
Label name	Data type	Comment
MCv_SpeedLimitFilter_1	MCv_SpeedLimitFilter	Speed Limit Filter FB
bSpeedLimitFilter	Bit	Speed limit filter command
lePositiveLimit	Double-precision real number	Positive direction limit value
leNegativeLimit	Double-precision real number	Negative direction limit value
bSpeedLimitValid	Bit	Output value valid
bSpeedLimitBusy	Bit	Executing
bSpeedLimitActive	Bit	Controlling
bSpeedLimitCommandAborted	Bit	Abortion of execution
bSpeedLimitError	Bit	Error
uwSpeedLimitErrorID	Word [unsigned]/bit string [16-bit]	Error code
leFilterPool	Double-precision real number	Filter cumulated value
MCv_SmoothingFilter_1	MCv_SmoothingFilter	Smoothing Filter FB
bSmoothingFilter	Bit	Smoothing filter command
leFrequency	Double-precision real number	Frequency
bSmoothingValid	Bit	Output value valid
bSmoothingInVelocity	Bit	Target velocity reached
bSmoothingBusy	Bit	Executing
bSmoothingActive	Bit	Controlling
bSmoothingCommandAborted	Bit	Abortion of execution
bSmoothingError	Bit	Error
uwSmoothingErrorID	Word [unsigned]/bit string [16-bit]	Error code
leSmoothingValidFrequency	Double-precision real number	Valid frequency
leSmoothingFilterPool	Double-precision real number	Filter cumulated value

■FBD/LD program

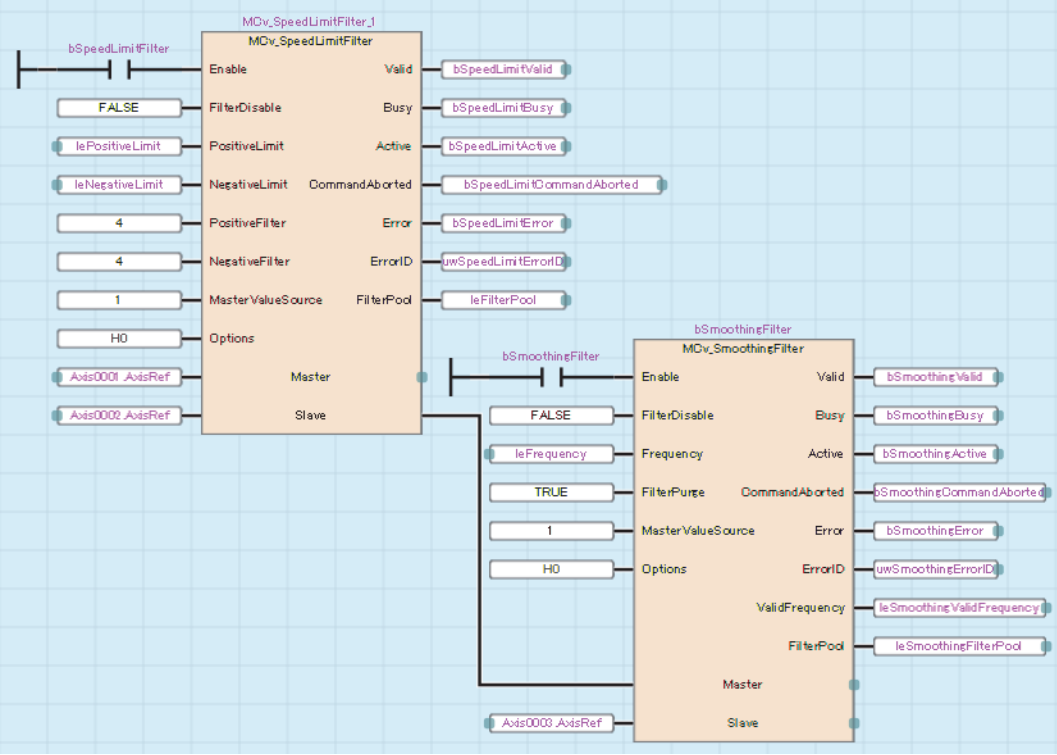
• Positive direction/negative direction limit value



• Frequency



• Speed Limit Filter/Smoothing Filter



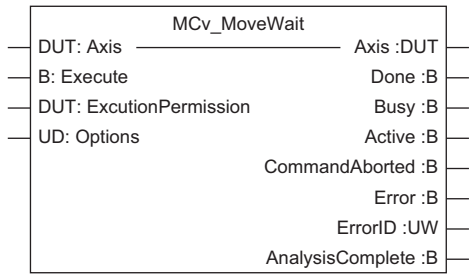
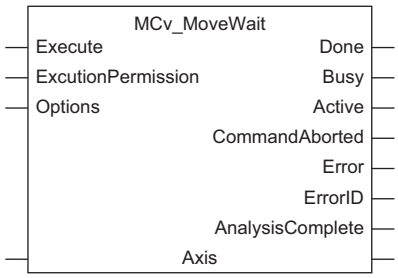
■ST program

```
//-----Positive direction/negative direction limit value-----
lePositiveLimit:= 100.0;
leNegativeLimit:= 0.0;

//-----Speed Limit Filter-----
MCv_SpeedLimitFilter_1(
    Master:= EncoderAxis0001.AxisRef ,
    Slave:= Axis0002.AxisRef ,
    Enable:= bSpeedLimitFilter ,
    FilterDisable:= FALSE ,
    PositiveLimit:= lePositiveLimit ,
    NegativeLimit:= leNegativeLimit ,
    PositiveFilter:= MC_VELOCITY_LIMIT_MODE__ClampWithoutRamp ,
    NegativeFilter:= MC_VELOCITY_LIMIT_MODE__ClampWithoutRamp ,
    MasterValueSource:= MC_SOURCE__mcSetValue ,
    Options:= H00000000 ,
    Valid=> bSpeedLimitValid ,
    Busy=> bSpeedLimitBusy ,
    Active=> bSpeedLimitActive ,
    CommandAborted=> bSpeedLimitCommandAborted ,
    Error=> bSpeedLimitError ,
    ErrorID=> uwSpeedLimitErrorID ,
    FilterPool=> leFilterPool
);

//-----Frequency-----
leFrequency:= 0.20;

//-----Smoothing Filter-----
MCv_SmoothingFilter_1(
    Master:= MCv_SpeedControl_1.Slave ,
    Slave:= Axis0003.AxisRef ,
    Enable:= bSmoothingFilter ,
    FilterDisable:= FALSE ,
    Frequency:= leFrequency ,
    FilterPurge:= TRUE ,
    MasterValueSource:= MC_SOURCE__mcSetValue ,
    Options:= H00000000 ,
    Valid=> bSmoothingValid ,
    Busy=> bSmoothingBusy ,
    Active=> bSmoothingActive ,
    CommandAborted=> bSmoothingCommandAborted ,
    Error=> bSmoothingError ,
    ErrorID=> uwSmoothingErrorID ,
    ValidFrequency=> leSmoothingValidFrequency ,
    FilterPool=> leSmoothingFilterPool
);
```

42.22 Axis Control Start Wait

MCv_MoveWait

This FB immediately performs the operation of the multiple-started motion FB for the specified axis by inputting an execution permission signal.

Ladder	FBD/LD	ST
		MCv_MoveWait(Axis:= ?AXIS_REF? , Execute:= ?BOOL? , ExecutionPermission:= ?MC_INPUT_REF? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD? , AnalysisComplete=> ?BOOL?);

Setting data

■Input/output variable

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is TRUE, MCv_MoveWait (Axis control start wait) is executed.
ExecutionPermission	Execution permission signal	MC_INPUT_REF	At start	—	Omission not possible	A signal used for clearing axis control start wait is set.
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	Set "00000000H". An error occurs if a value other than "00000000H" is set.

■Output variable

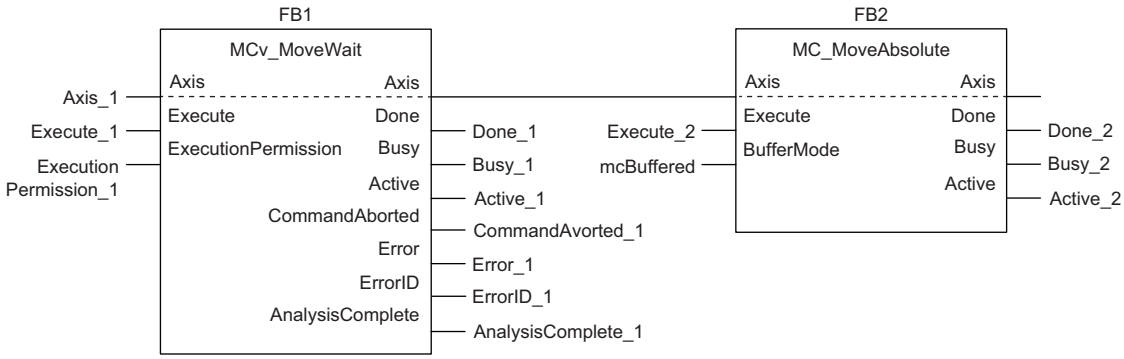
Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when input detection of the signal set by ExecutionPermission (Execution permission signal) and AnalysisComplete (Analysis completion) become TRUE, indicating MCv_MoveWait (Axis control start wait) control is completed.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_MoveWait (Axis control start wait) is being executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MCv_MoveWait (Axis control start wait) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution is aborted by another FB.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used
AnalysisComplete	Analysis completion	BOOL	FALSE	This variable becomes TRUE when the multiple-started motion FB analysis is complete.

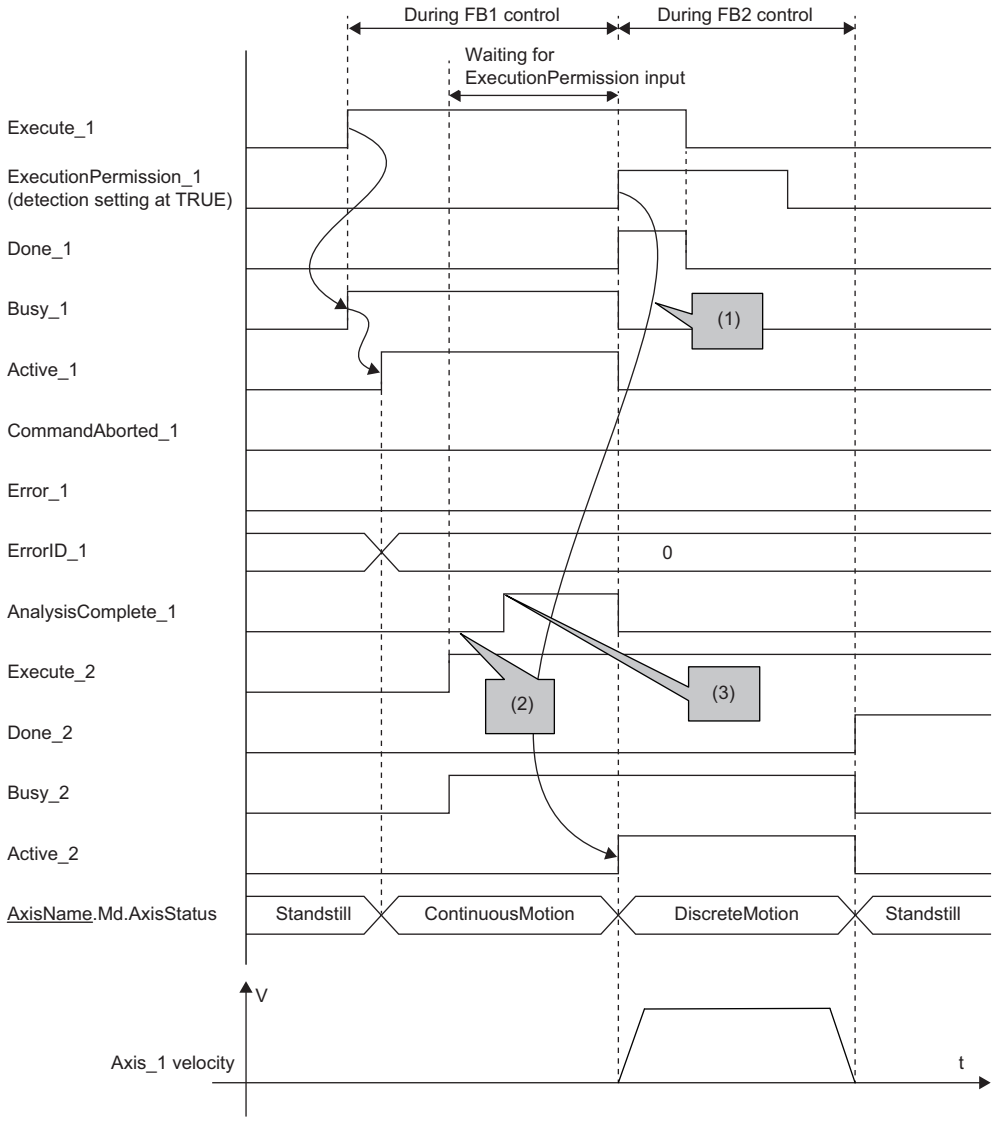
Processing details

- The axis operation of the multiple-started single axis control motion FB is immediately started when ExecutionPermission (Execution permission signal) is input.
- If another motion FB performs multiple start while MCv_MoveWait (Axis control start wait) is in operation, the multiple-started FB is executed until analysis is complete and waits until ExecutionPermission (Execution permission signal) is input. The timing of completion of multiple-started FB analysis can be checked with AnalysisComplete (Analysis completion).
- Execute ExecutionPermission (Execution permission signal) after AnalysisComplete (Analysis completion) becomes TRUE.
- Axis status (AxisName.Md.AxisStatus) is "6: During continuous operation (ContinuousMotion)" until ExecutionPermission (Execution permission signal) is input.
- When ExecutionPermission (Execution permission signal) is input, the multiple-started FB is executed after one operation cycle from when the input signal is detected. At this time, Axis Status (AxisName.Md.AxisStatus) transits to the state according to the multiple-started FB.
- Input ExecutionPermission (Execution permission signal) after AnalysisComplete (Analysis completion) becomes TRUE.

■Timing chart

• When the processing is completed successfully





(1): FB2 starts after one operation cycle from when the input of ExecutionPermission is detected.
(2): FB2 is being analyzed.
(3): Analysis of FB2 is completed.

• When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■ExecutionPermission (Execution permission signal)

Specify a signal used for clearing MCv_MoveWait (Axis control start wait).

ExecutionPermission (Execution permission signal) is set with an MC_INPUT_REF structure. For details on MC_INPUT_REF structures, refer to the following.

☞ Page 1442 MC_INPUT_REF

The following table shows the setting range for the input signal (ExecutionPermission.Signal) set in ExecutionPermission (Execution permission signal).

The type of input signal (ExecutionPermission.Signal) is SIGNAL_SELECT structure.

Structure	Variable name	Type	Setting range
SIGNAL_SELECT (Signal select)	Source.Target (Signal)*1	TARGET_REF ☞ Page 1444 TARGET_REF (Input signal)	■Type • BOOL ■Data category • [OBJ] • [VAR] • [DEV] • [CONST]
	Detection (Signal detection method)	INT (MC_SIGNAL_LOGIC) ☞ Page 1478 MC_SIGNAL_LOGIC	• 0: Detection at TRUE (HighLevel) • 1: Detection at FALSE (LowLevel)
	CompensationTime (Compensation time)*2	LREAL	-5.0 to 0.0 [s]
	FilterTime (Filter time)	LREAL	0.0 to 5.0 [s]

*1 An external signal high-accuracy input cannot be used. If a signal for an external signal high-accuracy input is specified, operation is performed with the same accuracy as a normal signal.

*2 If a value smaller than one operation cycle is specified for Compensation time (ExecutionPermission.Signal.CompensationTime), operation is performed as 0.0 (no compensation).

When [OBJ] is specified, if the target modification is omitted, the object set in the target axis specified by the operand axis (Axis) is referred. However, if the axis set to Axis (Axis) does not have a station address or a station address has not been set, External signal selection incorrect (error code: 1B31H) occurs.

When the station referred to when [OBJ] is specified has the Axis Emulate function enabled and also has a set station address, the emulating object will be referred.

To correct the advancement of ExecutionPermission (Execution permission signal) or adjust the execution timing of the multiple-started motion FB, set Compensation time (ExecutionPermission.Signal.CompensationTime).

To prevent signal misdetection caused by the chattering of ExecutionPermission (Execution permission signal), set Filter time (ExecutionPermission.Signal.FilterTime).

When a signal that cannot be used for Signal name (ExecutionPermission.Signal.Source.Target) is used, External signal selection string incorrect (error code: 1B05H), External signal selection incorrect (error code: 1B31H or 1B32H), or Add-on Signal IO internal error (error code: 1B33H) is output.

Precautions

- Input ExecutionPermission (Execution permission signal) after AnalysisComplete (Analysis completion) becomes TRUE. If no multiple-started motion FB exists or if analysis is not completed (when AnalysisComplete (Analysis completion) is FALSE), even if the input of ExecutionPermission (Execution permission signal) is detected, Done (Execution completion) does not become TRUE.
- Perform multiple start with the motion FB after Active (Executing) of MCv_MoveWait (Axis control start wait) becomes TRUE. If an error occurs while MCv_MoveWait (Axis control start wait) is being executed, a multiple-started motion FB may be executed without waiting for the input of ExecutionPermission (Execution permission signal).
- When using the mixed operation cycles function, the operation cycle of the MCv_MoveWait (Axis control start wait) target axis and the refresh cycle of ExecutionPermission (Execution permission signal) must be the same. If the refresh cycle is different, the positioning start timing may become irregular.
- To start axes with different operation cycles at the same time, set the same operation cycle to each axis. Otherwise, the axis start timing may deviate by one operation cycle. To start axes with different operation cycles at the same time, use multiple signals of ExecutionPermission (Execution permission signal) with the same Signal name (ExecutionPermission.Signal.Source.Target) and the same Signal logic (ExecutionPermission.Signal.Detection) set, and adjust the Compensation time (ExecutionPermission.Signal.CompensationTime).
- When executing MCv_MoveWait (axis control start wait), check that Axis Status (AxisName.Md.AxisStatus) is "4: Standby (Standstill)". If Axis status (AxisName.Md.AxisStatus) is unable to start, Start Not Possible (error code: 1AFCH) occurs, and control in operation stops.
- If MCv_MoveWait (axis control start wait) is started while another operation system FB is running, the start request will be ignored and Start during Operation Warning (event code: 00D01H) will occur.
- Set Buffered to Buffer mode (BufferMode) for the multiple start motion FB. If multiple start is performed by setting Aborting, MCv_MoveWait (Axis control start wait) is aborted, the input status of ExecutionPermission (Execution permission signal) is ignored, and the multiple-started motion FB is executed immediately. If multiple start is performed when Blending is set, BufferMode specification not allowed (Error code: 1AE5H) occurs.

Program example

The following is a program example. By setting Control start (bExecute) to TRUE, MC_MoveRelative and MCv_MoveLinearInterpolateRelative are analyzed and put in standby, and by setting Execution permission signal (bExecutionPermission) to TRUE, the FBs that are in standby mode are started at the same time.

■Axis

Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REAL	Axis 1
2	Axis0002	AXIS_REAL	Axis 2
3	Axis0003	AXIS_REAL	Axis 3

■Axes group

Axes group No.	Label name	Data type	Comment
1	AxisGroup0001	AXIS_GROUP	Axes group 1

Item	Setting value
Structuring axis[1]	Axis0001
Structuring axis[2]	Axis0002

■I/O data

Label name	Data category	Data type	Source type	Source	Signal detection method	Compensation time	Filter time
SignalData0001	Input signal	MC_INPUT_REF	Label	bExecutionPermission	HighLevel	0.0	0.0

■Labels used

Label name	Data type	Comment
bExecute	Bit	Control start
bExecutionPermission	Bit	Execution permission signal
MCv_MoveWait_1	MCv_MoveWait	Axis control start wait FB
MC_MoveRelative_1	MC_MoveRelative	Relative Value Positioning FB
MCv_GroupMoveWait_1	MCv_GroupMoveWait	Axes group control start wait FB
MCv_MoveLinearInterpolateRelative_1	MCv_MoveLinearInterpolateRelative	Relative Value Linear Interpolation Control FB
wMoveLinearInterpolateRelativeLinearAxes	Word [signed](0..15)	Linear interpolation axis
leMoveLinearInterpolateRelativePosition	Double-precision real number(0..15)	Target position

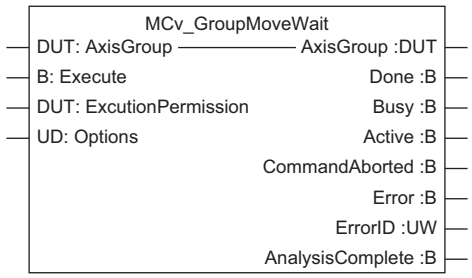
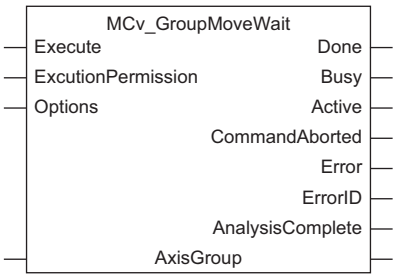
■ST program

```
SET( MCv_MoveWait_1.Active , bMoveRelativeExecute );
SET( MCv_GroupMoveWait_1.Active , bMoveLinearInterpolateRelativeExecute );
RST( NOT bExecute , bMoveRelativeExecute );
RST( NOT bExecute , bMoveLinearInterpolateRelativeExecute );
//-----Axis control start wait-----
MCv_MoveWait_1(
   Axis:= Axis0001.AxisRef ,
   Execute:= bExecute ,
   ExecutionPermission:= SignalData0001 ,
   Options:= H00000000
);
//-----Relative value positioning-----
MC_MoveRelative_1(
   Axis:= Axis0001.AxisRef ,
   Execute:= bMoveRelativeExecute ,
   ContinuousUpdate:= FALSE ,
   Distance:= 10000.0 ,
   Velocity:= 1000.0 ,
   Acceleration:= 1000.0 ,
   Deceleration:= 1000.0 ,
   Jerk:= 0.0 ,
   BufferMode:= MC_BUFFER_MODE__mcBuffered ,
   Options:= H00000000
);
//-----Axes group control start wait-----
MCv_GroupMoveWait_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bExecute ,
   ExecutionPermission:= SignalData0001 ,
   Options:= H00000000
);
//-----Data setting for relative value linear interpolation control-----
wMoveLinearInterpolateRelativeLinearAxes[0] := 1;
wMoveLinearInterpolateRelativeLinearAxes[1] := 2;
leMoveLinearInterpolateRelativeDistance[0] := 10000.0;
leMoveLinearInterpolateRelativeDistance[1] := 10000.0;
//-----Relative value linear interpolation control-----
MCv_MoveLinearInterpolateRelative_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bMoveLinearInterpolateRelativeExecute ,
   ContinuousUpdate:= FALSE ,
   LinearAxes:= wMoveLinearInterpolateRelativeLinearAxes ,
   Distance:= leMoveLinearInterpolateRelativeDistance ,
   Velocity:= 1000.0 ,
   Acceleration:= 1000.0 ,
   Deceleration:= 1000.0 ,
   Jerk:= 0.0 ,
   VelocityMode:= MC_INTERPOLATE_SPEED_MODE__VectorSpeed ,
   BufferMode:= MC_BUFFER_MODE__mcBuffered ,
   Options:= H00000000
);
```

42.23 Axes Group Control Start Wait

MCv_GroupMoveWait

This FB immediately performs the operation of the multiple-started motion FB for the specified axes group by inputting an execution permission signal.

Ladder	FBD/LD	ST
		MCv_GroupMoveWait(AxesGroup:= ?AXES_GROUP_REF? , Execute:= ?BOOL? , ExecutionPermission:= ?MC_INPUT_REF? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD? , AnalysisComplete=> ?BOOL?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
AxesGroup	Axes group information	AXES_GROUP_REF	At start	—	Omission not possible	An axes group is set. For the variable (AxesGroupName.AxesGroupRef.) used, refer to the following. ☞Page 1423 AxesGroupName.AxesGroupRef. (Axes group information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is TRUE, MCv_GroupMoveWait (Axes group control start wait) is executed.
ExecutionPermission	Execution permission signal	MC_INPUT_REF	At start	—	Omission not possible	A signal used for clearing axes group control start wait is set.
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	Set "00000000H". An error occurs if a value other than "00000000H" is set.

■Output variable

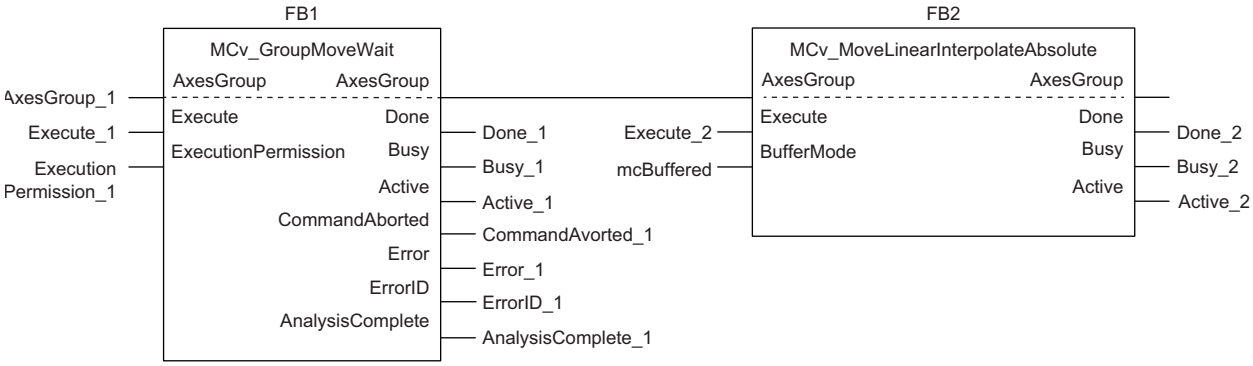
Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the input of the signal set by ExecutionPermission (Execution permission signal) and AnalysisComplete (Analysis completion) become TRUE, indicating MCv_GroupMoveWait (Axes group control start wait) control is completed.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_GroupMoveWait (Axes group control start wait) is being executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MCv_GroupMoveWait (Axes group control start wait) is controlling the axes.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution is aborted by another FB.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used
AnalysisComplete	Analysis completion	BOOL	FALSE	This variable becomes TRUE when the multiple-started motion FB analysis is complete.

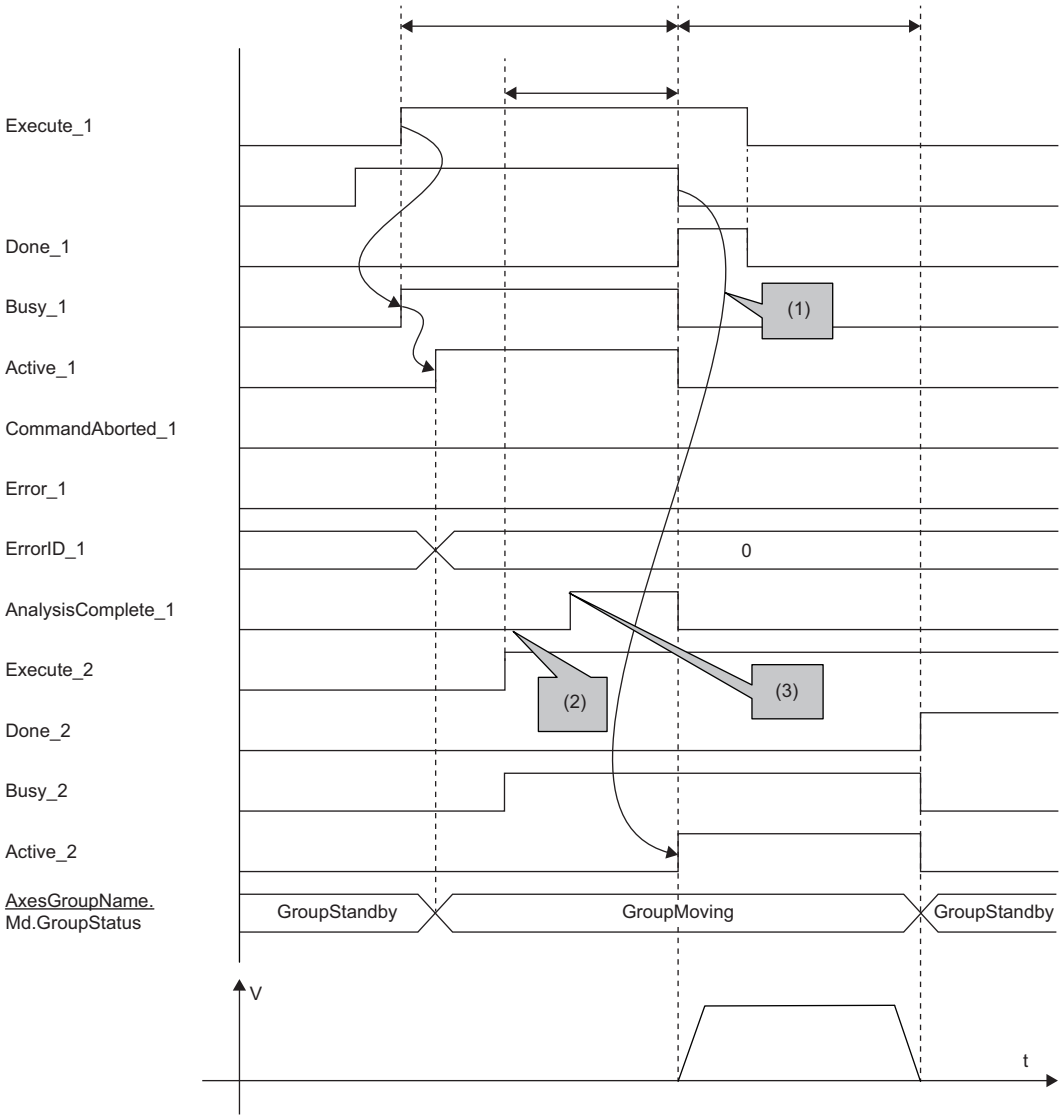
Processing details

- The axes group operation of a multiple-started multi-axis control motion FB is immediately executed when ExecutionPermission (Execution permission signal) is input.
- If another motion FB performs multiple start while MCv_GroupMoveWait (Axes group control start wait) is in operation, the multiple-started FB is executed until analysis is complete and waits until ExecutionPermission (Execution permission signal) is input. The timing of completion of multiple-started FB analysis can be checked with AnalysisComplete (Analysis completion).
- Execute ExecutionPermission (Execution permission signal) after AnalysisComplete (Analysis completion) becomes TRUE.
- Axes group status (AxisGroupName.Md.GroupStatus) is "5: Operating (GroupMoving)" until ExecutionPermission (Execution permission signal) is input.
- When ExecutionPermission (Execution permission signal) is input, the multiple-started FB is executed after one operation cycle from when the input signal is detected.
- Input ExecutionPermission (Execution permission signal) after AnalysisComplete (Analysis completion) becomes TRUE.

■Timing chart

• When the processing is completed successfully





(1): FB2 starts after one operation cycle from when the input of ExecutionPermission is detected.
(2): FB2 is being analyzed.
(3): Analysis of FB2 is completed.

• When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■ExecutionPermission (Execution permission signal)

Specify a signal used for clearing MCv_GroupMoveWait (Axes group control start wait).

ExecutionPermission (Execution permission signal) is set with an MC_INPUT_REF structure. For details on MC_INPUT_REF structures, refer to the following.

☞ Page 1442 MC_INPUT_REF

The following table shows the setting range for the input signal (ExecutionPermission.Signal) set in ExecutionPermission (Execution permission signal).

The type of input signal (ExecutionPermission.Signal) is SIGNAL_SELECT structure.

Structure	Variable name	Type	Setting range
SIGNAL_SELECT (Signal select)	Source.Target (Signal)*1	TARGET_REF ☞ Page 1444 TARGET_REF (Input signal)	■Type • BOOL ■Data category • [OBJ] • [VAR] • [DEV] • [CONST]
	Detection (Signal detection method)	INT (MC_SIGNAL_LOGIC) ☞ Page 1478 MC_SIGNAL_LOGIC	• 0: Detection at TRUE (HighLevel) • 1: Detection at FALSE (LowLevel)
	CompensationTime (Compensation time)*2	LREAL	-5.0 to 0.0 [s]
	FilterTime (Filter time)	LREAL	0.0 to 5.0 [s]

*1 An external signal high-accuracy input cannot be used. If a signal for an external signal high-accuracy input is specified, operation is performed with the same accuracy as a normal signal.

*2 If a value smaller than one operation cycle is specified for Compensation time (ExecutionPermission.Signal.CompensationTime), operation is performed as 0.0 (no compensation).

When [OBJ] is specified, if the target modification is omitted, External signal selection incorrect (error code: 1B31H) is output.

When the station referred to when [OBJ] is specified has the Axis Emulate function enabled and also has a set station address, the emulating object will be referred.

To correct the advancement of ExecutionPermission (Execution permission signal) and adjust the execution timing of the multiple-started motion FB, set Compensation time (ExecutionPermission.Signal.CompensationTime).

To prevent signal misdetection caused by the chattering of ExecutionPermission (Execution permission signal), set Filter time (ExecutionPermission.Signal.FilterTime).

When a signal that cannot be used for Signal name (ExecutionPermission.Signal.Source.Target) is used, External signal selection string incorrect (error code: 1B05H), External signal selection incorrect (error code: 1B31H or 1B32H), or Add-on Signal IO internal error (error code: 1B33H) is output.

Precautions

- Input ExecutionPermission (Execution permission signal) after AnalysisComplete (Analysis completion) becomes TRUE. If no multiple-started motion FB exists or if analysis is not completed (when AnalysisComplete (Analysis completion) is FALSE), even if the input of ExecutionPermission (Execution permission signal) is detected, Done (Execution completion) does not become TRUE.
- Perform multiple start with the motion FB after Active (Executing) of MCv_GroupMoveWait (Axes group control start wait) becomes TRUE. If an error occurs while MCv_GroupMoveWait (Axes group control start wait) is being executed, a multiple-started motion FB may be executed without waiting for the input of ExecutionPermission (Execution permission signal).
- When using the mixed operation cycles function, the operation cycle of the MCv_GroupMoveWait (Axes group control start wait) target axis and the refresh cycle of ExecutionPermission (Execution permission signal) must be the same. If the refresh cycle is different, the positioning start timing may become irregular.
- To start axes with different operation cycles at the same time, set the same operation cycle to each axis. Otherwise, the axis start timing may deviate by one operation cycle. To start axes with different operation cycles at the same time, use multiple signals of ExecutionPermission (Execution permission signal) with the same Signal name (ExecutionPermission.Signal.Source.Target) and the same Signal logic (ExecutionPermission.Signal.Detection) set, and adjust the Compensation time (ExecutionPermission.Signal.CompensationTime).
- When executing MCv_GroupMoveWait (Axes group control start wait), check that Axes group status (AxisGroupName.Md.GroupStatus) is "4: Standby (GroupStandby)". If Axes group status (AxisGroupName.Md.GroupStatus) is unable to start, Start Not Possible (error code: 1AFCH) occurs, and control in operation stops.
- If MCv_GroupMoveWait (axes group control start wait) is started while another operation system FB is running, the start request will be ignored and Start during Operation Warning (event code: 00D01H) will occur.
- Set Buffered to Buffer mode (BufferMode) for the multiple start motion FB. If multiple start is performed by setting Aborting, MCv_GroupMoveWait (Axes group control start wait) is aborted, the input status of ExecutionPermission (Execution permission signal) is ignored, and the multiple-started motion FB is executed immediately. If multiple start is performed when Blending is set, BufferMode specification not allowed (Error code: 1AE5H) occurs.

Program example

The following is a program example. By setting Control start (bExecute) to TRUE, MC_MoveRelative and MCv_MoveLinearInterpolateRelative are analyzed and put in standby, and by setting Execution permission signal (bExecutionPermission) to TRUE, the FBs that are in standby mode are started at the same time.

■Axis

Axis No.	Label name	Data type	Comment
1	Axis0001	AXIS_REAL	Axis 1
2	Axis0002	AXIS_REAL	Axis 2
3	Axis0003	AXIS_REAL	Axis 3

■Axes group

Axes group No.	Label name	Data type	Comment
1	AxisGroup0001	AXIS_GROUP	Axes group 1

Item	Setting value
Structuring axis[1]	Axis0001
Structuring axis[2]	Axis0002

■I/O data

Label name	Data category	Data type	Source type	Source	Signal detection method	Compensation time	Filter time
SignalData0001	Input signal	MC_INPUT_REF	Label	bExecutionPermission	HighLevel	0.0	0.0

■Labels used

Label name	Data type	Comment
bExecute	Bit	Control start
bExecutionPermission	Bit	Execution permission signal
MCv_MoveWait_1	MCv_MoveWait	Axis control start wait FB
MC_MoveRelative_1	MC_MoveRelative	Relative Value Positioning FB
MCv_GroupMoveWait_1	MCv_GroupMoveWait	Axes group control start wait FB
MCv_MoveLinearInterpolateRelative_1	MCv_MoveLinearInterpolateRelative	Relative Value Linear Interpolation Control FB
wMoveLinearInterpolateRelativeLinearAxes	Word [signed](0..15)	Linear interpolation axis
leMoveLinearInterpolateRelativePosition	Double-precision real number(0..15)	Target position

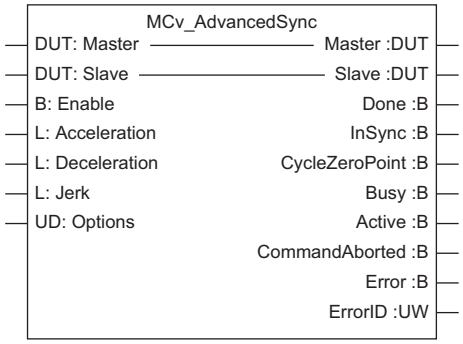
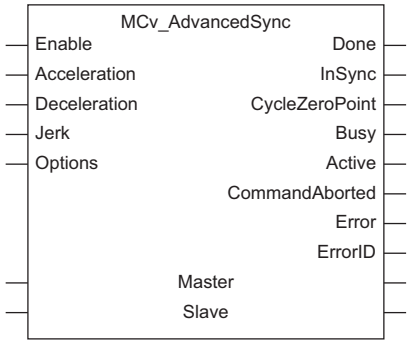
■ST program

```
SET( MCv_MoveWait_1.Active , bMoveRelativeExecute );
SET( MCv_GroupMoveWait_1.Active , bMoveLinearInterpolateRelativeExecute );
RST( NOT bExecute , bMoveRelativeExecute );
RST( NOT bExecute , bMoveLinearInterpolateRelativeExecute );
//-----Axis control start wait-----
MCv_MoveWait_1(
   Axis:= Axis0001.AxisRef ,
   Execute:= bExecute ,
   ExecutionPermission:= SignalData0001 ,
   Options:= H00000000
);
//-----Relative value positioning-----
MC_MoveRelative_1(
   Axis:= Axis0001.AxisRef ,
   Execute:= bMoveRelativeExecute ,
   ContinuousUpdate:= FALSE ,
   Distance:= 10000.0 ,
   Velocity:= 1000.0 ,
   Acceleration:= 1000.0 ,
   Deceleration:= 1000.0 ,
   Jerk:= 0.0 ,
   BufferMode:= MC_BUFFER_MODE__mcBuffered ,
   Options:= H00000000
);
//-----Axes group control start wait-----
MCv_GroupMoveWait_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bExecute ,
   ExecutionPermission:= SignalData0001 ,
   Options:= H00000000
);
//-----Data setting for relative value linear interpolation control-----
wMoveLinearInterpolateRelativeLinearAxes[0] := 1;
wMoveLinearInterpolateRelativeLinearAxes[1] := 2;
leMoveLinearInterpolateRelativeDistance[0] := 10000.0;
leMoveLinearInterpolateRelativeDistance[1] := 10000.0;
//-----Relative value linear interpolation control-----
MCv_MoveLinearInterpolateRelative_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bMoveLinearInterpolateRelativeExecute ,
   ContinuousUpdate:= FALSE ,
   LinearAxes:= wMoveLinearInterpolateRelativeLinearAxes ,
   Distance:= leMoveLinearInterpolateRelativeDistance ,
   Velocity:= 1000.0 ,
   Acceleration:= 1000.0 ,
   Deceleration:= 1000.0 ,
   Jerk:= 0.0 ,
   VelocityMode:= MC_INTERPOLATE_SPEED_MODE__VectorSpeed ,
   BufferMode:= MC_BUFFER_MODE__mcBuffered ,
   Options:= H00000000
);
```

42.24 Advanced Synchronous Control

MCv_AdvancedSync

This FB starts synchronous control according to the specified advanced synchronous control settings.

Ladder	FBD/LD	ST
		MCv_AdvancedSync(Master:= ?AXIS_REF? , Slave:= ?AXIS_REF? , Enable:= ?BOOL? , Acceleration:= ?LREAL? , Deceleration:= ?LREAL? , Jerk:= ?LREAL? , Options:= ?DWORD? , Done=> ?BOOL? , InSync=> ?BOOL? , CycleZeroPoint=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Master	Master axis	AXIS_REF	At start	—	Omission not possible	The input axis is set. If the value is omitted, Out of master axis No. range (error code: 1B0BH) occurs. If no corresponding input setting exists, Advanced synchronization control axis unset (error code: 1B22H) occurs at start.
Slave	Slave axis	AXIS_REF	At start	—	Omission not possible	The output axis is set. If the value is omitted, Out of slave axis No. range (error code: 1B0CH) occurs. If no corresponding output setting exists, Advanced synchronization control axis unset (error code: 1B22H) occurs at start.

■Input variables

Input variable	Name	Data type	Input	Valid range	Default value	Description
Enable	Enable	BOOL	Always	TRUE, FALSE	FALSE	This FB is executed when the value is set to TRUE.
Acceleration	Acceleration	LREAL	At start	0.0	0.0	Set "0.0". If a value other than "0.0" is set, Out of acceleration range (error code: 1A8DH) occurs.
Deceleration	Deceleration	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration is set. For details, refer to the following. ☞ Page 1866 Deceleration (Deceleration)
Jerk	Jerk	LREAL	At start	0.0	0.0	Set "0.0". If a value other than "0.0" is set, Out of jerk range (error code: 1A8FH) occurs.
Options	Options	DWORD(HEX)	At start	Refer to the following. ☞ Page 1866 Options (Options)	00000000H	The function options for this FB are set in the specified bit position. For details, refer to the following. ☞ Page 1866 Options (Options)

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when any of the following conditions are met. • After the execution status is transitioned from this FB to the next motion control FB in which multiple start (buffer mode) is set to "1: Buffered" • When the analysis in synchronous control analysis mode is completed
InSync	In synchronization	BOOL	FALSE	This variable becomes TRUE when the output axis starts synchronization.
CycleZeroPoint	Cam cycle completion	BOOL	FALSE	This variable becomes TRUE whenever the current position per cycle exceeds the permissible range, when the target is a linear cam, or when a cam whose cam start point is "0" is used after Controlling (Active) becomes TRUE. This variable becomes TRUE when the current position per cycle passes the cam data 0th-point position when a cam whose cam start point is other than "0" is used.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when this FB is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE when this FB is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of this FB is aborted. The variable becomes FALSE if Enable (Enable) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

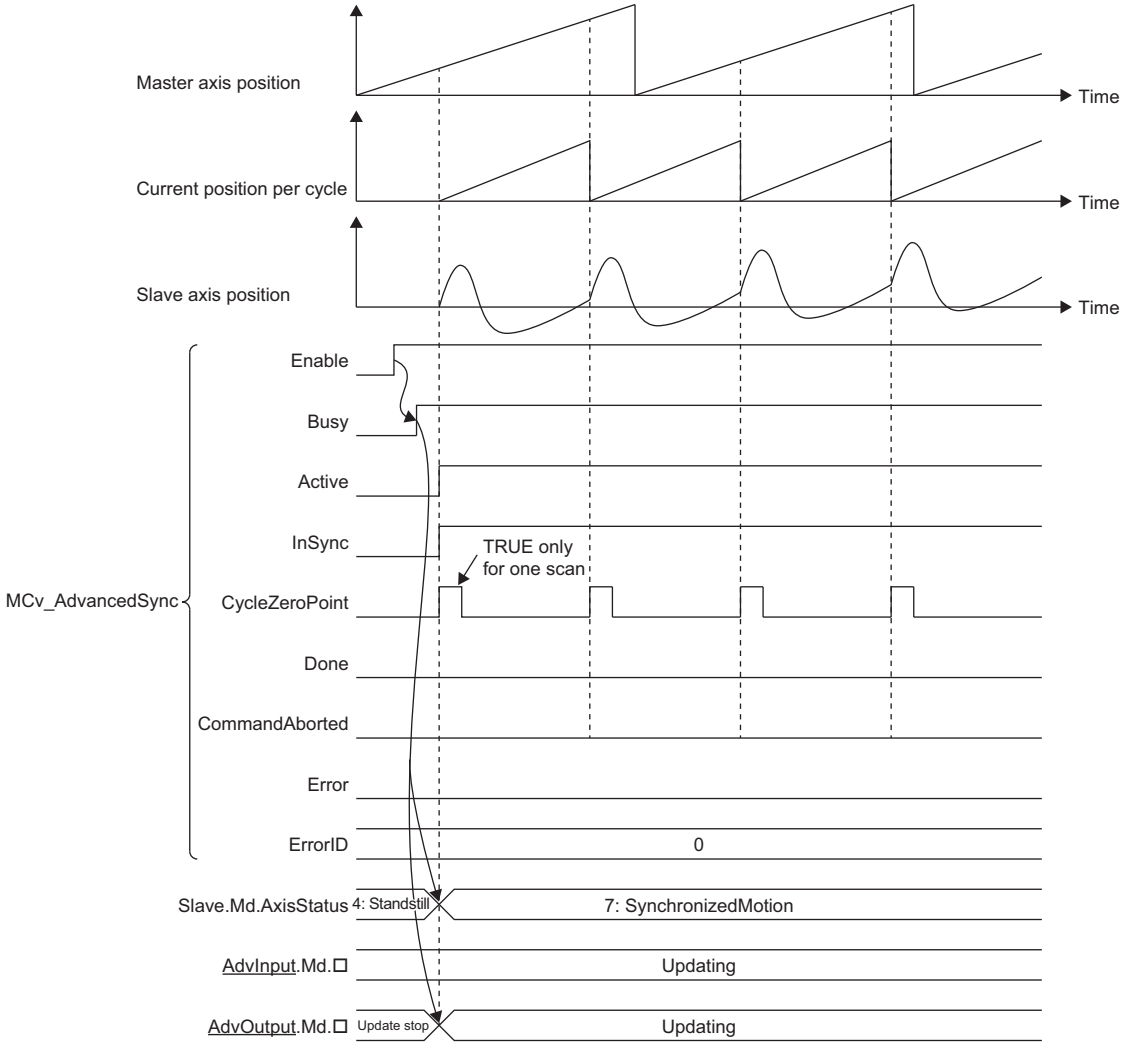
Processing details

- When this FB is executed, it starts advanced synchronous control according to the specified advanced synchronous control settings.
- Synchronous control can only be started when Axis status (AxisName.Md.AxisStatus) of the slave axis is "4: Standby (Standstill)". When control is started, Axis status (AxisName.Md.AxisStatus) of the slave axis changes to "7: During synchronous operation (SynchronizedMotion)". When synchronous control is started, if "4: Standby (Standstill)" is not set, Axis status (AxisName.Md.AxisStatus) of the slave axis becomes "1: Stopping on error (ErrorStop)" and Start not possible (error code: 1AFCH) occurs.
- If Enable (Enable) of this FB is set to FALSE, or a stop cause occurs in the axis set in the axis information on the output axis, Slave axis (Slave) immediately stops or is decelerated to stop according to the stop cause and its stop processing.
- For details on advanced synchronous control, refer to the following.

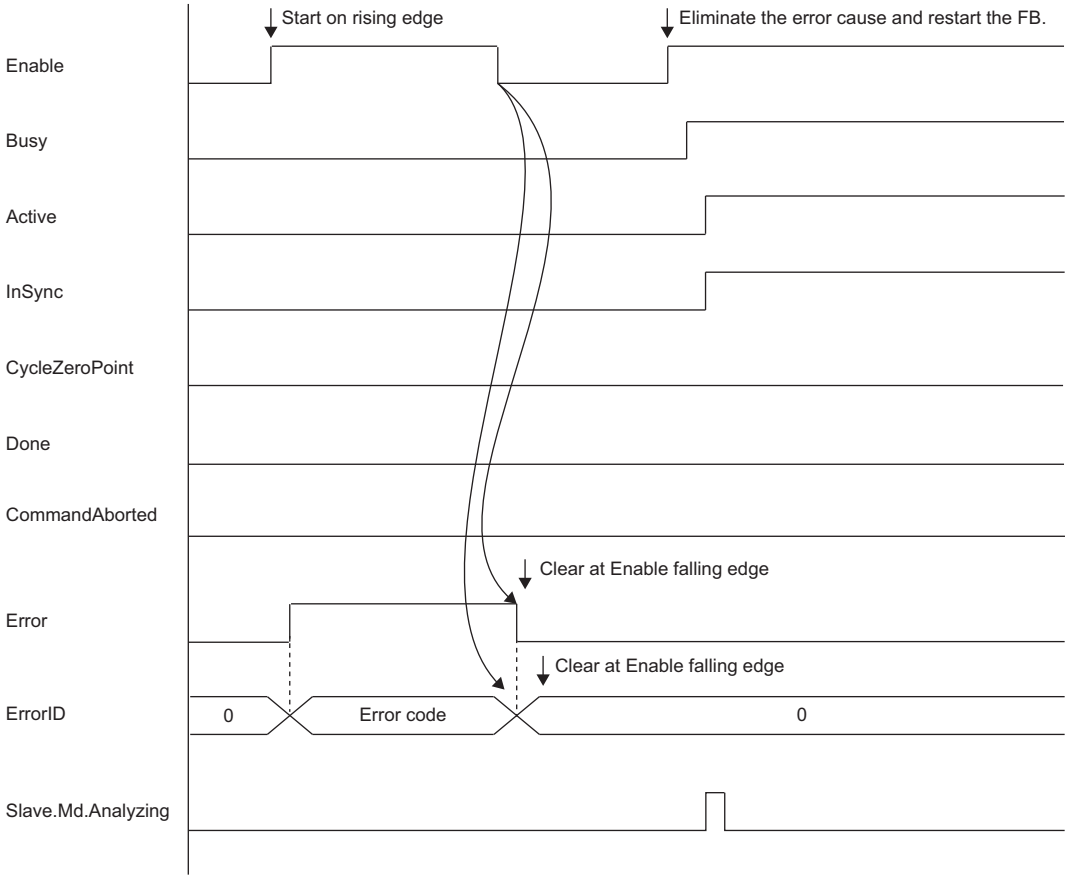
📖 User's manual for the controller used

■Timing chart

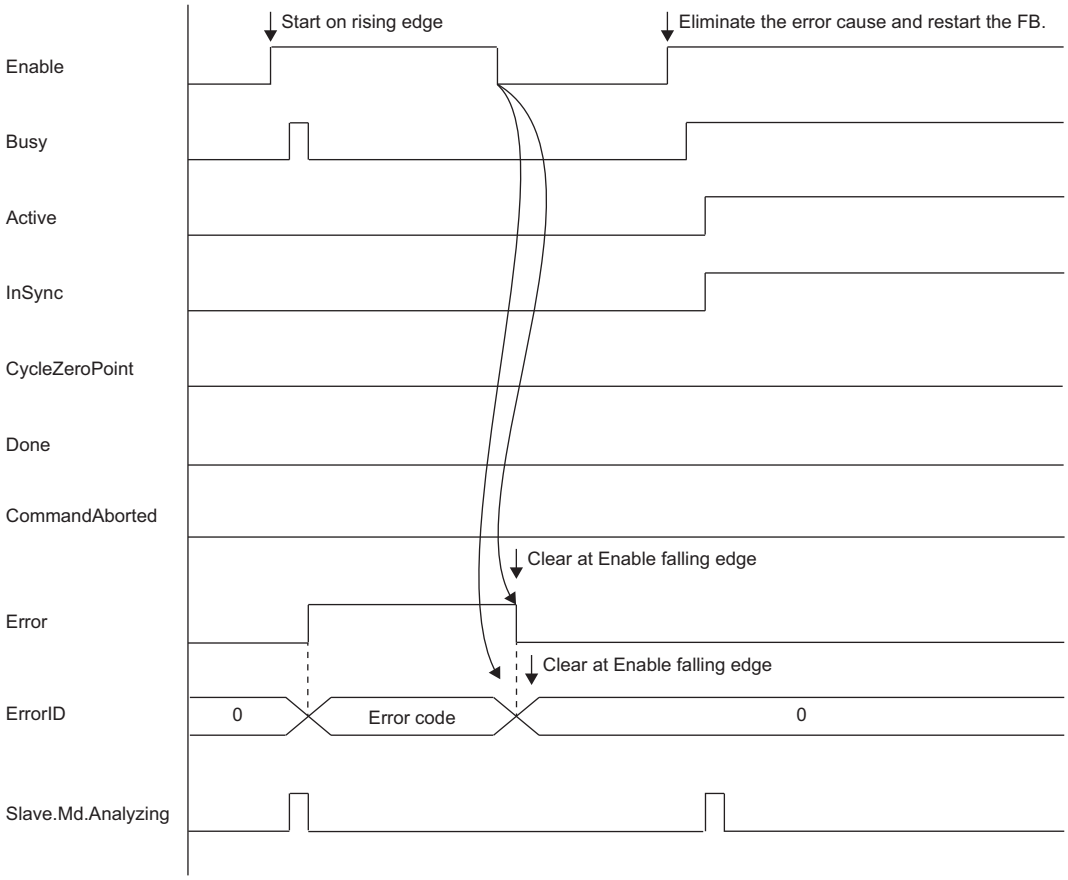
• When the processing is completed successfully



• When I/O variables are abnormal



• When an input variable is abnormal



■Deceleration (Deceleration)

The deceleration is set.

Specify the setting range if Acceleration/deceleration method setting (Options (Options): Bit 0 to 1) is set to "0: Acceleration/deceleration specification method (mcAccDec)".

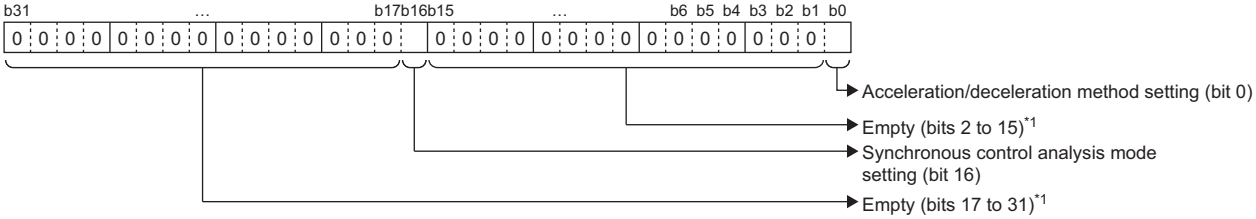
Acceleration/deceleration method setting (Options (Options): Bit 0 to 1)	Setting range
0: Acceleration/deceleration specification method (mcAccDec)	0.0000[*1], positive numbers from 0.0001 to 2147483647.0 [U/s^2]

*1 The operation performed at start varies depending on the setting of Operation selection at start acceleration/deceleration 0 (AxisName.Pr.AccelerationZeroBehavior). When changing the acceleration or deceleration, the changes are not accepted.

■Options (Options)

The function options for this FB are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to "0".

Bit	Name	Description
0	Acceleration/deceleration method setting	The acceleration/deceleration method used to perform control is set. • 0: Acceleration/deceleration specification method (mcAccDec)
16	Synchronous control analysis mode setting	The synchronous control analysis mode is set. • 0: Invalid • 1: Valid

• Acceleration/deceleration method setting (bit 0 to 1)

Setting value	Description
0: Acceleration/deceleration specification method (mcAccDec)	An acceleration/deceleration method using Deceleration (Deceleration) set in this FB

Precautions

- Even if a stop cause occurs in an axis set to the input axis information of Master axis (Master), Auxiliary axis (AdvOutputName.Pr.AuxAxis), or Sub-input axis (AdvOutputName.Pr.SubAxis) while this FB is being executed, Slave axis (Slave) maintains control under the command transmitted from the input axis. The axis status of Slave axis (Slave) does not change. Also, even if a stop cause occurs in an axis set to Axis information (AdvOutputName.Axis) of the output axis, the axis set to Axis information (AdvInputName.Axis) of the input axis is not affected.
- When this FB is started, Master axis and slave axis cyclic reference (error code: 1AADH) occurs in the following cases.

Error occurrence condition
When multiple axes are concatenated, Master axis (Master), Auxiliary axis (AdvOutputName.Pr.AuxAxis), and Sub-input axis (AdvOutputName.Pr.SubAxis) are set to Slave axis (Slave) of the single axis synchronous control FB in the later part.
257 or more single axis synchronous control FBs are concatenated.

- If Master axis (Master), Auxiliary axis (AdvOutputName.Pr.AuxAxis), or Sub-input axis (AdvOutputName.Pr.SubAxis) becomes serve OFF due to a servo alarm or emergency stop, Slave axis (Slave) may operate unexpectedly. Therefore, using MC_Stop (Forced Stop) to stop the operation of this FB is recommended.
- When the input axis is operated before Axis status (AxisName.Md.AxisStatus) of Slave axis (Slave) being analyzed for starting this FB becomes "7: During synchronous operation (SynchronizedMotion)", the information on the input axis being analyzed, such as the movement amount and the amount of change due to changes in the current value or other factors, will be reflected immediately after the start of synchronous control. The output axis may suddenly accelerate depending on the amount of movement or change in the input axis. Therefore, start the operation of the input axis after checking that synchronous control is active.
- If the setting values of the member variables ADV_INPUT and ADV_OUTPUT, which are imported during synchronous control, are changed to values outside the range during the execution of this FB, Out of Parameter Range (Advanced synchronous control input axis) or Out of Parameter Range (Advanced synchronous control output axis) occurs, and the control continues with the setting value before the change.
- When the setting values of the member variables ADV_INPUT and ADV_OUTPUT are outside the range at the start of this FB, Out of Parameter Range (Advanced synchronous control input axis) or Out of Parameter Range (Advanced synchronous control output axis) occurs, and the operation will not start.
- When this FB is started, Operation cycle mismatch (error code: 1B29H) occurs in the following cases. Even when MCV_SyncOperationCycles (Control operation cycle synchronization) is used, Operation cycle mismatch (error code: 1B29H) occurs if the following conditions are satisfied.

Error occurrence condition
The control operation cycles of Master axis (Master), Auxiliary axis (AdvOutputName.Pr.AuxAxis), and Sub-input axis (AdvOutputName.Pr.SubAxis) do not match that of the Slave axis (Slave).
The control operation cycle of the output axis does not match the refresh cycle of the clutch signals.

Program example

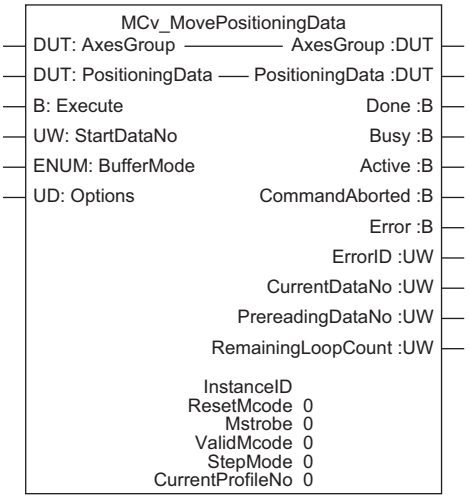
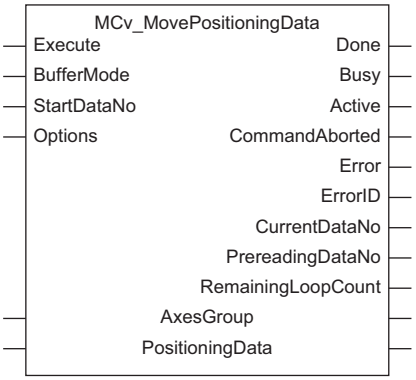
For a program example of this FB, refer to the following manual.

User's manual for the controller used

42.25 Multiple Axes Positioning Data Operation

MCv_MovePositioningData

This FB operates the axes group according to the specified positioning data.

Ladder	FBD/LD	ST
		MCv_MovePositioningData(AxesGroup:= ?AXES_GROUP_REF? , PositioningData:= ?MC_POSITIONING_DATA_ID? , Execute:= ?BOOL? , StartDataNo:= ?WORD? , BufferMode:= ?INT? , Options:= ?DWORD? , Done=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD? , CurrentDataNo=> ?WORD? , PrereadingDataNo=> ?WORD? , RemainingLoopCount=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
AxesGroup	Axes group information	AXES_GROUP_REF	At start	—	Omission not possible	An axes group is set.
PositioningData	Positioning data	MC_POSITIONING_DATA_ID	At start	—	Omission not possible	The operation profile ID of the positioning data is set. The operation profile ID is set with an MC_POSITIONING_DATA_ID structure. For details on the MC_POSITIONING_DATA_ID structure, refer to the following. ☞ Page 1442 MC_POSITIONING_DATA_ID

■Input variables

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	This FB is executed when the value is TRUE.
StartDataNo	Start positioning data number	WORD (UINT)	At start	0 to 5000	0	The positioning data number from which to start is set. When this FB is executed with "0" set, it starts from positioning data number 1.
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0, 1	0	The buffer mode is set. • 0: Aborting(mcAborting) • 1: Buffered(mcBuffered) For details, refer to the following. ☞ Page 1871 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	Refer to the following. ☞ Page 1871 Options (Options)	00000000H	The function options for this FB are set in the specified bit position. For details, refer to the following. ☞ Page 1871 Options (Options)

■Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the target position is reached.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when this FB is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE when this FB is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of this FB is aborted.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used
CurrentDataNo	Executing positioning data number	WORD (UINT)	0	Indicates the positioning data number currently being executed. The value is updated when the positioning data number is switched.
PrereadingDataNo	Pre-read positioning data number	WORD (UINT)	0	The last of the pre-read positioning data numbers is stored. Positioning data numbers will be pre-read from Executing positioning data number (CurrentDataNo) to Preread positioning data number (PrereadingDataNo).
RemainingLoopCount	Number of remaining loops	WORD (UINT)	0	The number of remaining repetitions when executing the positioning data whose control method is LOOP is stored. The stored value decrements by one at the start of the loop. If the stored value is "0", the loop is not performed with the positioning data whose control method is LEND.

■Public variables

Public variable	Name	Data type	Default value	Description
InstanceID	Instance ID	INSTANCE_ID	0	The instance ID is stored. This variable is automatically set by the system when an instance is created.
ResetMcode	M code reset	BOOL	FALSE	When this variable is set to TRUE, the M code is reset. M strobe (Mstrobe) becomes FALSE after the reset.
Mstrobe	M strobe	BOOL	FALSE	This variable becomes TRUE when the M code output is valid. This variable becomes TRUE at the M code output timing (Options (Options): Bit 16). This variable becomes FALSE when M code reset (ResetMcode) is TRUE.
ValidMcode	Valid M code	WORD (UINT)	0	The valid M code (that is output in the latest status) is stored. When the M code output for the positioning data is valid, the M code is updated to that of the positioning data number at the M code output timing (Options (Options): Bit 16). *: The value is not reset when M code reset (ResetMcode) is TRUE.
StepMode	Step mode	INT (MC_STEP_MODE)	0	The step mode is set. • 0: Ignored (Ignored) • 1: Deceleration Unit Step (StepPerDec) • 2: Positioning Data Unit Step (StepPerPositioningData) For details, refer to the following. ☞ Page 1871 Step mode (StepMode)
CurrentProfileNo	Executing profile number	WORD (UINT)	0	The profile ID number currently being executed is stored. For details, refer to the following. ☞ Page 1872 Executing profile number (CurrentProfileNo)

• The following table shows the cycles at which public variables are imported or updated.

Public variable	Import/update cycle
InstanceID	Execution cycle of a POU (Program Organization Unit) that calls an FB
ResetMcode	Control cycle of an axes group structuring axis*1
Mstrobe	
ValidMcode	
StepMode	Execution cycle of a POU (Program Organization Unit) that calls an FB
CurrentProfileNo	

*1 The output variable may not always match the update timing. To adjust the update timing, call an FB from a program in the same cycle.

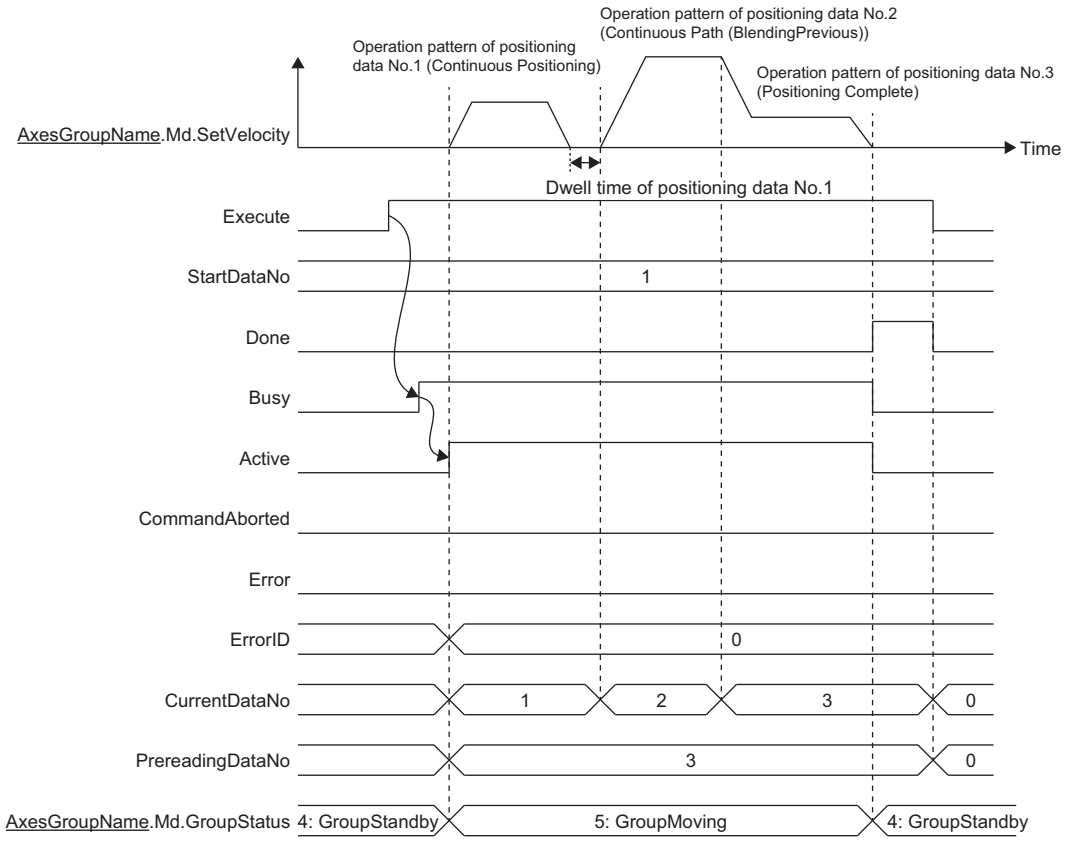
Processing details

- The continuous positioning operation specified by the operation profile data (in the format of multiple axes positioning data) is performed on the axes group.
- In the multiple axes positioning data operation, the positioning data is read in ascending order starting from Start positioning data number (StartDataNo), and the multiple axes positioning operations are internally executed continuously.
- For details on the multiple axes positioning data operation, refer to the following.

📖 User's manual for the controller used

■Timing chart

- When the processing is completed successfully



- When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following table shows the buffer modes that can be set for this FB.

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.
1: Buffered(mcBuffered)	Buffers the next FB on the under-control FB. If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after completion of the under-control FB.

Point

For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

■Options (Options)

The function options for this FB are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to 0. If a value other than 0 is set, Out of options range (error code: 1ABBH) occurs.

Bit	Name	Description
16	M code output timing	The timing for outputting the M code is set. • 0: WITH Mode • 1: AFTER Mode

• M code output timing (Bit 16)

Setting value	Description
0: WITH Mode*2	M strobe (Mstrobe) is set to TRUE when the positioning data starts positioning, and M code (Mcode) of the positioning data is stored in Valid M code (ValidMcode).
1: AFTER Mode*2	M strobe (Mstrobe) is set to TRUE when the target position is reached (or after the setting time has elapsed when the dwell time is set), and the M code of the positioning data is stored in M code (Mcode).

*2 For details on each mode, refer to the following manual.

User's manual for the controller used

■Step mode (StepMode)

The step mode is set.

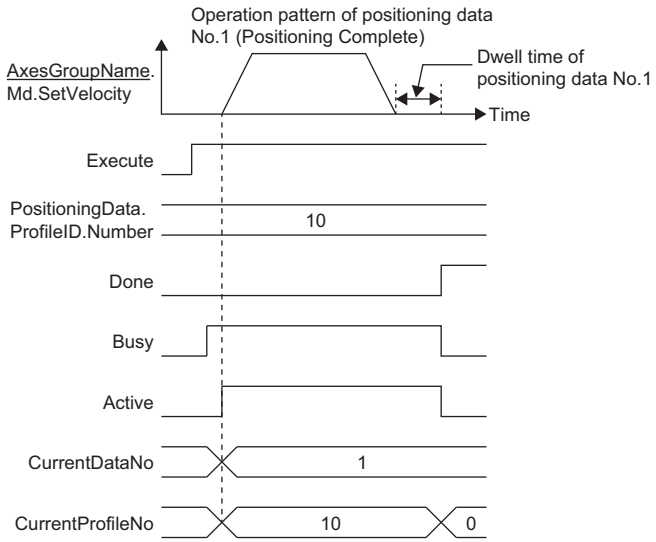
Setting value	Description
0: Ignored (Ignored)	The step function is disabled. The step mode is ignored.
1: Deceleration Unit Step (StepPerDec)	The step function is enabled. The step operation is performed at each deceleration.
2: Positioning Data Unit Step (StepPerPositioningData)	The step function is enabled. The step operation is performed for each positioning data.

■Executing profile number (CurrentProfileNo)

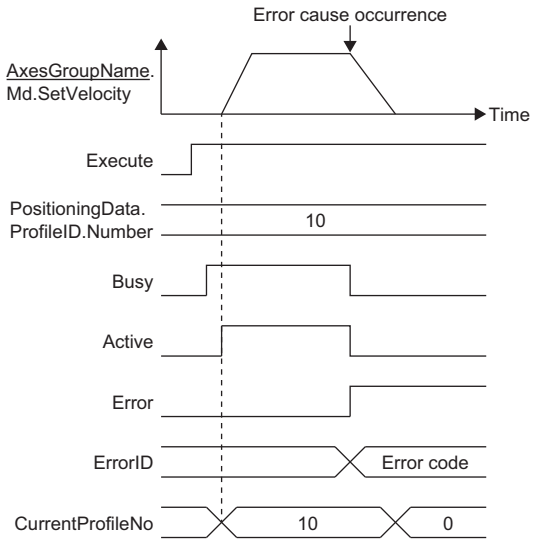
The profile ID number currently being executed is stored.

- This variable indicates that the operation profile data in the open area that is associated with the profile ID number is in use.
- The profile ID number is stored at the start of operation and cleared to 0 at the completion of the operation.

■When the FB is normal



■When an error occurred after the start of operation



- The stored value of Executing profile number (CurrentProfileNo) is not stored in Execution profile ID number (AxisName.Md.ProfileID) of the structuring axis in the axes group.

■Necessary object data

When using this FB, map the object assigned to the following object data for all structuring axes in the specified axes group to a PDO.

- Target position (607AH)

If a structuring axis that exists is the one without the object being mapped to PDO, Necessary object data unset (error code: 1AF7H) occurs and the operation will not start.

For details on object data settings, refer to the following manual.

📖 User's manual for the controller used

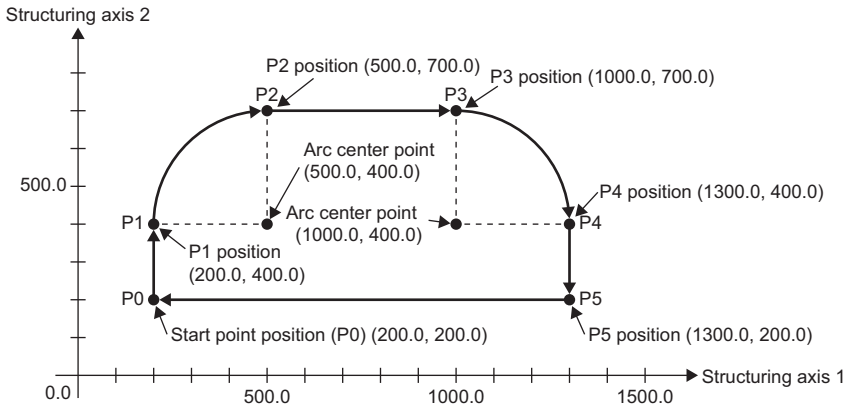
Precautions

- In this FB, the start and multiple start (Aborting) operations may be delayed due to pre-reading and the operation profile data analysis on external signal settings and other items.

Program example

This section provides examples of positioning control on Axes group 1 (AxesGroup001).

• Operation



Positioning data number	Path	Positioning operation
1	P0→P1	Positioning with Absolute Value Linear Interpolation (LinearAbsolute) from the start point position (P0) to the position (P1)
2	P1→P2	Positioning with Absolute Value Circular Interpolation (CircularAbsolute) from the position (P1) to the position (P2)
3	P2→P3	Positioning with Absolute Value Linear Interpolation (LinearAbsolute) from the position (P2) to the position (P3)
4	P3→P4	Positioning with Relative Value Circular Interpolation (CircularRelative) from the position (P3) to the position (P4)
5	P4→P5	Positioning with Relative Value Linear Interpolation (LinearRelative) from the position (P4) to the position (P5)
6	P5→P0	Positioning with Absolute Value Linear Interpolation (LinearAbsolute) from the position (P5) to the end point position (P0)

• Axes group

Item	Setting value
Structuring axis[1]	Axis0001
Structuring axis[2]	Axis0002

■Axis

Axis No.	Label name	Data type	Comment
1	Axis0001*1	AXIS_REF	Axis 1
2	Axis0002*1	AXIS_REF	Axis 2

*1 To perform circular interpolation control, set Software stroke limit target (AxisName.Pr.SwStrokeLimit_Target) of an axis parameter to valid (a value other than "-1: Invalid (Invalid)"). (In the program example, "1: Set position (SetPosition)" is set.)

■Axes group

Axes group No.	Label name	Data type	Comment
1	AxesGroup001	AXES_GROUP_REF	Axes group 1

■Operation profile data

Profile ID	Label name	Data type	Comment
1	PositioningData0001	MC_POSITIONING_DATA_REF	Multiple axes positioning data 1

• Operation profile data (PositioningData0001 (Multiple axes positioning data)) settings

Item		Positioning data number					
		1	2	3	4	5	6
Operation pattern		Continuous Path (BlendingPrevious)	Continuous Path (BlendingPrevious)	Continuous Path (BlendingPrevious)	Continuous Path (BlendingPrevious)	Continuous Positioning (ContinuousPositioning)	Positioning Complete (PositioningComplete)
Control method		Absolute Value Linear Interpolation	Absolute Value Circular Interpolation	Absolute Value Linear Interpolation	Relative Value Circular Interpolation	Relative Value Linear Interpolation	Absolute Value Linear Interpolation
Interpolation axes	Interpolation axis 1	Axis0001	Axis0001	Axis0001	Axis0001	Axis0001	Axis0001
	Interpolation axis 2	Axis0002	Axis0002	Axis0002	Axis0002	Axis0002	Axis0002
Circular interpolation mode		—	Center point specification	—	Center point specification	—	—
Path selection		—	CW	—	CW		—
Target position/ movement amount/end point	Interpolation axis 1	200.0	500.0	1000.0	300.0	0.0	200.0
	Interpolation axis 2	400.0	700.0	700.0	-300.0	-200.0	200.0
Direction selection	Interpolation axis 1	Positive direction	—	Positive direction	—	—	Positive direction
	Interpolation axis 2	Positive direction	—	Positive direction	—	—	Positive direction
Border point/ center point/ radius	Interpolation axis 1	—	500.0	—	0.0	—	—
	Interpolation axis 2	—	400.0	—	-300.0	—	—
Speed mode		Vector speed specification	—	Vector speed specification	—	Vector speed specification	Vector speed specification
Velocity		100.0	150.0	100.0	150.0	100.0	100.0
Acceleration/deceleration method		Acceleration/ deceleration specification method	—	—	—	—	—
Acceleration/deceleration time		400.0	0.0	0.0	0.0	0.0	0.0
Deceleration		500.0	0.0	0.0	0.0	0.0	0.0
Jerk		500.0	0.0	0.0	0.0	0.0	0.0
Circular interpolation error tolerance		—	—	—	—	—	—
Target position specification exceeding the ring counter		Disabled	—	Disabled	—	—	Disabled
Dwell time		500.0	0.0	0.0	0.0	0.3	0.0
M code		0	0	0	0	0	0
M code output timing override		—	—	—	—	—	—

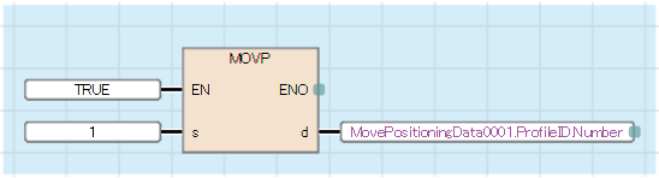
■Labels used

Label name	Data type	Comment
MCv_MovePositioningData_1	MCv_MovePositioningData	Multiple axes positioning data operation FB
bMovePosDataDone	Bit	Execution completion
bMovePosDataBusy	Bit	Executing
bMovePosDataActive	Bit	Controlling
bMovePosDataCommandAborted	Bit	Abortion of execution
bMovePosDataError	Bit	Error
uwMovePosDataErrorID	Word [unsigned]/bit string [16 bits]	Error code
uwMovePosDataCurrentDataNo	Word [unsigned]/bit string [16 bits]	Executing positioning data number
uwMovePosDataPrereadingDataNo	Word [unsigned]/bit string [16 bits]	Pre-read positioning data number
uwMovePosDataRemainingLoopCount	Word [unsigned]/bit string [16 bits]	Number of remaining loops
MC_GroupEnable_1	MC_GroupEnable	Axes Group Enabled FB
bMovePositioningDataCMD	Bit	Multiple axes positioning data operation start
bGroupEnableDone	Bit	Axes Group Enabled Completion
bGroupEnableBusy	Bit	Executing
bGroupEnableError	Bit	Error
uwGroupEnableErrorID	Word [unsigned]/bit string [16-bit]	Error code
MC_GroupDisable_1	MC_GroupDisable	Axes Group Disabled FB
bGroupDisableDone	Bit	Axes Group Disabled Completion
bGroupDisableBusy	Bit	Executing
bGroupDisableError	Bit	Error
uwGroupDisableErrorID	Word [unsigned]/bit string [16-bit]	Error code
MovePositioningData0001*1	MC_POSITIONING_DATA_ID	Multiple axes positioning data 0001

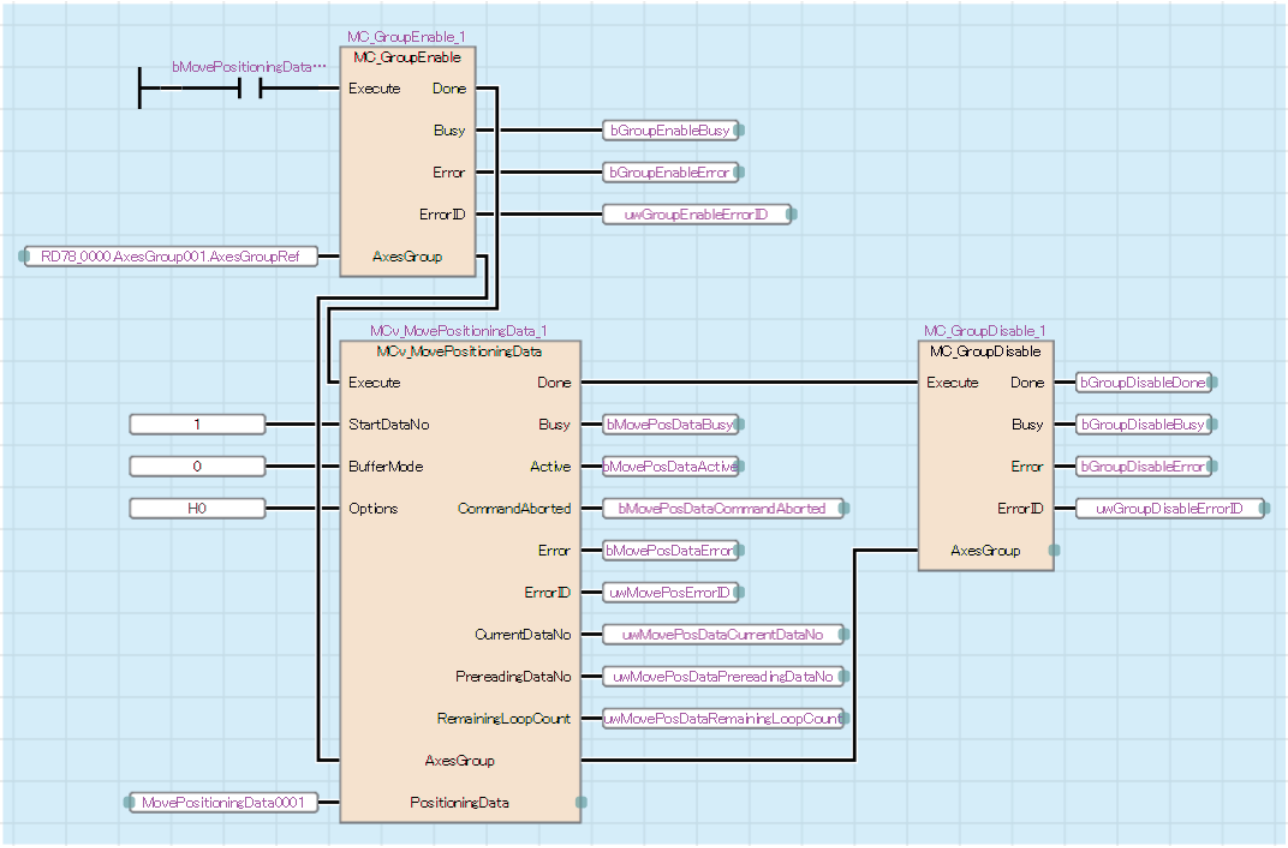
*1 Only used in FBD/LD programs. (Required to use the operation profile data in FBD/LD programs.)

■FBD/LD program

- Data setting for 2-axis linear interpolation control



- Axes Group Enabled/Multiple Axes Positioning Data Operation/Axes Group Disabled



■ST program

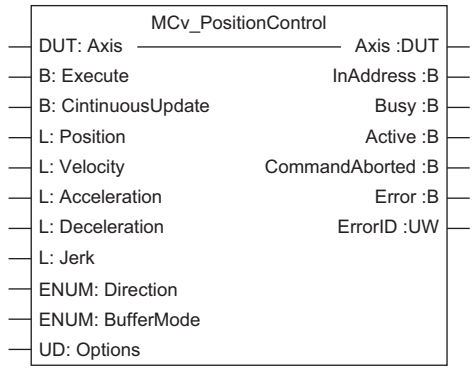
```
//-----Axes Group Enabled-----
MC_GroupEnable_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bMovePositioningDataCMD ,
   Done=> bGroupEnableDone,
   Busy=> bGroupEnableBusy ,
   Error=> bGroupEnableError ,
   ErrorID=> uwGroupEnableErrorID
);

//-----Multiple Axes Positioning Data Operation-----
MCv_MovePositioningData_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   PositioningData:= PositioningData0001.PositioningData ,
   Execute:= bGroupEnableDone ,
   StartDataNo:= 1 ,
   BufferMode:= MC_BUFFER_MODE__mcAborting ,
   Options:= H00000000 ,
   Done=> bMovePosDataDone ,
   Busy=> bMovePosDataBusy ,
   Active=> bMovePosDataActive ,
   CommandAborted=> bMovePosDataCommandAborted ,
   Error=> bMovePosDataError ,
   ErrorID=> uwMovePosDataErrorID ,
   CurrentDataNo=> uwMovePosDataCurrentDataNo ,
   PrereadingDataNo=> uwMovePosDataPrereadingDataNo ,
   RemainingLoopCount=> uwMovePosDataRemainingLoopCount
);

//-----Axes Group Disabled-----
MC_GroupDisable_1(
   AxesGroup:= AxesGroup001.AxesGroupRef ,
   Execute:= bMovePosDataDone ,
   Done=> bGroupDisableDone ,
   Busy=> bGroupDisableBusy ,
   Error=> bGroupDisableError ,
   ErrorID=> uwGroupDisableErrorID
);
```

42.26 Absolute Position Tracking Control

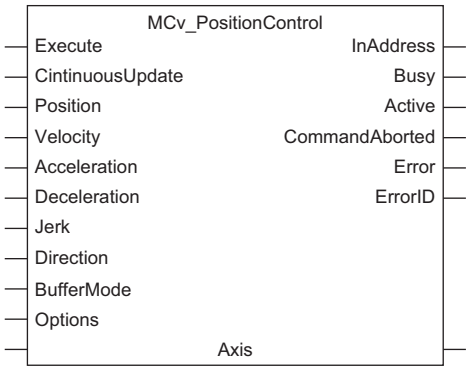
MCv_PositionControl

This function moves the specified axis to the target position according to the limit velocity, acceleration, deceleration, and jerk. In addition, even after the command has reached the target position, the axis can be moved by changing the target position with retrigger or continuous update.

Ladder



FBD/LD



ST

```
MCv_PositionControl(
  Axis:= ?AXIS_REF? ,
  Execute:= ?BOOL? ,
  ContinuousUpdate:= ?BOOL? ,
  Position:= ?LREAL? ,
  Velocity:= ?LREAL? ,
  Acceleration:= ?LREAL? ,
  Deceleration:= ?LREAL? ,
  Jerk:= ?LREAL? ,
  Direction:= ?INT? ,
  BufferMode:= ?INT? ,
  Options:= ?DWORD? ,
  InAddress=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	TRUE, FALSE	FALSE	This FB is executed when the value is TRUE.
Continuous Update	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	The target position, velocity, acceleration, and deceleration can be continuously changed while the value is TRUE.
Position	Target position	LREAL	At start/Retrigger possible/ Continuous update possible	■When the ring counter is disabled -10000000000.0 ≤ Target position < 10000000000.0 ■When the ring counter is enabled Ring counter lower limit value ≤ Target position < Ring counter upper limit value	0.0	The target position is set. For details, refer to the following. ☞ Page 1881 Target position (Position)
Velocity	Velocity	LREAL	At start/Retrigger possible/ Continuous update possible	0.0, 0.0001 to 2500000000.0	0.0	The velocity is set. For details, refer to the following. ☞ Page 1883 Velocity (Velocity)

Input variable	Name	Data type	Input	Valid range	Default value	Description
Acceleration	Acceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The acceleration is set. For details, refer to the following. ☞ Page 1883 Acceleration (Acceleration)
Deceleration	Deceleration	LREAL	At start/Retrigger possible/ Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	The deceleration is set. For details, refer to the following. ☞ Page 1883 Deceleration (Deceleration)
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	The jerk is set. For details, refer to the following. ☞ Page 1883 Jerk (Jerk)
Direction	Direction selection	INT (MC_DIRECTION[0..15])	At start/Retrigger possible/ Continuous update possible	1, 2, 3	3	The direction is set. For details, refer to the following. ☞ Page 1884 Direction selection (Direction)
BufferMode	Buffer mode selection	INT (MC_BUFFER_MODE)	At start	0, 1, 2, 3, 4, 5	0	The buffer mode is set. For details, refer to the following. ☞ Page 1887 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	00000000H to 00070021H	00000000H	The function options of this FB are set. For details, refer to the following. ☞ Page 1888 Options (Options)

■Output variables

Output variable	Name	Data type	Default value	Description
InAddress	Target position reached	BOOL	FALSE	This variable becomes TRUE when Commanded position (AxisName.Md.SetPosition) calculated by the motion system reaches the target position. When the target position has been changed when Continuous update (ContinuousUpdate) is valid (TRUE), this variable remains FALSE until the set position reaches the post-change target position.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when this FB is being executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE when this FB is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution is aborted by another FB. This variable becomes TRUE when this FB is stopped due to error occurrence or this FB is started during an error. The variable becomes FALSE when Execution command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

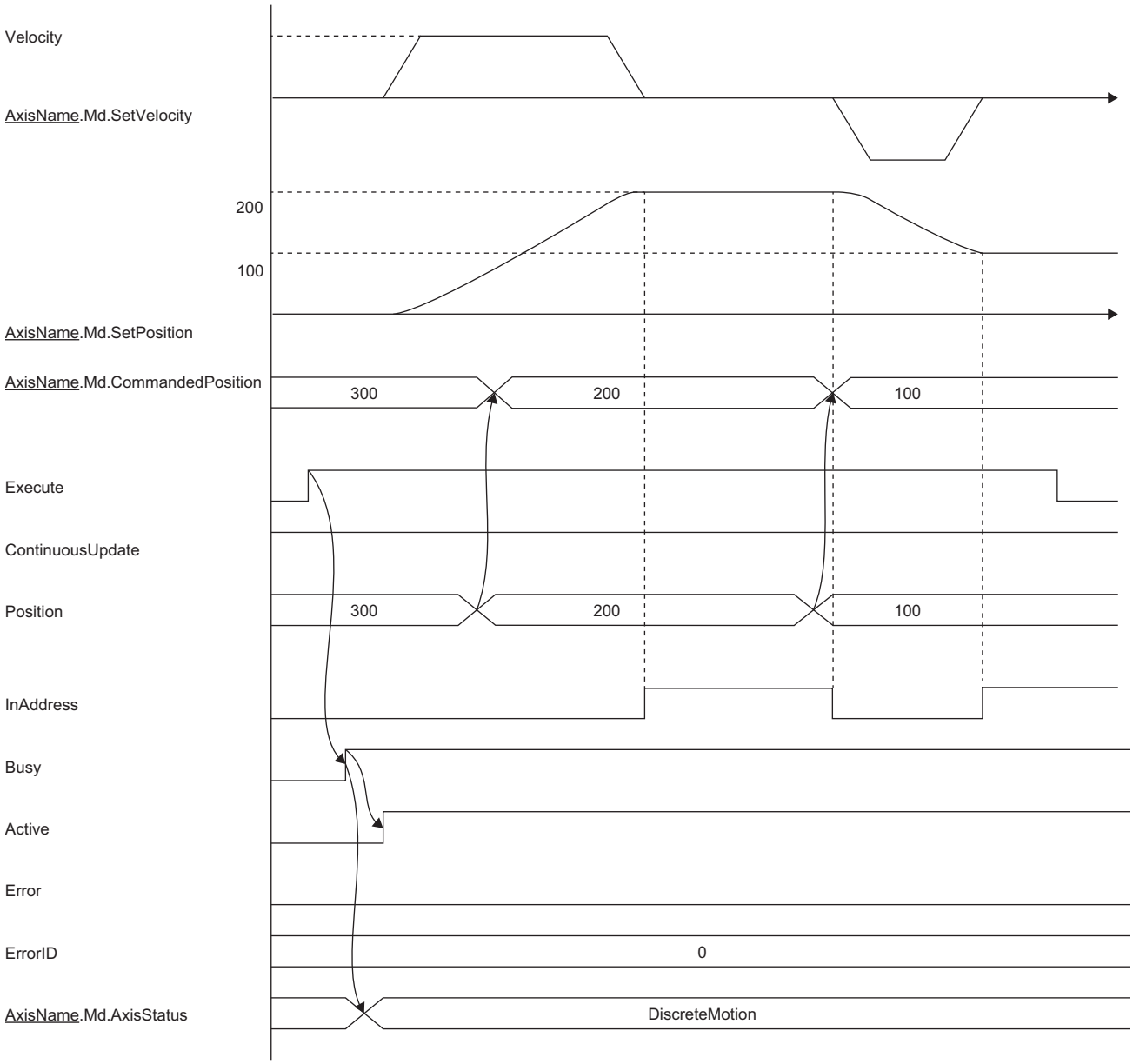
This function moves the specified Axis (Axis) to Target position (Position) according to the limit velocity, Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk). In addition, even after the command has reached the target position, the axis can be moved by changing Target position (Position) with retrigger or Continuous update (ContinuousUpdate).

When Target position (Position) is changed before the axis reaches Target position (Position), the axis moves to the target position after change.

Execution of this FB is completed when MC_Stop (Forced Stop) or other motion FB is executed.

■Timing chart

• When the processing is completed successfully



• When the processing is completed with an error

For the timing chart, refer to the following.

☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Target position (Position)

The target position of the absolute position is set according to the axis unit. When the setting value is negative, the axis moves in the negative direction. When 0 is set, the axis does not move, but Axis status (AxisName.Md.AxisStatus) of the target axis becomes "5: During positioning operation (DiscreteMotion)".

The following table summarizes the valid ranges. If a value outside the range is set, Out of target position range (error code: 1A85H) occurs and the axis will not start.

Ring counter setting	Valid range
Invalid	-10000000000.0 ≤ Target position < 10000000000.0
Valid	Ring counter lower limit value ≤ Target position < Ring counter upper limit value

When the software stroke limit is valid (when Software stroke limit target (AxisName.Md.SwStrokeLimit_Target) is set to a value other than "-1: Invalid (Invalid)")

- Software stroke limit over (target position) (error code: 1A80H) occurs and the axis will not start if a position which exceeds the software stroke limit is specified as the target position.
- The Direction selection (Direction) setting is ignored and positioning control is performed in a direction which does not cross over into the area outside the software stroke limit range.
- When the value does not cross over into the area outside the software stroke limit range both in the positive direction and the negative direction, positioning control is performed in the direction closest to Target position (Position) (the direction in which the absolute value of the movement amount is shortest) based on the current position.
- When the distance is the same between the positive direction and negative direction, the moving direction of the motion control FB executed the last time or the moving direction when the target position was changed in the previous execution will be carried over for the operation. If the previous moving direction is not available, the axis will move in the positive direction.

When the software stroke limit is invalid (when Software stroke limit target (AxisName.Md.SwStrokeLimit_Target) is set to "-1: Invalid (Invalid)"), positioning control is performed within the valid range.

The target position change will be ignored one operation cycle before Target position reached (InAddress) becomes TRUE.

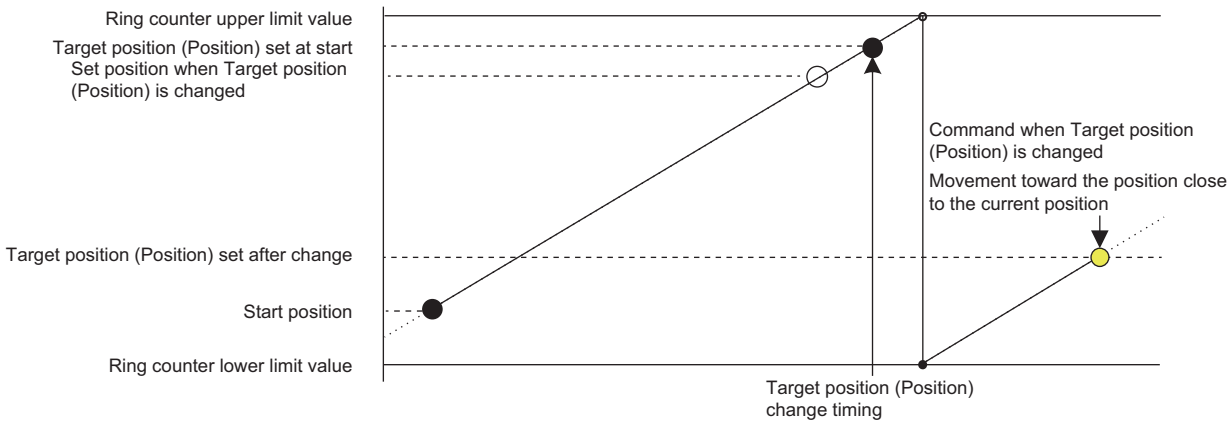
If Target position (Position) is changed, the changed operation varies depending on the Direction selection (Direction) setting.

- If a value outside the range is set for Target position (Position) during execution of the motion control FB, Out of Position/Distance setting range warning (event code: 00D0DH) occurs and the operation continues with the previous setting.
- When Out of Position/Distance setting range warning (event code: 00D0DH) occurs, if Direction selection (Direction) has been changed, the operation continues with the previous Direction selection (Direction) setting.
- When Out of direction selection range warning (event code: 00D4DH) occurs, the operation continues with the previous Target position (Position) setting.

Ex.

When the shortest path is set, the axis moves from the set position at the time of the change in the direction closest to Target position (Position) after the change.

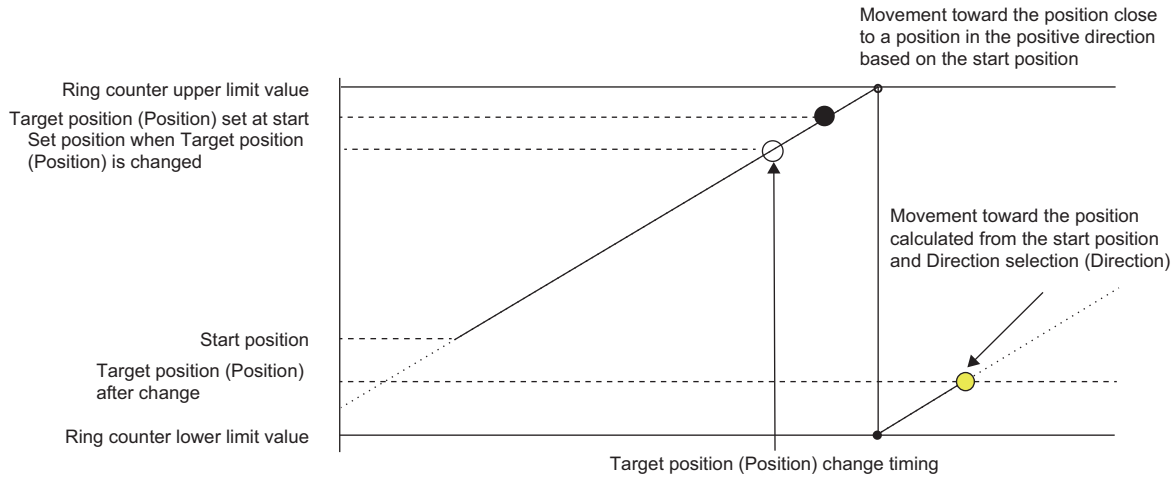
"Start" in the following refers to the start of the motion control FB after the first operation and the change of Target position (Position) after Target position reached (InAddress) becomes TRUE in the previous operation.



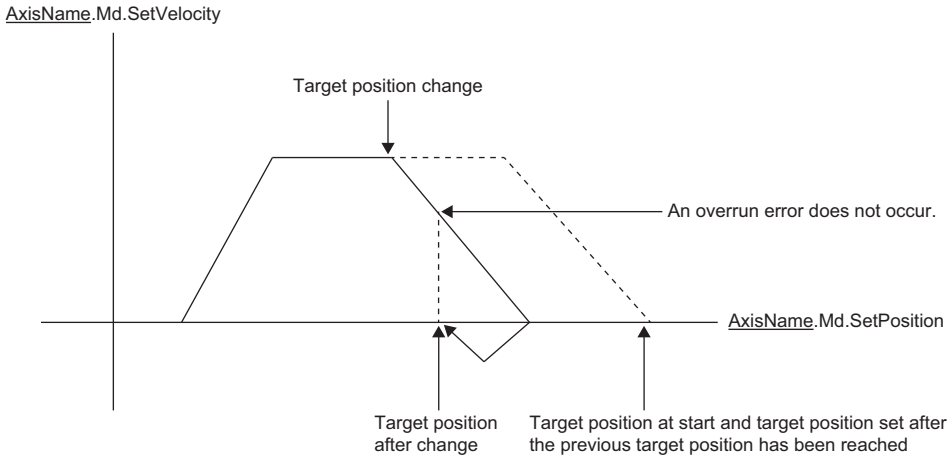
Ex.

When a path other than the shortest path is set, Target position (Position) is calculated from the set position and Direction selection (Direction) at the time of change.

"Start" in the following refers to the start of the motion control FB after the first operation and the change of Target position (Position) after Target position reached (InAddress) becomes TRUE in the previous operation.



If the reverse operation is performed due to the change in Target position (Position), or if the axis has passed Target position (Position) after the change during the change, the axis decelerates to stop. After the deceleration stop is completed, the operation starts toward the changed Target position (Position). In that case, no overrun error will occur.



■Velocity (Velocity)

The command velocity is set for this FB.

Setting range
0.0, 0.0001 to 2500000000.0

When the setting is omitted or set as 0.0, the axis does not operate, but Axis status (AxisName.Md.AxisStatus) changes to "6: During continuous operation (ContinuousMotion)".

If a value outside the range is set, Out of velocity range (error code: 1ABAH) occurs and the axis will not start.

For details on the operation while the motion control FB is being executed, refer to the user's manual for the controller used.

■Acceleration (Acceleration)

The acceleration is set for this FB. For details on the acceleration/deceleration method, refer to the user's manual for the controller used.

If a value outside the range is set, Out of acceleration range (error code: 1A8DH) occurs and the axis will not start.

For details on the operation while the motion control FB is being executed, refer to the user's manual for the controller used.

■Deceleration (Deceleration)

The deceleration is set for this FB. For details on the acceleration/deceleration method, refer to the user's manual for the controller used.

If a value outside the range is set, Out of deceleration range (error code: 1A8EH) occurs and the axis will not start.

For details on the operation while the motion control FB is being executed, refer to the user's manual for the controller used.

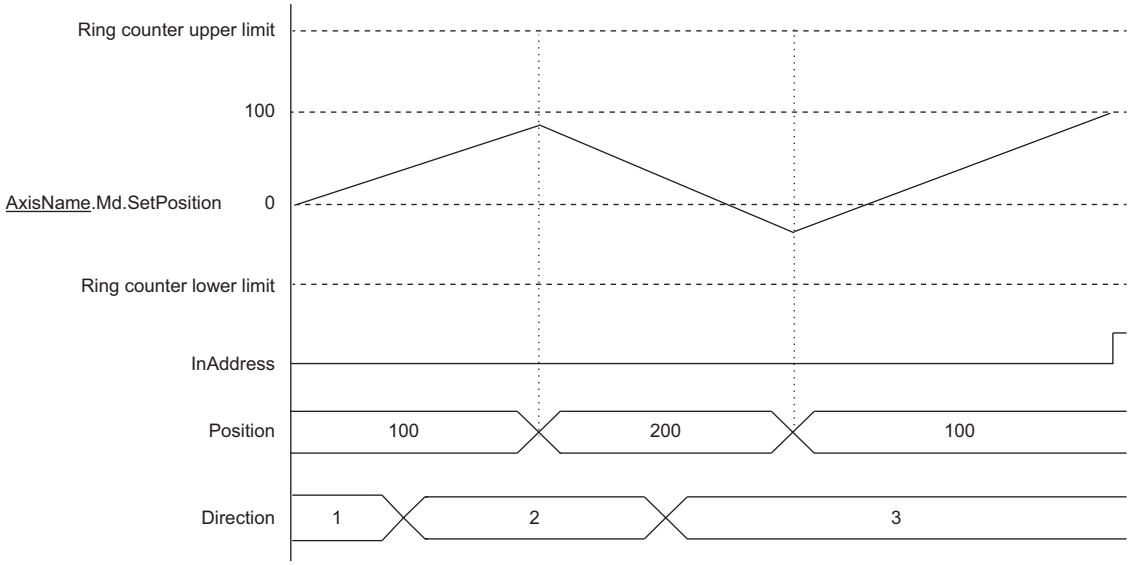
■Jerk (Jerk)

The jerk is set for this FB. For details on the acceleration/deceleration method, refer to the user's manual for the controller used.

If a value outside the range is set, Out of jerk range (error code: 1A8FH) occurs and the axis will not start.

■Direction selection (Direction)

The operation direction is specified. When continuous update or retrigger is performed, Direction selection (Direction) will not be reflected at the timing when the value is changed, but will be reflected when Target position (Position) is changed.



Choose one of the following:

- 1: Positive direction (mcPositiveDirection)
- 2: Negative direction (mcNegativeDirection)
- 3: Shortest path (mcShortestWay)

If a value outside the range is set for Direction selection (Direction) at start, Out of direction selection range (error code: 1AA5H) occurs and the axis will not start.

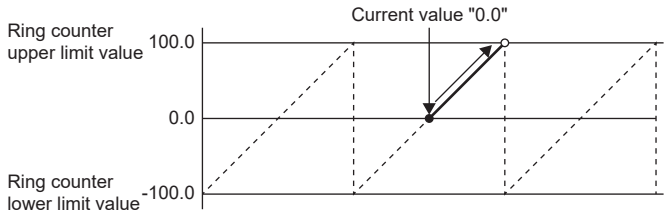
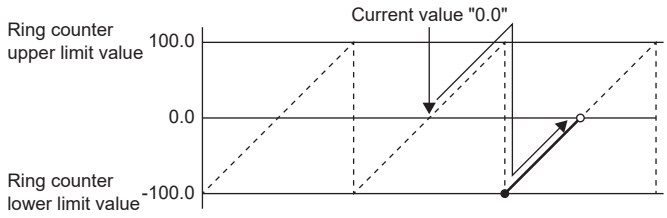
If a value outside the range is set for Direction selection (Direction) during control, Out of direction selection range warning (event code: 00D4DH) occurs and the operation continues with the previous setting. If Target position (Position) is changed at the same time as Direction selection (Direction), the operation continues with the previous Target position (Position) setting.

The setting for Direction selection (Direction) is ignored when the software stroke limit is valid.

When the software stroke limit is invalid, the direction used to move from the current position to Target position (Position) can be set by selecting from "1: Positive direction (mcPositiveDirection)", "2: Negative direction (mcNegativeDirection)", or "3: Shortest path (mcShortestWay)" in Direction selection (Direction).

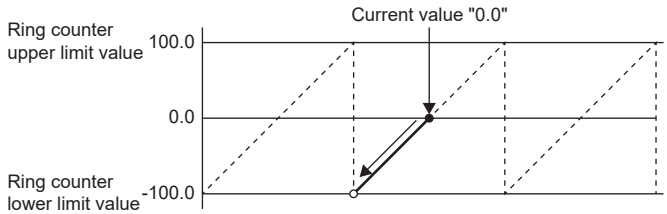
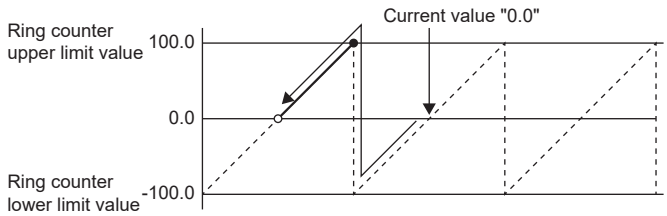
• 1: Positive direction (mcPositiveDirection)

Positioning is performed from the current position to the target position in the positive direction (address increasing).

Setting range	Operation
Current value ≤ Target position < Ring counter upper limit value	
Ring counter lower limit value ≤ Target position < Current value	

• 2: Negative direction (mcNegativeDirection)

Positioning is performed from the current position to the target position in the negative direction (address decreasing).

Setting range	Operation
Ring counter lower limit value ≤ Target position ≤ Current value	
Current value < Target position < Ring counter upper limit value	

• 3: Shortest path (mcShortestWay)

Based on the current position, positioning control is performed in the direction closest to the target position (the direction in which the absolute value of the movement amount is shortest). If the distance is the same in both the positive direction and the negative direction, control is performed using the current direction. The setting range of the target position is "Ring counter lower limit value ≤ Target position < Ring counter upper limit value". If a value outside the range is set, Out of target position range (error code: 1A85H) occurs and the axis will not start.

Ex.

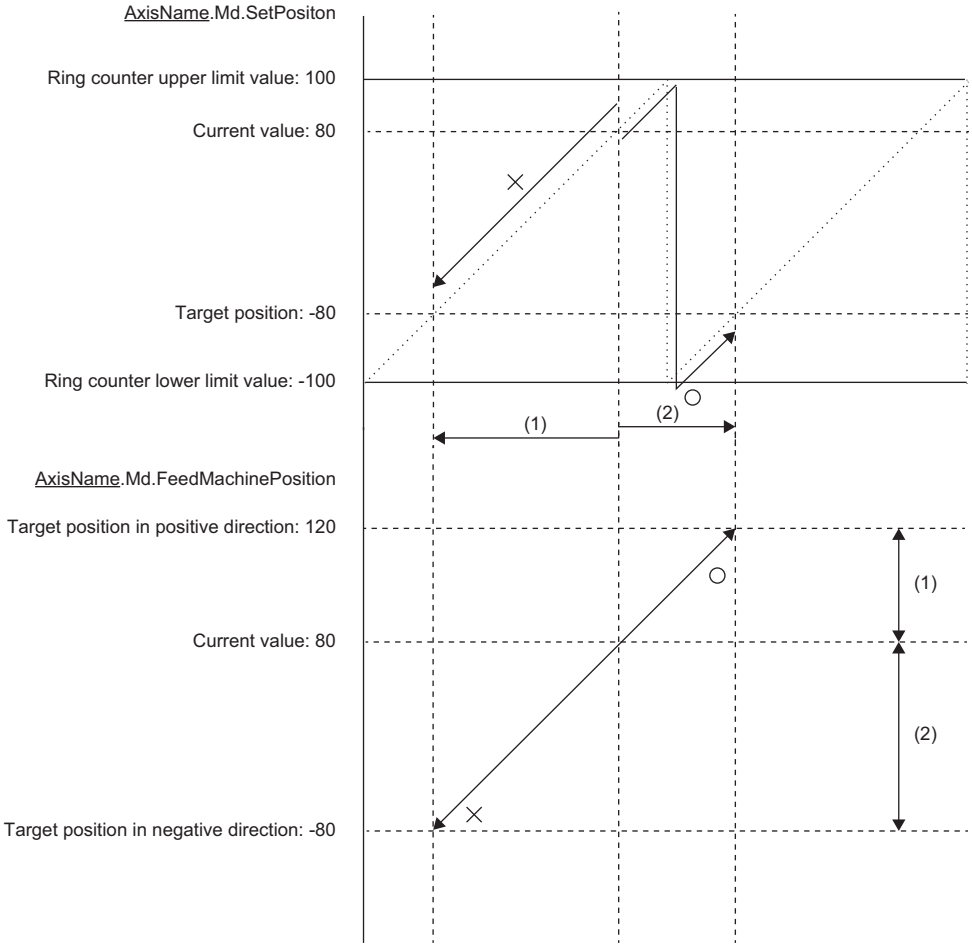
Operation when set to the values shown below

· Ring counter upper limit value: 100.0

· Ring counter lower limit value: -100.0

· Current position: 80.0

· Target position: -80.0



Since the movement amount of (1) is "160.0" and the movement amount of (2) is "40.0", positioning moves in the positive direction from (2).

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following table shows the buffer modes that can be set for this FB.

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.
1: Buffered(mcBuffered)	Buffers the next FB on the under-control FB. If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after completion of the under-control FB.
2: BlendingLow(mcBlendingLow)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the lower target velocity between the under-control FB and the buffering FB is used as the switching speed.
3: BlendingPrevious(mcBlendingPrevious)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the under-control FB is used as the switching speed.
4: BlendingNext(mcBlendingNext)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the target velocity of the buffering FB is used as the switching speed.
5: BlendingHigh(mcBlendingHigh)	Buffers the next FB on the under-control FB.*1 If the under-control FB already has an FB buffering on it, subsequent FBs are buffered consecutively. (Up to 2.) Buffering FBs are executed in order after the under-control FB reaches the target position. When the under-control FB reaches the target position, the higher target velocity between the under-control FB and the buffering FB is used as the switching speed.

*1 Stops are not performed between the under-control FB and the buffering FB.

Point

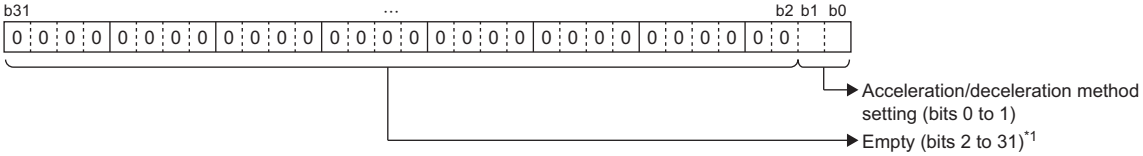
For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

■Options (Options)

The function options for this FB are set in the specified bit position.

The following figure shows how the settings should be configured with bit specification.



*1 Set empty areas to "0". If a value other than "0" is set, Out of Options Range (error code: 1ABBH) occurs.

Bit	Name	Description
0 to 1	Acceleration/deceleration method setting	The acceleration/deceleration method used to perform control is set. • 0: Acceleration/deceleration specification method (mcAccDec) • 1: Acceleration/deceleration time-fixed method (mcFixedTime)

• Acceleration/deceleration method setting (bit 0 to 1)

Setting value	Description
0: Acceleration/deceleration specification method (mcAccDec)	An acceleration/deceleration method using Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk) set in this FB
1: Acceleration/deceleration time-fixed method (mcFixedTime)	An acceleration/deceleration method using the acceleration/deceleration time set in this FB regardless of the velocity The acceleration/deceleration time does not use Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk).

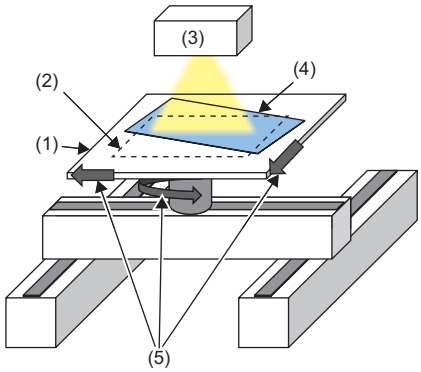
Precautions

If Target position (Position) and Direction selection (Direction) are changed at the same time, data inconsistency may occur. By changing Direction selection (Direction) before changing Target position (Position), both changes can be applied when Target position (Position) is changed. If the values of Target position (Position) and Direction selection (Direction) need to be changed at the same time, execute this FB with the program execution cycle set to "Fixed Scan" or restart this FB.

Program example

The following is a program example. This FB is used to move a workpiece on the XYθ table to the reference position based on the correction information of the sensor (θ/X/Y axis correction values).

• Operation



(1): XYθ table
(2): Reference position
(3): Sensor
(4): Workpiece
(5): The position is corrected by position tracking based on the correction information from the sensor.

■Axis

Setting item	X axis	Y axis	θ axis
Axis type	Real drive axis	Real drive axis	Real drive axis
Position command unit	μm	μm	degree

■Labels used

Label name	Data type	Comment
bMCv_PositionControl_Execute	Bit	Control start
leMCv_PositionControl_Pos_1	Double-precision real number	Input to MCv_PositionControl_1.Position
leMCv_PositionControl_Pos_2	Double-precision real number	Input to MCv_PositionControl_2.Position
leMCv_PositionControl_Pos_3	Double-precision real number	Input to MCv_PositionControl_3.Position
leCorrectionValue_Theta	Double-precision real number	θ-axis correction value (Input correction value obtained from the sensor)
leCorrectionValue_XPos	Double-precision real number	X-axis correction value (Input correction value obtained from the sensor)
leCorrectionValue_YPos	Double-precision real number	Y-axis correction value (Input correction value obtained from the sensor)

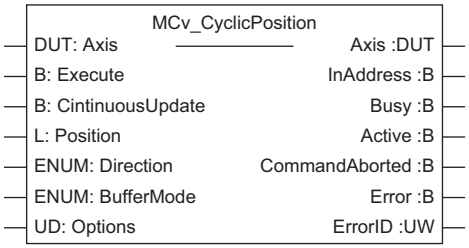
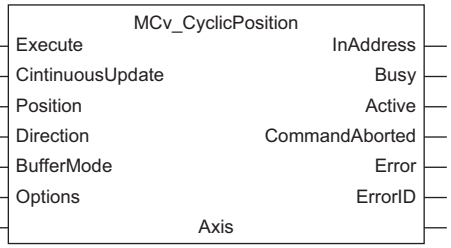
■ST program

```
//Input correction value from the sensor
leMCv_PositionControl_Pos_1:= leCorrectionValue_Theta;
leMCv_PositionControl_Pos_2:= leCorrectionValue_XPos;
leMCv_PositionControl_Pos_3:= leCorrectionValue_YPos;

//-----Axis 1: Absolute Position Tracking Control-----
MCv_PositionControl_1 (
  Axis:=  Axis0001.AxisRef ,
  Execute:= bMCv_PositionControl_Execute,
  ContinuousUpdate:= TRUE ,
  Position:= leMCv_PositionControl_Pos_1
  Velocity:= 100000.0 ,
  Acceleration:= 1000.0 ,
  Deceleration:= 1000.0 ,
  Jerk:= 0.0 ,
  Direction:= MC_DIRECTION__mcPositiveDirection ,
  BufferMode:= MC_BUFFER_MODE__mcAborting ,
  Options:= 0
);

//-----Axis 2: Absolute Position Tracking Control-----
MCv_PositionControl_2 (
  Axis:=  Axis0002.AxisRef ,
  Execute:= bMCv_PositionControl_Execute,
  ContinuousUpdate:= TRUE ,
  Position:= leMCv_PositionControl_Pos_2
  Velocity:= 10000000.0 ,
  Acceleration:= 100000.0 ,
  Deceleration:= 100000.0 ,
  Jerk:= 0.0 ,
  Direction:= MC_DIRECTION__mcPositiveDirection ,
  BufferMode:= MC_BUFFER_MODE__mcAborting ,
  Options:= 0
);

//-----Axis 3: Absolute Position Tracking Control-----
MCv_PositionControl_3 (
  Axis:=  Axis0003.AxisRef ,
  Execute:= bMCv_PositionControl_Execute,
  ContinuousUpdate:= TRUE ,
  Position:= leMCv_PositionControl_Pos_3
  Velocity:= 10000000.0 ,
  Acceleration:= 100000.0 ,
  Deceleration:= 100000.0 ,
  Jerk:= 0.0 ,
  Direction:= MC_DIRECTION__mcPositiveDirection ,
  BufferMode:= MC_BUFFER_MODE__mcAborting ,
  Options:= 0
);
```

42.27 Motion Cyclic Position Control

MCv_CyclicPosition

This FB sends the target position to a drive unit according to the specified target position.

Ladder	FBD/LD	ST
		MCv_CyclicPosition(Axis:= ?AXIS_REF? , Execute:= ?BOOL? , ContinuousUpdate:= ?BOOL? , Position:= ?LREAL? , Direction:= ?INT? , BufferMode:= ?INT? , Options:= ?DWORD? , InAddress=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	—	FALSE	This FB is executed when the value is TRUE.
Continuous Update	Continuous update	BOOL	At start	—	FALSE	The target position can be continuously changed while the value is TRUE.
Position	Target position	LREAL	At start/ Retrigger possible/ Continuous update possible	■When the ring counter is invalid -10000000000.0 ≤ Setting value < 10000000000.0 ■When the ring counter is valid Ring counter lower limit value ≤ Setting value < Ring counter upper limit value	0	The target position of the absolute position is set in the positioning range. For details, refer to the following. ☞ Page 1894 Target position (Position)
Direction	Direction selection	MC_DIRECTION	At start/ Retrigger possible/ Continuous update possible	1, 2, 3	0	When the software stroke limit is disabled, the direction used to move from the current position to the target position is specified. For details, refer to the following. ☞ Page 1895 Direction selection (Direction)
BufferMode	Buffer mode	MC_BUFFER_MODE	At start	mcAborting	mcAborting	The buffer mode is set. For details, refer to the following. ☞ Page 1898 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	Fixed to 0	0	This variable cannot be set for this FB. Set 0. If a value other than 0 is set, Out of options range (error code: 1ABBH) occurs.

■Output variables

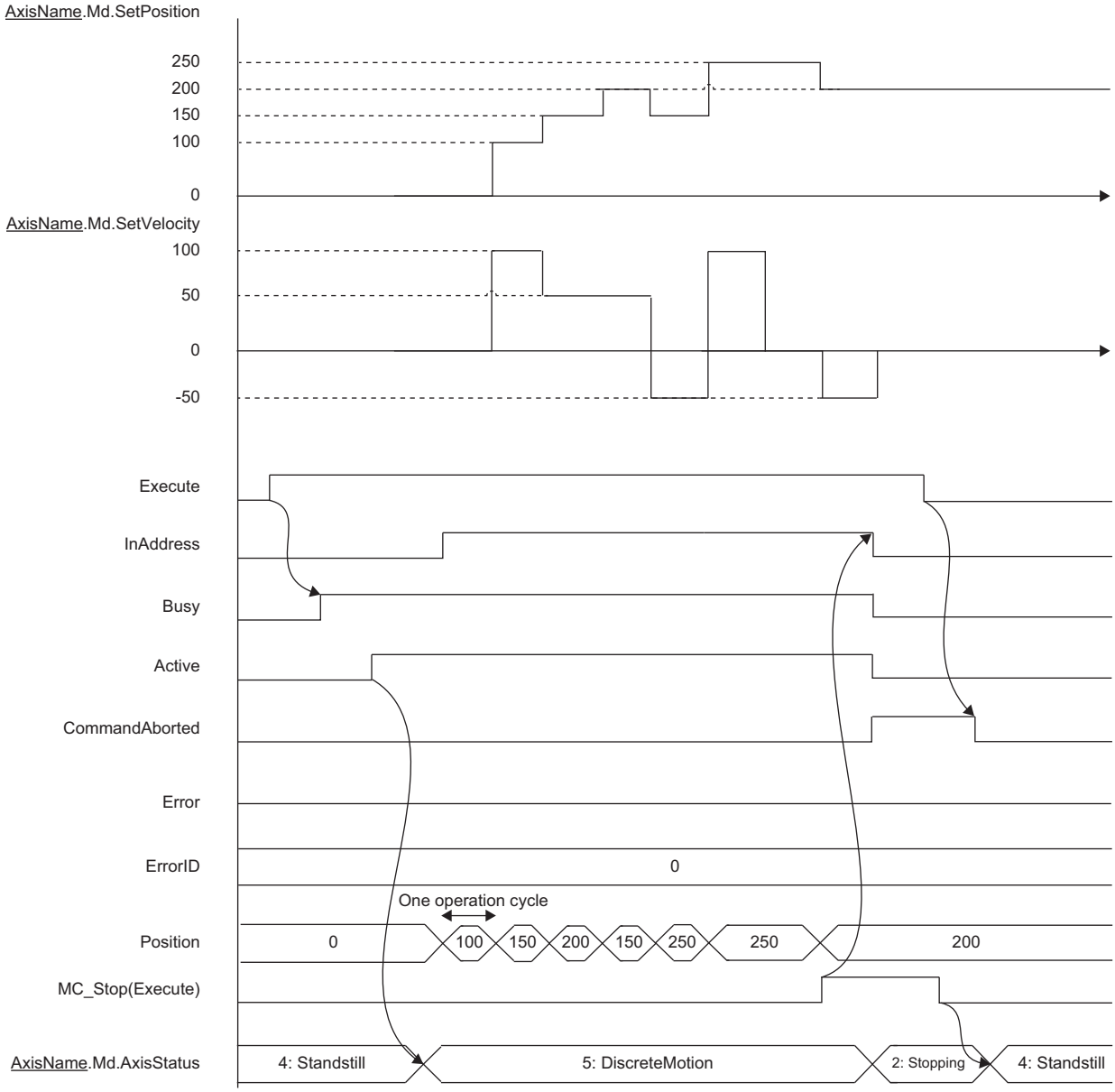
Output variable	Name	Data type	Default value	Description
InAddress	Target position reached	BOOL	FALSE	This variable becomes TRUE when AxisName.Md.SetPosition reaches the target position. This variable becomes FALSE when an error or a stop cause occurs.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when this FB is operating.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE when this FB is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution is aborted by another FB.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. User's manual for the controller used

Processing details

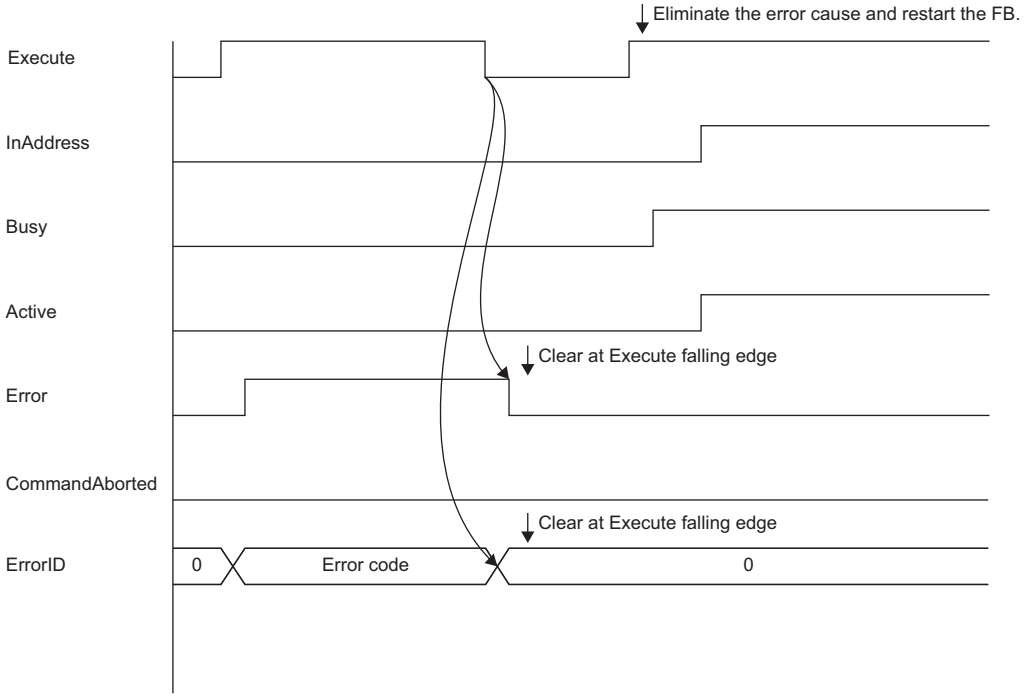
This FB switches the control mode on the drive unit side to csp (cyclic synchronous position mode) and sends the command position. To stop this FB, use MC_Stop (Forced Stop).

■Timing chart

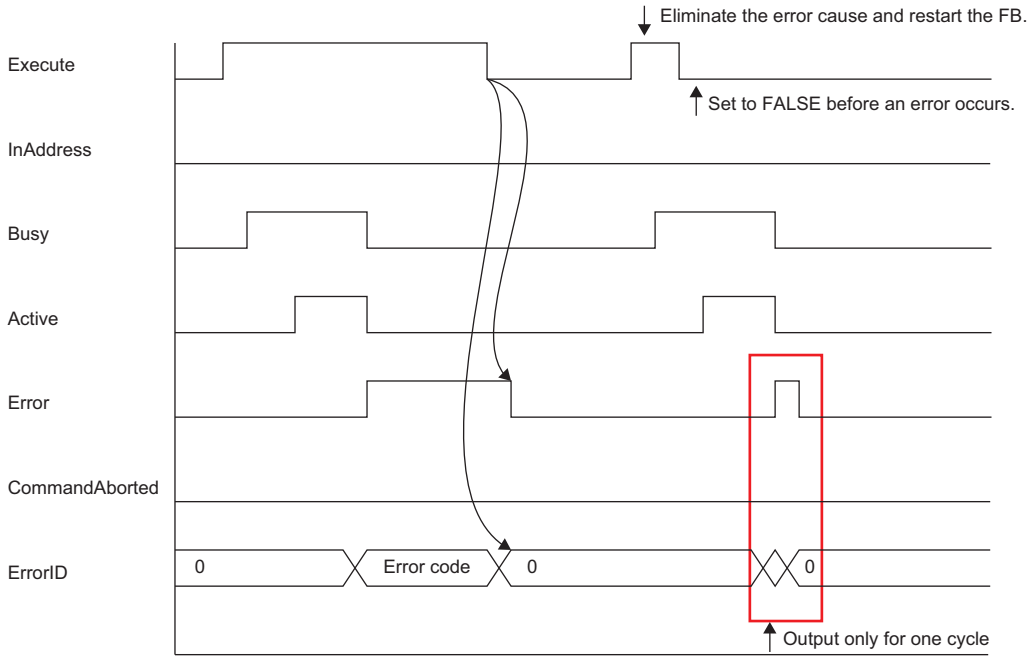
- When the processing is completed successfully



• When I/O variables are abnormal



• When an input variable is abnormal



■Target position (Position)

The target position of the absolute position is set in the positioning range. The following table summarizes the valid ranges. If a value outside the range is set, Out of target position range (error code: 1A85H) occurs.

Ring counter setting	Valid range
Invalid	-10000000000.0 ≤ Target position < 10000000000.0
Valid	Ring counter lower limit value ≤ Target position < Ring counter upper limit value

When the software stroke limit is valid (when Software stroke limit target (AxisName.Md.SwStrokeLimit_Target) is set to a value other than "-1: Invalid (Invalid)"), positioning control is performed in a direction which does not cross over into the area outside the software stroke limit range. When the value does not cross over into the area outside the software stroke limit range both in the positive direction and the negative direction, positioning control is performed in the direction closest to the target position (the direction in which the absolute value of the movement amount is shortest) based on the current position. When the distance is the same between the positive direction and negative direction, the moving direction of the motion control FB executed previously or the moving direction at the previous target position change will be carried over for the operation. In addition, even when the value set is within the valid range, Software stroke limit over (target position) (error code: 1A80H) occurs if a position which exceeds the software stroke limit is specified as the target position.
When the software stroke limit is invalid (when Software stroke limit target (AxisName.Md.SwStrokeLimit_Target) is set to "-1: Invalid (Invalid)"), refer to the following.
☞ Page 1895 Direction selection (Direction)
When changing Target position (Position), change it within the following range. If the setting is configured so that the value after the electronic gear conversion exceeds 31 bits, Out of the movement amount range after driver position unit conversion (error code: 1B3CH) occurs.
Change amount of target position × Driver unit conversion numerator/Driver unit conversion denominator ≤ 2000000000

Ex.
An example of calculating the target position under the following conditions is shown below.

Item	Value
Motor encoder resolution	67108864 [pulse/rev]
Worm gear pitch	1000 [μm]
External gear ratio	1/10

When the electronic gear settings specify the number of pulses per servo motor rotation as 67108864 [pulse] and the movement amount per servo motor rotation as 100.0 [μm], the change amount of the target position is within or falls below the following range.

$$\text{Target position variation amount} = 2000000000 \div \frac{\text{Driver unit conversion numerator}}{\text{Driver unit conversion denominator}} = 2000000000 \div \frac{\text{Number of pulses per rotation}}{\text{Movement amount per rotation}} = 2000000000 \div \frac{67108864}{100.0} = 2980.23\ldots$$

■Direction selection (Direction)

When the software stroke limit is disabled, the direction used to move from the current position to the target position is specified.

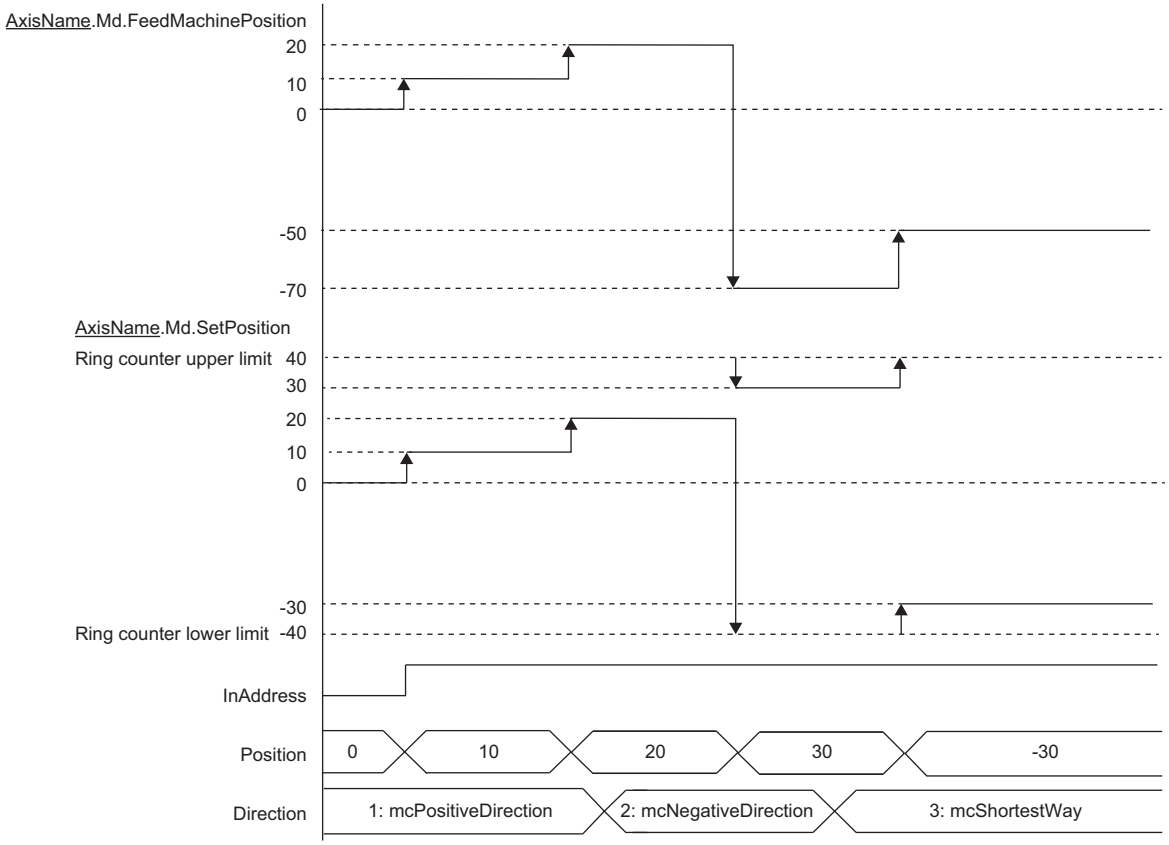
The setting range can be selected from "1: Positive direction (mcPositiveDirection)", "2: Negative direction (mcNegativeDirection)", and "3: Shortest path (mcShortestWay)".

The setting for Direction selection (Direction) is ignored when the software stroke limit is valid.

If a value outside the range is set for Direction selection (Direction) at start, Out of direction selection range (error code: 1AA5H) occurs and the axis will not start.

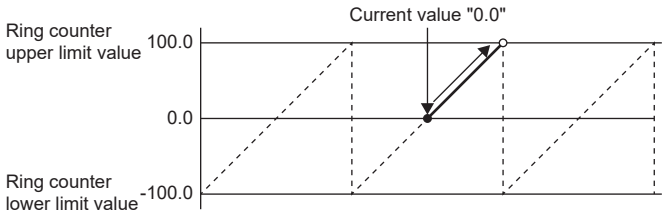
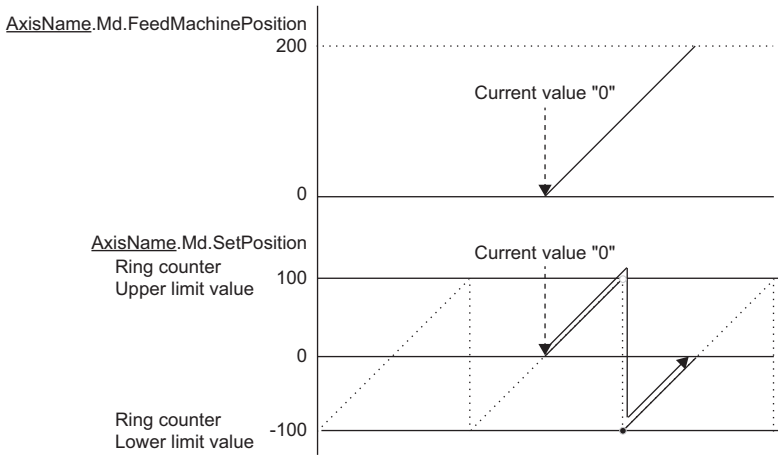
If a value outside the range is set for Direction selection (Direction) during control, Out of direction selection range warning (event code: 00D4DH) occurs and the operation continues with the previous setting. If Target position (Position) is changed at the same time as Direction selection (Direction), the operation continues with the previous Target position (Position) setting.

For the moving direction according to Direction selection (Direction), the operation at the start of the motion control FB is based on the set position at the start, and the operation after the start of the motion control FB is based on the set position at the target position change.



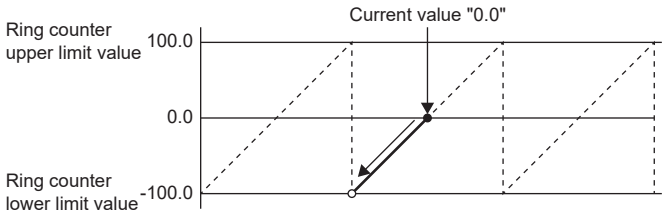
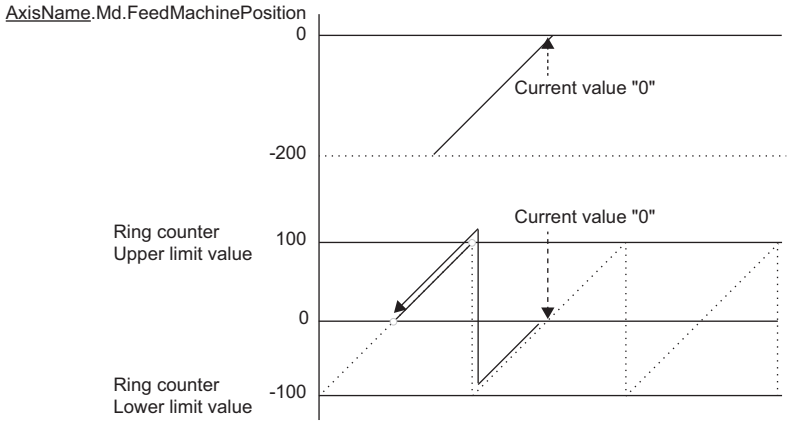
• 1: Positive direction (mcPositiveDirection)

Positioning is performed from the current position to the target position in the positive direction (address increasing).

Setting range	Operation
Current value ≤ Target position < Ring counter upper limit value	
Ring counter lower limit value ≤ Target position < Current value	

• 2: Negative direction (mcNegativeDirection)

Positioning is performed from the current position to the target position in the negative direction (address decreasing).

Setting range	Operation
Ring counter lower limit value ≤ Target position ≤ Current value	
Current value < Target position < Ring counter upper limit value	

• 3: Shortest path (mcShortestWay)

Based on the current position, positioning control is performed in the direction closest to the target position (the direction in which the absolute value of the movement amount is shortest). If the distance is the same in both the positive direction and the negative direction, control is performed using the current direction. The setting range of the target position is "Ring counter lower limit value ≤ Target position < Ring counter upper limit value". If a value outside the range is set, Out of target position range (error code: 1A85H) occurs and the axis will not start.

Ex.

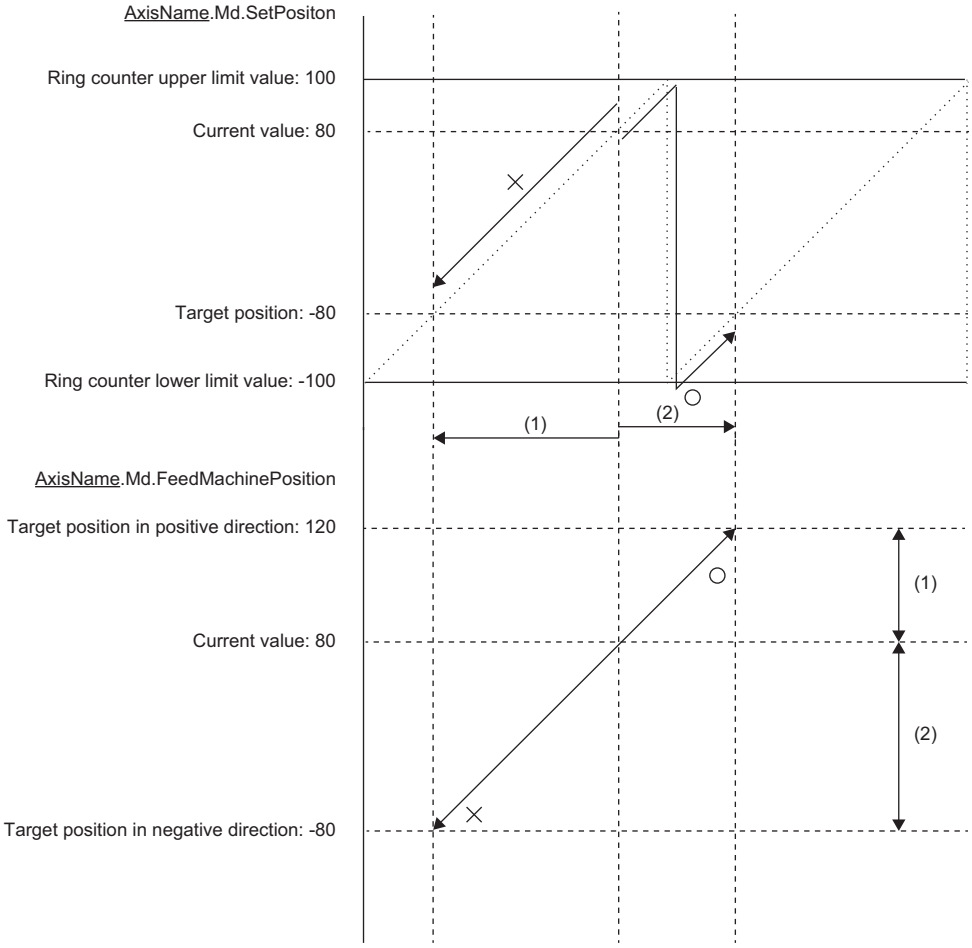
Operation when set to the values shown below

· Ring counter upper limit value: 100.0

· Ring counter lower limit value: -100.0

· Current position: 80.0

· Target position: -80.0



Since the movement amount of (1) is "160.0" and the movement amount of (2) is "40.0", positioning moves in the positive direction from (2).

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following table shows the buffer modes that can be set for this FB.

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.

Point

For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

Precautions

- Motion cyclic synchronous position control does not set acceleration, deceleration, or jerk and sends only the commanded position, so rapid acceleration and deceleration may occur. Therefore, set the speed limit value (Positive Direction Speed Limit Value (AxisName.Pr.VelocityLimit_Positive) for the positive direction and Negative Direction Speed Limit Value (AxisName.Pr.VelocityLimit_Negative) for the negative direction), taking safety into consideration.
- If the set velocity exceeds the speed limit value for the axis, the operation follows Operation setting at speed limit value exceeded (AxisName.Pr.VelocityLimit_OverOperation). Also, if the absolute value of the set velocity exceeds the maximum velocity, Velocity range over during controlling (error code: 1B1AH) occurs and the operation stops.
- When switching to csp from a control mode other than csp by performing multiple start with MCv_CyclicPosition (Motion Cyclic Position Control), check that the motor has stopped before switching. If switching is performed without stopping the motor, the current position and MCv_CyclicPosition (Motion Cyclic Position Control) may be largely different, which may result in sudden movement.

Program example

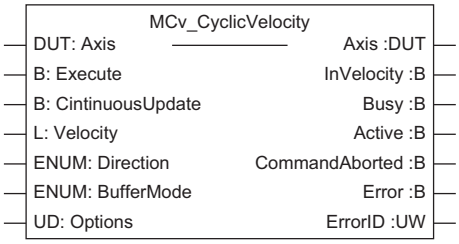
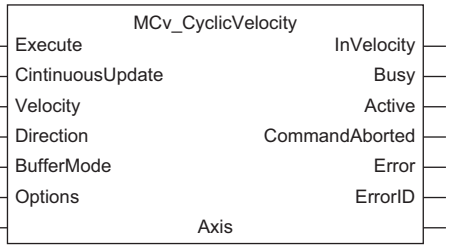
For a program example of this FB, refer to the following manual.

User's manual for the controller used

42.28 Motion Cyclic Velocity Control

MCv_CyclicVelocity

This FB sends the target velocity to a drive unit according to the specified target velocity.

Ladder	FBD/LD	ST
		MCv_CyclicVelocity(Axis:= ?AXIS_REF? , Execute:= ?BOOL? , ContinuousUpdate:= ?BOOL? , Velocity:= ?LREAL? , Direction:= ?INT? , BufferMode:= ?INT? , Options:= ?DWORD? , InVelocity=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	—	FALSE	This FB is executed when the value is TRUE.
Continuous Update	Continuous update	BOOL	At start	—	FALSE	The target position can be continuously changed while the value is TRUE.
Velocity	Target velocity	LREAL	At start/Retrigger possible/ Continuous update possible	-2500000000.0 to 2500000000.0	0	The target velocity is set. For details, refer to the following. ☞ Page 1902 Target velocity (Velocity)
Direction	Direction selection	MC_DIRECTION	At start	mcPositiveDirection (Positive direction) (1), mcNegativeDirection (Negative direction) (2)	0	The direction is specified. For details, refer to the following. ☞ Page 1902 Direction selection (Direction)
BufferMode	Buffer mode	MC_BUFFER_MODE	At start	mcAborting	mcAborting	The buffer mode is set. For details, refer to the following. ☞ Page 1902 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	Fixed to 0	0	This variable cannot be set for this FB. Set 0. If a value other than 0 is set, Out of options range (error code: 1ABBH) occurs.

■Output variables

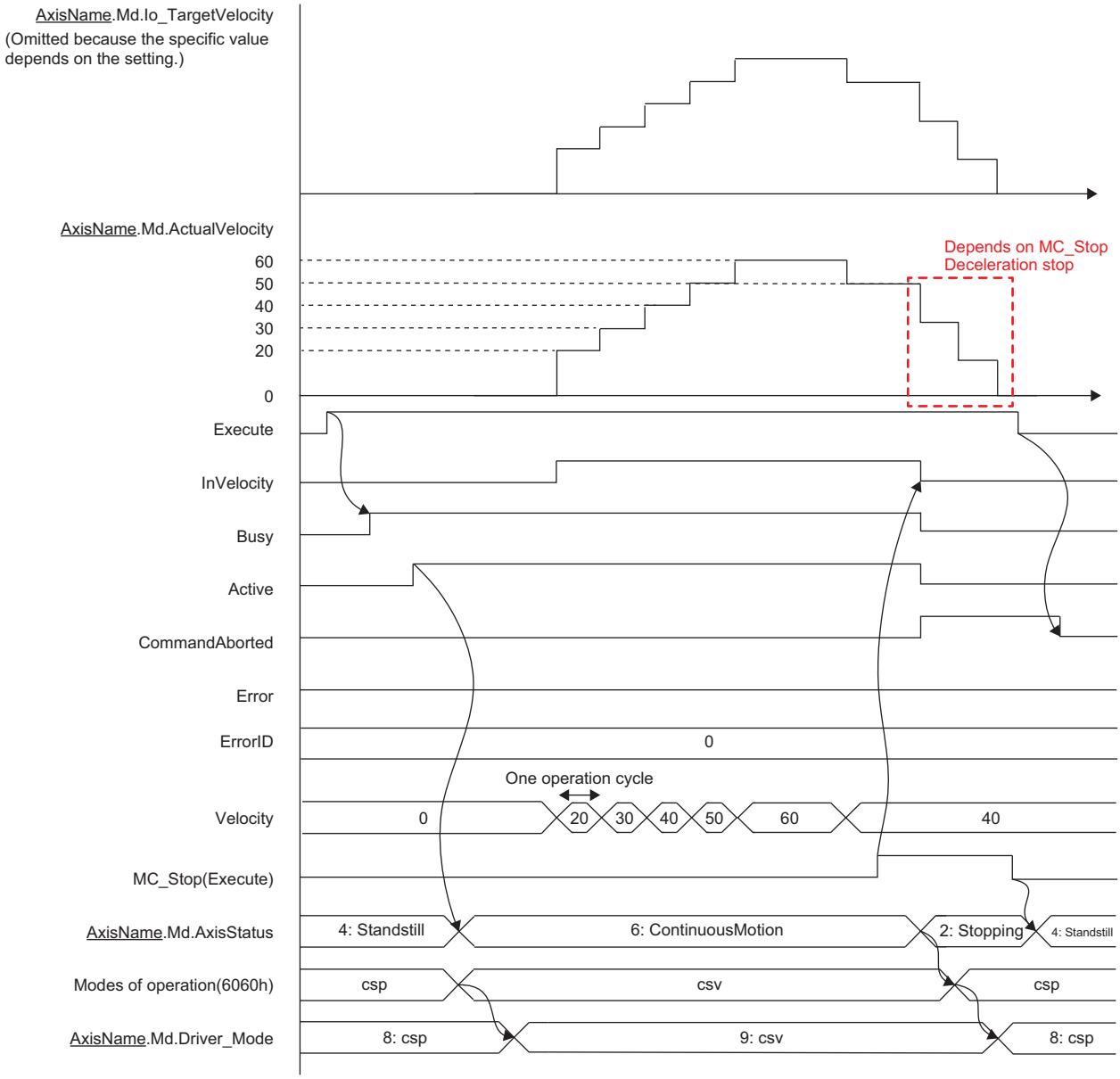
Output variable	Name	Data type	Default value	Description
InVelocity	Target velocity reached	BOOL	FALSE	This variable becomes TRUE when AxisName.Md.IO_TargetVelocity reaches the target velocity. This variable becomes FALSE when an error or a stop cause occurs.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when this FB is being executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE when this FB is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution is aborted by another FB.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

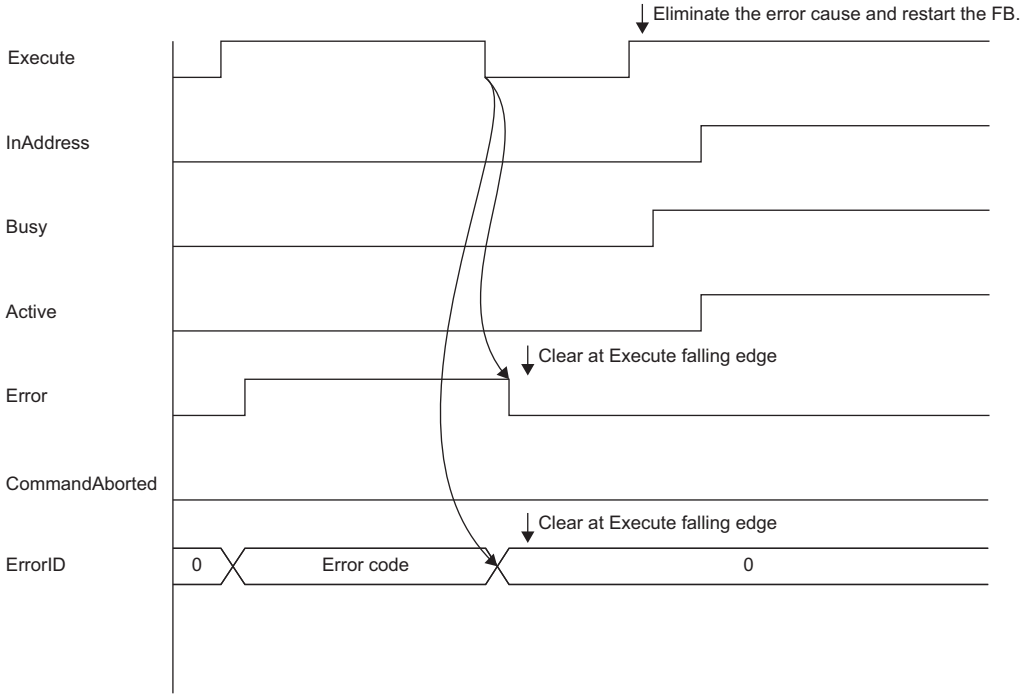
This FB switches the control mode on the drive unit side to csv (cyclic synchronous velocity mode) and sends the command velocity. To stop this FB, use MC_Stop (Forced Stop).

■Timing chart

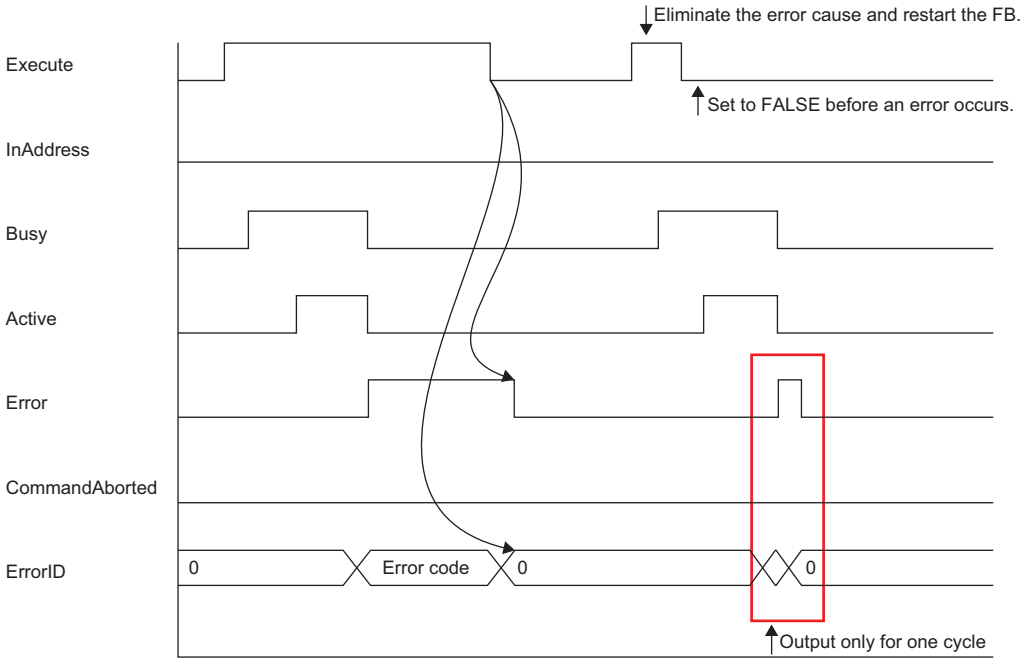
- When the processing is completed successfully



• When I/O variables are abnormal



• When an input variable is abnormal



■Target velocity (Velocity)

The target velocity is set. If a value outside the range is set at the start, Out of velocity range (error code: 1ABAH) occurs. If a value outside the range is set during control, Out of velocity range warning (event code: 00D0FH) occurs.
Set the value so that the command velocity after driver velocity unit conversion falls within the range from -2147483648 to 2147483647. If the command velocity after conversion is outside the range, Set velocity of converted driver velocity unit out of range (error code: 1A90H) occurs regardless of the operation status.
The command velocity after driver velocity unit conversion is determined by the following equation:
Command velocity after driver velocity unit conversion = Velocity command unit × (Driver unit conversion numerator ÷ Driver unit conversion denominator)
When the velocity is set to a negative value, the axis moves in the negative direction.
When 0 is set at the start, the axis does not operate, but Axis status (AxisName.Md.AxisStatus) of the target axis changes to "6: During continuous operation (ContinuousMotion)".
When this FB is started multiple times and Target velocity (Velocity) is set to 0, the axis operates at the command velocity of the previous motion control FB.

■Direction selection (Direction)

The direction used to move from the current position to the target position is specified.
The setting range can be selected from "1: Positive direction (mcPositiveDirection)", "2: Negative direction (mcNegativeDirection)", and "3: Shortest path (mcShortestWay)". When "2: Negative direction (mcNegativeDirection)" is set and Target velocity (Velocity) is set to a negative value, the movement direction of the motor will be positive.
If a value outside the range is set for Direction selection (Direction) at start, Out of direction selection range (error code: 1AA5H) occurs and the axis will not start.

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.
The following table shows the buffer modes that can be set for this FB.

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.

Point

For details on multiple start (buffer mode), refer to the following.
User's manual for the controller used

Precautions

When setting the target velocity, set the value so that the difference between the current velocity and the target velocity is not too large. If the difference is too large, the axis may suddenly accelerate.
When a stop cause occurs while this FB is starting, the axis will stop immediately.

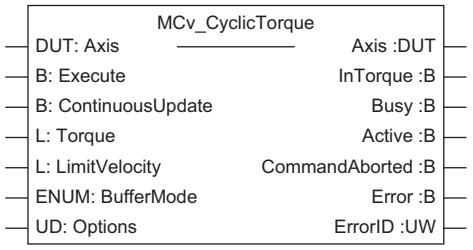
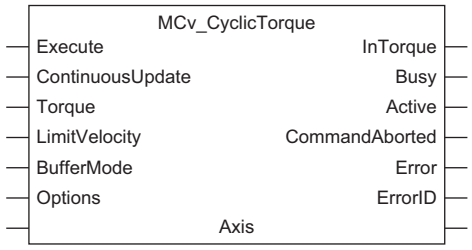
Program example

For a program example of this FB, refer to the following.
User's manual for the controller used

42.29 Motion Cyclic Torque Control

MCv_CyclicTorque

The target torque is sent to the drive unit according to the specified target torque.

Ladder	FBD/LD	ST
		MCv_CyclicTorque(Axis:= ?AXIS_REF? , Execute:= ?BOOL? , ContinuousUpdate:= ?BOOL? , Torque:= ?LREAL? , LimitVelocity:= ?LREAL? , BufferMode:= ?INT? , Options:= ?DWORD? , InTorque=> ?BOOL? , Busy=> ?BOOL? , Active=> ?BOOL? , CommandAborted=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Valid range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Omission not possible	An axis is specified. For the variable (AxisName.AxisRef.) used, refer to the following. ☞ Page 1398 AxisName.AxisRef. (Axis information)

■Input variable

Input variable	Name	Data type	Input	Valid range	Default value	Description
Execute	Execution command	BOOL	At start	—	FALSE	This FB is executed when the value is TRUE.
ContinuousUpdate	Continuous update	BOOL	At start	—	FALSE	The target position can be continuously changed while the value is TRUE.
Torque	Target torque	LREAL	At start/Retrigger possible/ Continuous update possible	-1000.0 to 1000.0 [%]	0	The ratio of the servo motor used to the rated torque is set. For details, refer to the following. ☞ Page 1907 Target torque (Torque)
LimitVelocity	Limit velocity	LREAL	At start/Retrigger possible/ Continuous update possible	-2500000000.0 to 2500000000.0	0	The direction is specified. For details, refer to the following. ☞ Page 1907 Limit velocity (LimitVelocity)
BufferMode	Buffer mode	MC_BUFFER_MODE	At start	mcAborting	mcAborting	The buffer mode is set. For details, refer to the following. ☞ Page 1907 Buffer mode (BufferMode)
Options	Options	DWORD(HEX)	At start	Fixed to 0	0	This variable cannot be set for this FB. Set 0. If a value other than 0 is set, Out of options range (error code: 1ABBH) occurs.

■Output variables

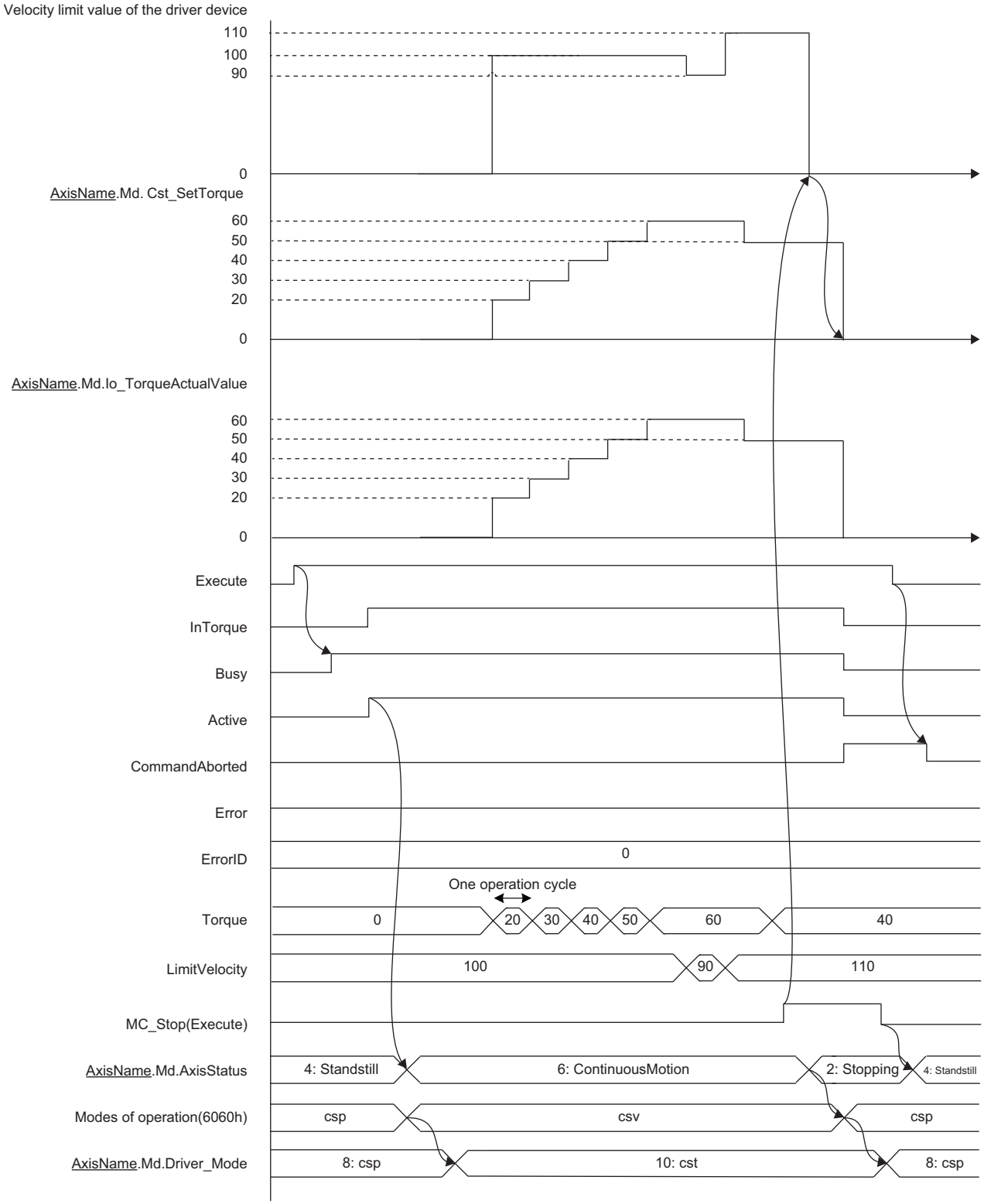
Output variable	Name	Data type	Default value	Description
InTorque	Target torque reached	BOOL	FALSE	This variable becomes TRUE when Cst_SetTorque sent from a drive unit reaches the target torque. This variable becomes FALSE after the axis stops due to error or stop cause occurrence.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when this FB is operating.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE when this FB is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution is aborted by another FB.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. User's manual for the controller used

Processing details

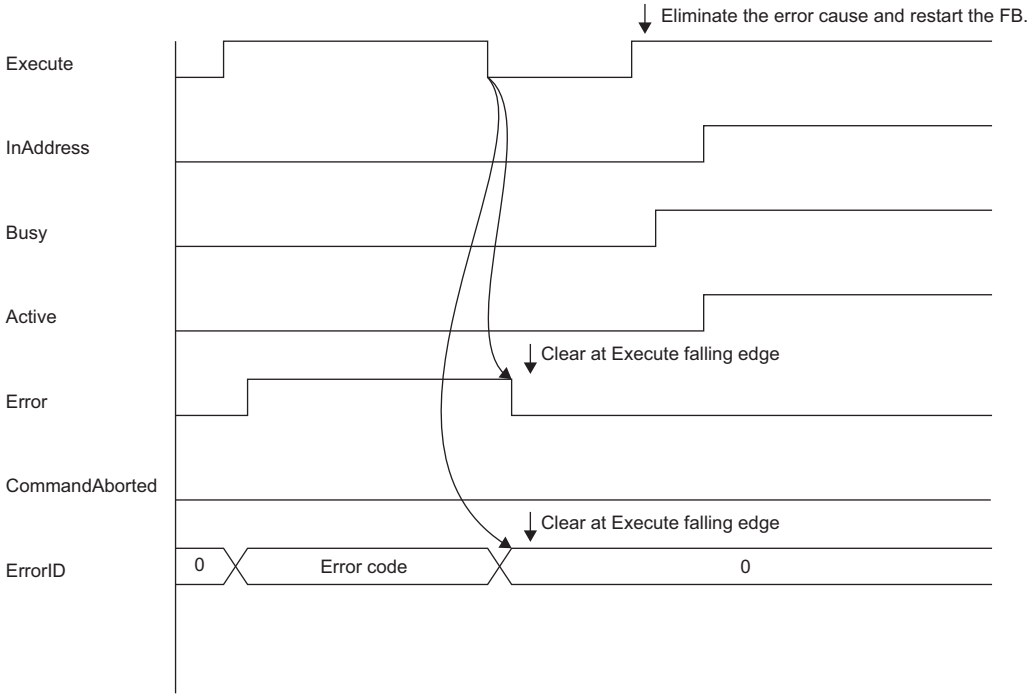
This FB switches the control mode on the drive unit side to cst (cyclic synchronous torque mode) and sends the command velocity. To stop this FB, use MC_Stop (Forced Stop).

■Timing chart

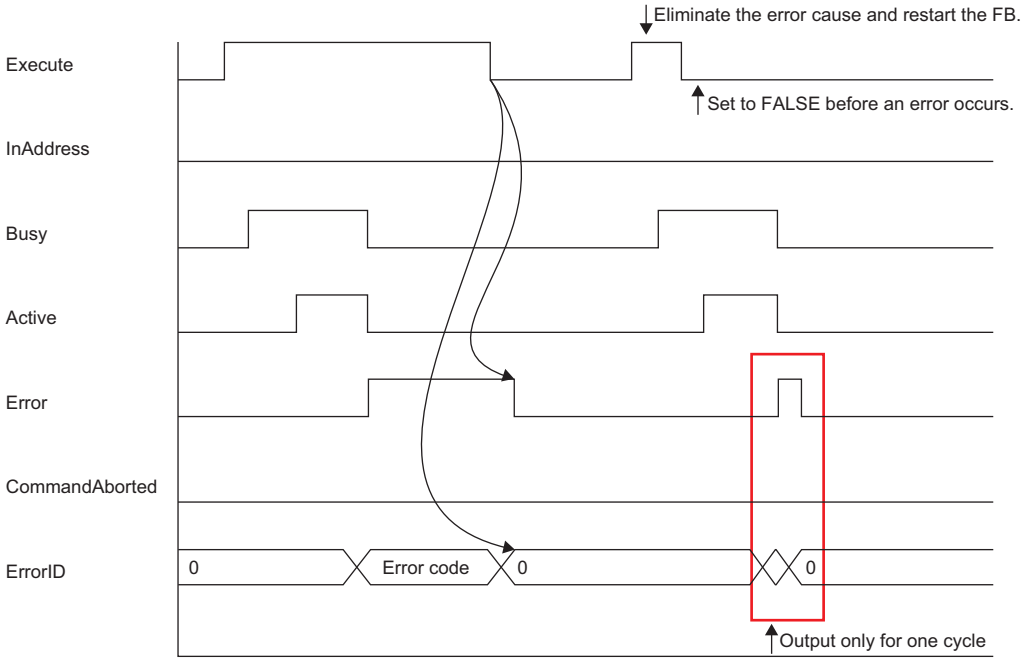
• When the processing is completed successfully



• When I/O variables are abnormal



• When an input variable is abnormal



■Target torque (Torque)

The ratio of the servo motor used to the rated torque is set in percentage.

If a value outside the range is set at the start, Out of target torque range (error code: 1A96H) occurs.

If a value outside the range is set during control, Out of torque setting range warning (event code: 00D16H) occurs.

The number of significant digits for the target torque varies depending on the specifications for Target Torque of the connected drive unit. If the value set for the target torque contains a value equal to or less than the number of significant digits of Target Torque, it is rounded down.

To output the target torque in the positive direction (address increasing direction), set a positive value. To output the target torque in the negative direction (address decreasing direction), set a negative value.

To reverse the target torque direction for safety reasons, set the target torque to 0 once.

Ex.

To change the target torque from -10.0 to 10.0, change the value in the following order.

-10.0 → 0.0 → 10.0

■Limit velocity (LimitVelocity)

The velocity limit value for the cyclic torque mode is set in the range from -2500000000.0 to 2500000000.0. If a value outside the range is set, Out of limit velocity range (event code: 1A86H) occurs at start, and Out of limit velocity range warning (event code: 00D1BH) occurs during control.

Set the value so that the command velocity after driver velocity unit conversion does not go out of the range from -2147483648 to 2147483647 [0.01rpm]. If a value outside the range is set, Set velocity of converted driver velocity unit out of range (error code: 1A90H) occurs.

The command velocity after driver velocity unit conversion is determined by the following equation:

Command velocity after driver velocity unit conversion = Velocity command unit × (Driver unit conversion numerator ÷ Driver unit conversion denominator)

When 0 is set at the start, the axis does not operate, but Axis status (AxisName.Md.AxisStatus) of the target axis changes to "6: During continuous operation (ContinuousMotion)".

When MC_CyclicTorque (torque control) is executed by multiple start and 0 is stored in LimitVelocity (LimitVelocity), the motor speed of the previous motion control FB is controlled as Limit velocity (LimitVelocity).

Limit velocity (LimitVelocity) becomes valid when the following conditions are met.

- An object corresponding to Limit velocity (LimitVelocity) during cyclic synchronous torque mode (cst) of the connected drive unit is mapped and assigned to a PDO.
- The mapped object data is set to vVelLimitValue (AxisName.PrConst.SlaveObject.vVelLimitValue).

The setting method for enabling the velocity limit with Limit velocity (LimitVelocity) varies depending on the drive unit.

- When using the MR-J5(W)-G, map and assign Velocity limit value to a PDO.
- When using a drive unit other than the MR-J5(W)-G, map and assign an object corresponding to the limit velocity during cyclic synchronous torque mode (cst) of the connected drive unit to a PDO. Then, set the mapped object data to vVelLimitValue (AxisName.PrConst.SlaveObject.vVelLimitValue).

When the velocity limit in Limit velocity (LimitVelocity) is invalid, the velocity limit of the controller cannot be valid.

- When using the MR-J5(W)-G, use an engineering tool to set the setting value of the servo parameter [Pr.PT67 Speed limit].
- When using a drive unit other than the MR-J5(W)-G, use the velocity limit function of the connected drive unit. For details, refer to the user's manual for the connected drive unit.

■Buffer mode (BufferMode)

The operation used to perform multiple start (buffer mode) is set.

The following table shows the buffer modes that can be set for this FB.

Setting value	Description
0: Aborting(mcAborting)	Aborts (cancels) the under-control FB and executes the next FB immediately.

Point

For details on multiple start (buffer mode), refer to the following.

User's manual for the controller used

Precautions

- The set position and the feed machine position are updated by follow up.
- The switching time of the control mode by this FB varies depending on the specifications for the drive unit used.
- When a stop cause occurs while this FB is starting, the axis will stop immediately.
- Do not start a positioning control FB while this FB is operating. Check that Driver control mode (AxisName.Md.Driver_Mode) has switched to "10: Torque control (cst)" before starting the positioning control FB.
- If the torque set as the target torque is insufficient, the torque may suddenly drop depending on the mechanism or load. When setting a small value for the target torque, take sufficient care for safety.
- When the axis stops, store the value in Limit velocity (LimitVelocity) after checking that the control mode is switched. If the value is already stored in Limit velocity (LimitVelocity) before the mode is switched, the velocity may suddenly increase, creating a dangerous situation.
- Limit velocity (LimitVelocity) is recommended during cst. When Limit velocity (LimitVelocity) is not used, a large difference between the current torque and the target torque may cause the velocity to suddenly increase, depending on the mechanism and load application, creating a dangerous situation.
- For the torque limit change during cst, pay attention to the operations when the following changes are made.

Change content	Operation
At restart or continuous update, the target torque was changed to a value bigger than the torque limit value (Positive direction torque limit value (AxisName.Md.TorqueLimit_Positive) or Negative direction torque limit value (AxisName.Md.TorqueLimit_Negative)).	Torque limit value over warning (event code: 00D12H) occurs, and the operation is performed with the value before the change.
During torque control, Positive direction torque limit value (AxisName.Cd.TorqueLimit_Positive) or Negative direction torque limit value (AxisName.Cd.TorqueLimit_Negative) was changed to a value smaller than the target torque.	The target torque is changed to the torque limit value. In this case, the set torque is changed in one operation cycle.

- While this FB is operating, the continuous operation to torque control mode cannot be switched by multiple starts of MC_TorqueControl (Torque Control). When the control mode is switched, Control mode switching disabled warning (event code: 00D31H) occurs and the operation is performed in the current control mode.
- This FB cannot be executed during the continuous operation to torque control. If the control mode is switched, Control mode switching error (error code: 1A95H) occurs.
- When the control mode is switched from cst using multiple starts of MCv_CyclicTorque (Motion Cyclic Torque Control), the command torque set in MCv_CyclicTorque (Motion Cyclic Torque Control) is immediately sent. Set a value so that the difference between the command torque and speed limit before and after multiple starts does not become large. If the difference is large, the sudden operation may occur.

Program example

For a program example of this FB, refer to the following.

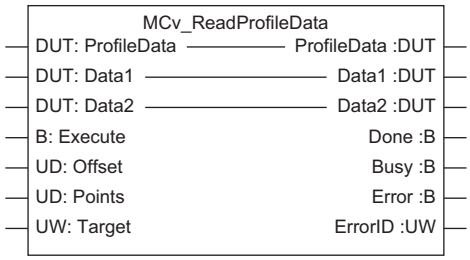
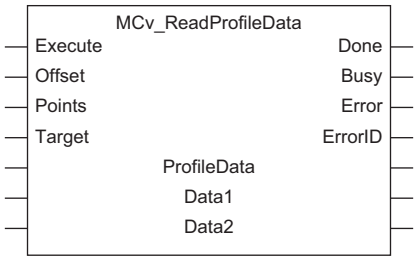
User's manual for the controller used

43 STANDARD FUNCTION BLOCK

43.1 Profile Read

MCv_ReadProfileData

This FB reads the specified operation profile data from the open area or a file.

Ladder	FBD/LD	ST
		MCv_ReadProfileData(ProfileData:= ?PROFILE_DATA? , Data1:= ?TARGET_REF? , Data2:= ?TARGET_REF? , Execute:= ?BOOL? , Offset:= ?DWORD? , Points:= ?DWORD? , Target:= ?WORD? , Done=> ?BOOL? , Busy=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
ProfileData	Profile data	PROFILE_DATA	At start	—	Omission not possible	The operation profile data to be read is set. The operation profile data is set with a PROFILE_DATA structure. For the PROFILE_DATA structure, refer to the following. ☞ Page 1447 PROFILE_DATA The operation profile data in which the read data structure has been defined can be read.
Data1	Read data 1	TARGET_REF	At start	—	Omission not possible	The read data structure suitable for the operation profile data format to be read is set. For details, refer to the following. ☞ Page 1916 Read data 1 (Data1)/Read data 2 (Data2)
Data2	Read data 2	TARGET_REF	At start	—	Omission not possible	The read data structure suitable for the operation profile data format to be read is set. For details, refer to the following. ☞ Page 1916 Read data 1 (Data1)/Read data 2 (Data2)

■Input variables

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Start	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_ReadProfileData (Profile Read) is executed.
Offset	Offset	DWORD(UDINT)	At start	0 to resolution (2 to 65535)	0	The offset from the start of the operation profile data is set. If Offset (Offset) exceeds the range of the operation profile data, Out of Offset Range (error code: 1B92H) occurs and reading will not be performed. When a value other than "0" is set, set a value other than "0" to Number of read data (Points). (When "0" is set to Number of read data (Points), Number of Offset/Read/Write Data Incorrect (error code: 1BADH) occurs.) *: The read operation differs depending on the set values set to Offset (Offset) and Number of read data (Points). For details, refer to the following. ☞ Page 1911 Offset (Offset) and Number of read data (Points) setting
Points	Number of read data	DWORD(UDINT)	At start	0 to 4294967295	0	The number of read data points is set. When Number of read data (Points) exceeds the number of elements in the operation profile data, the operation profile data for the number of elements is read. When Interpolation method specification (Interpolate) of the operation profile data is set to "1: Specifying for each section" or "2: Spline interpolation", set "0". (When a value other than "0" is set, Number of Offset/Read/Write Data Incorrect (error code: 1BADH) occurs.)
Target	Read destination	WORD(UINT)	At start	0, 1, 2	0	The read destination of the operation profile data is set. • 0: Open area • 1: File • 2: Open area (motion service processing) For details, refer to the following. ☞ Page 1918 Read destination (Target)

■Output variables

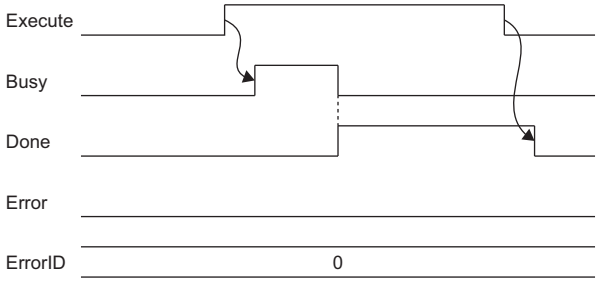
Output variable	Name	Data type	Default value	Description
Done	Completion	BOOL	FALSE	This variable becomes TRUE when the control is completed. The value differs depending on the status of Start (Execute) at operation completion. ■When Start (Execute) is TRUE This variable remains TRUE until Start (Execute) is set to FALSE. ■When Start (Execute) is FALSE This variable becomes TRUE only for one cycle.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_ReadProfileData (Profile Read) is executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used

Processing details

- This FB reads the operation profile data.
- The file specified in Profile data (ProfileData) and Read destination (Target), or the operation profile data in the open area are read for the number of points specified in Number of read data (Points) from the data specified with Offset (Offset).
- To read the whole operation profile data, specify "0" in both Offset (Offset) and Number of read data (Points).
- Read data is stored in variables specified in Read data 1 (Data1) and Read data 2 (Data2).

■Timing chart

- When the processing is completed successfully



- When the processing is completed with an error

  For the timing chart, refer to the following.

  ☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

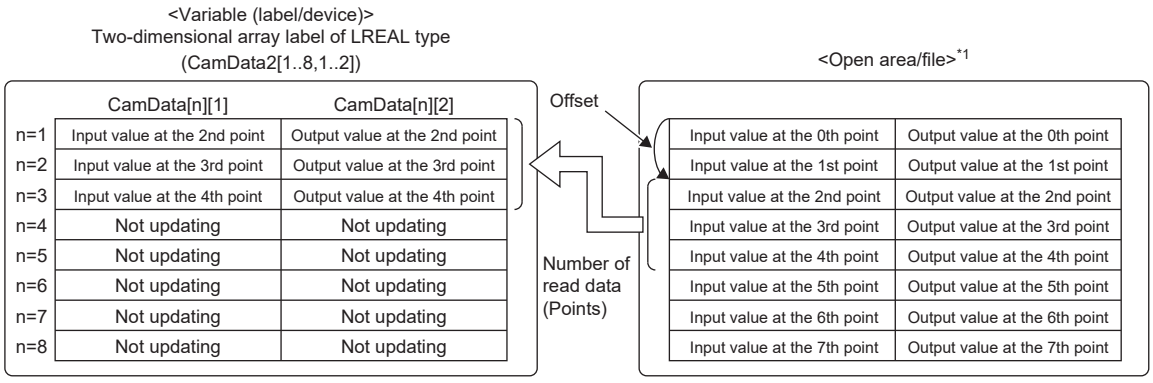
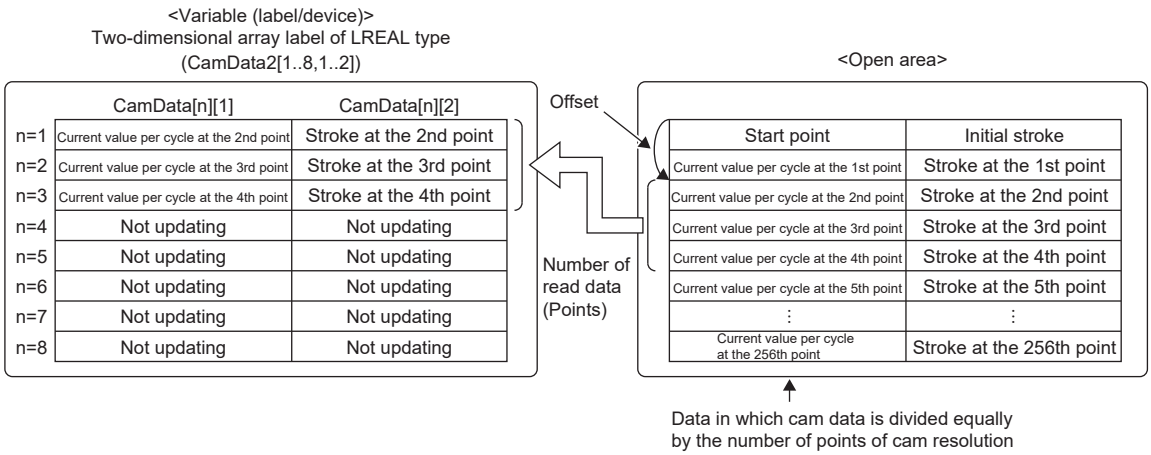
■Offset (Offset) and Number of read data (Points) setting

The following operations are based on the values set in Offset (Offset) and Number of read data (Points).

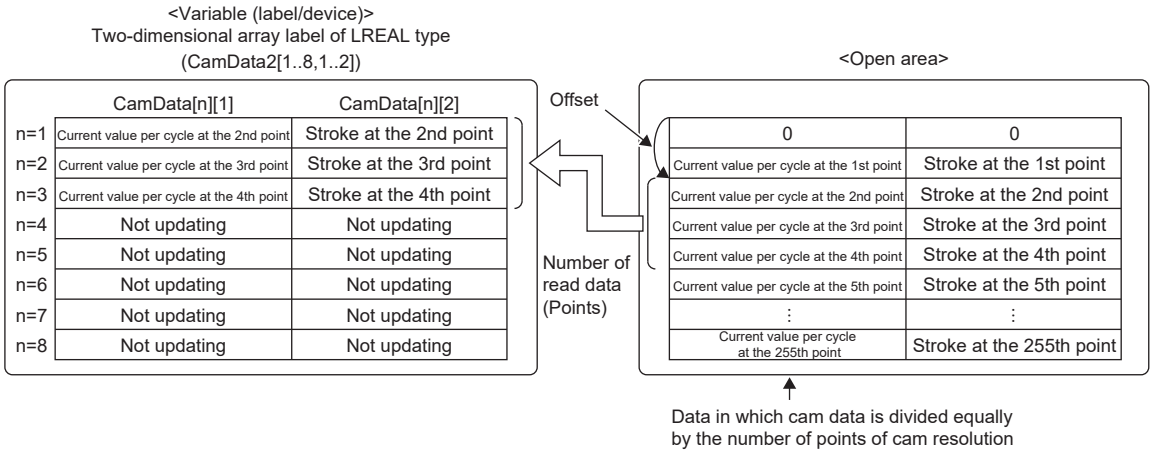
Offset (Offset)	Number of read data (Points)	Size relationship with the number of elements of the operation profile data	Operation
0	0	—	All operation profile data elements are read.
	Other than 0	Offset (Offset) + Number of read data (Points) ≤ number of elements of the operation profile data	The number of operation profile data elements set in Number of read data (Points) is read from the top.
	Other than 0	Offset (Offset) + Number of read data (Points) > number of elements of the operation profile data	All operation profile data elements are read.
Other than 0	0	Offset (Offset) ≤ number of elements of the operation profile data	Number of Offset/Read/Write Data Incorrect (error code: 1BADH) occurs, and read is not executed.
	Other than 0	Offset (Offset) + Number of read data (Points) ≤ number of elements of the operation profile data	The number of operation profile data elements specified in Number of read data (Points) from Offset (Offset) is read.
	Other than 0	Offset (Offset) + Number of read data (Points) > number of elements of the operation profile data	Reading is performed until the final element of the operation profile data from Offset (Offset).
	—	Offset (Offset) > number of elements of the operation profile data	Out of Offset Range (error code: 1B92H) occurs, and read is not executed.

The contents of the values set for Offset (Offset) and Number of read data (Points) vary depending on the operation profile data format.

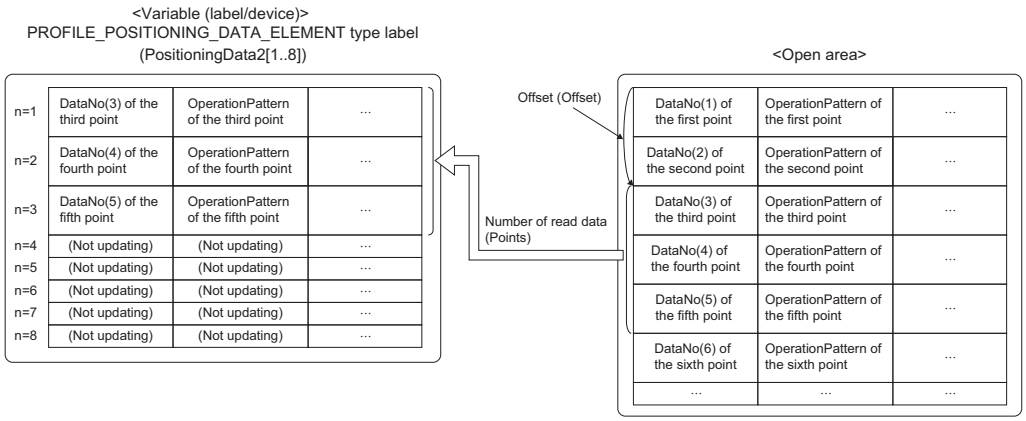
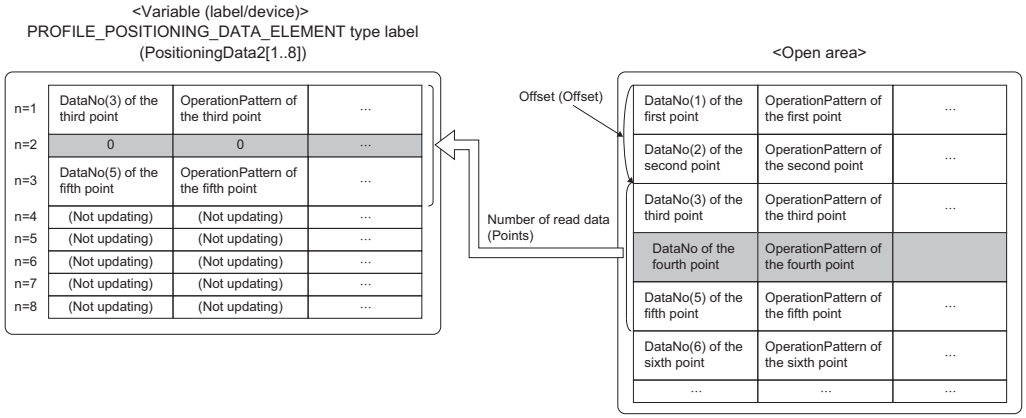
• When the operation profile data format is "cam data"

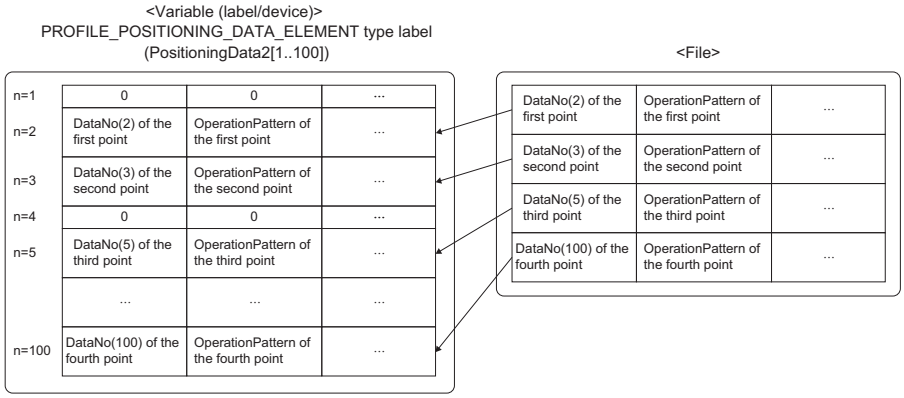
Specifying interpolation method (Interpolate)	Description
0: Linear interpolation	The top of the coordinate to be read in Offset (Offset) and the number of coordinates in Number of read data (Points) are set. The data stored in Read data 1 (Data1) is the PROFILE_CAM_DATA type data when Specifying interpolation method (Interpolate) is "0: Linear interpolation". The data stored in Read data 2 (Data2) is data defined by using set coordinate data (a pair of input value (current value per cycle) and output value (stroke)) in LREAL type two-dimensional array. Example: Read result of Read data 2 (Data2) when Offset (Offset) is set to "2" and Number of read data (Points) is set to "3". (When the variable set to Read data 2 (Data2) is a "LREAL[1..8,1..2]" data type "CamData2" label.)  *1: Depends on Read destination (Target).
1: Specifying for each section 2: Spline interpolation	The data that can be read differs depending on Read destination (Target). ■When Read destination (Target) is "0: Open area" or "2: Open area (motion service processing)" The top of the section to be read in Offset (Offset) and the Number of sections (NumberOfSections) in Number of read data (Points) are set. The data stored in Read data 1 (Data1) is the PROFILE_CAM_DATA type data when Specifying interpolation method (Interpolate) is "0: Linear interpolation". The data stored in Read data 2 (Data2) is data defined by using target cam data divided equally by the cam resolution points (a pair of current value per cycle and stroke) which is defined in LREAL type two-dimensional array. Example: Read result of Read data 2 (Data2) when Offset (Offset) is set to "2" and Number of read data (Points) is set to "3" and cam resolution is "256". (When the variable set to Read data 2 (Data2) is a "LREAL[1..8,1..2]" data type "CamData2" label.)  ■When Read destination (Target) is "1: File" Offset (Offset) and Number of read data (Points) are set to "0", and all elements are read. (When a value other than "0" is set, Number of Offset/Read/Write Data Incorrect (error code: 1BADH) occurs.) The data stored in Read data 1 (Data1) is the PROFILE_CAM_DATA type data when Specifying interpolation method (Interpolate) is "1: Specifying for each section" or "2: Spline Interpolation" The data stored in Read data 2 (Data2) is section data defined in PROFILE_CAM_ELEMENT type array.

• When the operation profile data format is "rotary cutter"

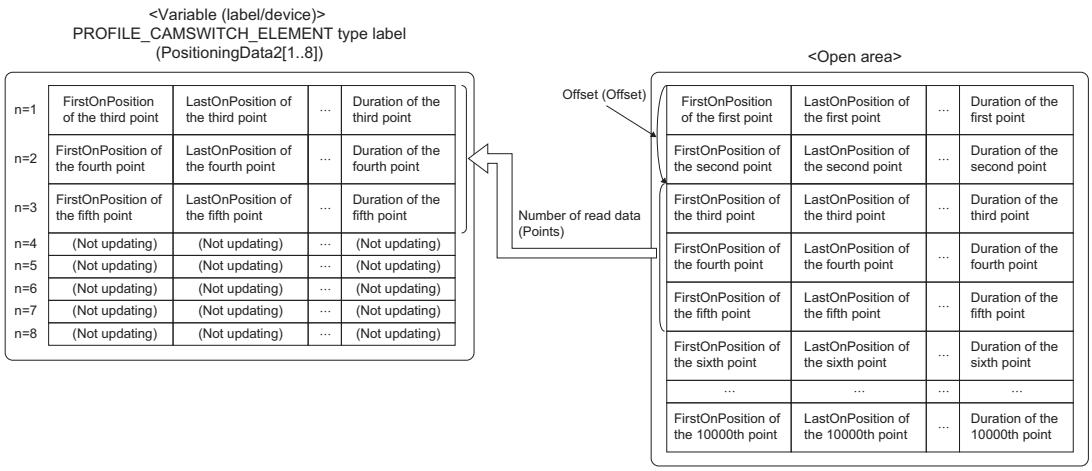
Read destination (Target)	Description
0: Open area 2: Open area (motion service processing)	The top of the section to be read in Offset (Offset) and the Number of sections (NumberOfSections) in Number of read data (Points) are set. The data stored in Read data 1 (Data1) is the PROFILE_CAM_DATA type data when Specifying interpolation method (Interpolate) is "0: Linear interpolation" and the operation profile data format is "cam data". The data stored in Read data 2 (Data2) is data defined by using target cam data divided equally by the cam resolution points (a pair of current value per cycle and stroke) which is defined in LREAL type two-dimensional array. Example: Read result of Read data 2 (Data2) when Offset (Offset) is set to "2" and Number of read data (Points) is set to "3". (When the variable set to Read data 2 (Data2) is a "LREAL[1..8,1..2]" data type "CamData2" label.) 
1: File	The setting for Offset (Offset) and Number of read data (Points) is ignored. The data stored in Read data 1 (Data1) is data where the operation profile data format is PROFILE_ROTARY_CUTTER type "rotary cutter". There is no data stored in Read data 2 (Data2).

• When the operation profile data format is "Multiple Axes Positioning Data"

Read destination (Target)	Description
0: Open area 2: Open area (motion service processing)	Specify the top of the section from which positioning data is to be read in Offset (Offset), and the Number of sections (NumberOfSections) in Number of read data (Points). Data is read to a PROFILE_POSITIONING_DATA type variable specified in Data1 (TARGET_REF type). When a PROFILE_POSITIONING_DATA_ELEMENT type array variable is specified in Data2 (TARGET_REF type), positioning data is read in the specified variable. Example: Read result of Read data 2 (Data2) when Offset (Offset) is set to "2" and Number of read data (Points) is set to "3". (When the variable set to Read data 2 (Data2) is a "PROFILE_POSITIONING_DATA_ELEMENT[1..8]" data type "PositioningData2" label)  When positioning data that has not been set for open or write is present in the read target section, the initial value is stored in the section data with the corresponding DataNo. (DataNo will be 0.) Example: When the fourth open area (positioning data number 4) in the above example has not been set for open or write  When a PROFILE_POSITIONING_DATA_EXTENDED type variable is specified in Data2 (TARGET_REF type) The condition signal, skip signal, and positioning data are stored in their corresponding variables specified as the WSTRING type in variable members ConditionSignal, SkipSignal, and PositioningData. For the condition signal and skip signal, PROFILE_SIGNAL_SELECT_ELEMENT type array data is read. For positioning data, PROFILE_POSITIONING_DATA_ELEMENT type array data is read. (This behavior is the same as specifying a PROFILE_POSITIONING_DATA_ELEMENT type array as Data2 (TARGET_REF type).)

Read destination (Target)	Description
1: File	Specify Offset (Offset) and Number of read data (Points) to 0 to read all elements. (When a value other than 0 is set, Number of Offset/Read/Write Data Incorrect (error code: 1BADH) occurs.) Data is read to a PROFILE_POSITIONING_DATA type variable specified in Data1 (TARGET_REF type). When a PROFILE_POSITIONING_DATA_ELEMENT type array variable is specified in Data2 (TARGET_REF type), positioning data is read in the specified variable. When an empty DataNo is prepared in the file and positioning data is set, the initial value is stored in the section data corresponding to the empty DataNo. (DataNo will be 0.) Example: Read result of Read data 2 (Data2) when Offset (Offset) is set to "0" and Number of read data (Points) is set to "0" (When the variable set to Read data 2 (Data2) is a "PROFILE_POSITIONING_DATA_ELEMENT[1..100]" data type "PositioningData2" label)  When a PROFILE_POSITIONING_DATA_EXTENDED type variable is specified in Data2 (TARGET_REF type) The condition signal, skip signal, and positioning data are stored in their corresponding variables specified as the WSTRING type in variable members ConditionSignal, SkipSignal, and PositioningData. For the condition signal and skip signal, PROFILE_SIGNAL_SELECT_ELEMENT type array data is read. For positioning data, PROFILE_POSITIONING_DATA_ELEMENT type array data is read. (This behavior is the same as specifying a PROFILE_POSITIONING_DATA_ELEMENT type array variable as Data2 (TARGET_REF type).)

• When the operation profile data format is "Digital Cam Switch Data"

Read destination (Target)	Description
0: Open area 2: Open area (motion service processing)	The top of the section to be read in Offset (Offset) and the Number of sections (NumberOfSections) in Number of read data (Points) are set. The data stored in Read data 1 (Data1) is the PROFILE_CAMSWITCH_DATA type data. The data stored in Read data 2 (Data2) is the data for which the data of the specified switch settings (FirstOnPosition, LastOnPosition, Direction, CamSwitchMode, and Duration) is defined as the PROFILE_CAMSWITCH_ELEMENT type array. Example: Read result of Read data 2 (Data2) when Offset (Offset) is set to "2" and Number of read data (Points) is set to "3". (When the variable set to Read data 2 (Data2) is a "PROFILE_CAMSWITCH_ELEMENT[1..8]" data type "DigitalCamSwitch Data2" label) 
1: File	Specify Offset (Offset) and Number of read data (Points) to 0 to read all elements. (When a value other than 0 is set, Number of Offset/Read/Write Data Incorrect (error code: 1BADH) occurs.) The data stored in Read data 1 (Data1) is the PROFILE_CAMSWITCH_DATA type data. The data stored in Read data 2 (Data2) is the data for which the data of the specified switch settings (FirstOnPosition, LastOnPosition, Direction, CamSwitchMode, and Duration) is defined as the PROFILE_CAMSWITCH_ELEMENT type array.

■Read data 1 (Data1)/Read data 2 (Data2)

Set read data of the operation profile data with a TARGET_REF structure. The [VAR] and [DEV] data types can be used.

For details on the TARGET_REF structure, refer to the following.

☞ Page 1444 TARGET_REF (Input signal)

For data types, read data structures of each operation profile data format can be used.

• Read data 1 (Data1)

The operation profile data formats and data types set in Read data 1 (Data1) are shown below.

Operation profile data format	Data type	Reference
Cam data	PROFILE_CAM_DATA	☞ Page 1449 PROFILE_CAM_DATA
Cam data for rotary cutter	PROFILE_ROTARY_CUTTER	☞ Page 1451 PROFILE_ROTARY_CUTTER
Multiple axes positioning data	PROFILE_POSITIONING_DATA	☞ Page 1454 PROFILE_POSITIONING_DATA
Digital cam switch data	PROFILE_CAMSWITCH_DATA	☞ Page 1453 PROFILE_CAMSWITCH_DATA

Ex.

When reading a PROFILE_CAM_DATA type label (CamData1) using a device or label, describe as follows:

Data1.Target := "[VAR]CamData1"

• Read data 2 (Data2)

The operation profile data formats and data types set in Read data 2 (Data2) are shown below. The data type of the structure that is set varies depending on Read destination (Target) and Specifying interpolation method (Interpolate) of the target cam data.

If the number of elements of the specified array is less than (the resolution + 1/number of coordinates) when the entire operation profile data is read, Number of Read/Write Data Mismatch (error code: 1BACH) occurs.

○: Supported, ×: Not supported

Operation profile data format	Read destination (Target)	Data type	Specifying interpolation method (Interpolate)			Reference
			0: Linear interpolation	1: Specifying for each section	2: Spline interpolation	
Cam data	0: Open area 2: Open area (motion service processing)	LREAL[ ]*1	○	○*3	○*3	—
	1: File	LREAL[ ]*1	○	×	×	—
		PROFILE_CAM_ELEMENT[ ]*2	×	○	○*4	☞ Page 1450 PROFILE_CAM_ELEMENT
Cam data for rotary cutter	Setting is unnecessary. If set, the setting is ignored.					—
Multiple axes positioning data	—	PROFILE_POSITIONING_DATA_ELEMENT[ ]*5	—			☞ Page 1454 PROFILE_POSITIONING_DATA
Digital cam switch data	—	PROFILE_CAMSWITCH_ELEMENT*6	—			☞ Page 1453 PROFILE_CAMSWITCH_ELEMENT

*1 The LREAL type two-dimensional array is set as follows.
LREAL[m..n,o..p]

Item	Description
Number of elements (n-m+1)	Set the array elements so that they are equal to or greater than the number of points that are to be read. When reading the entire operation profile data (Offset (Offset), Number of read data (Points) are set to "0"), set this to the cam data resolution +1/the number of coordinates or more. However, an area bigger than the resolution +1/number of coordinates in Read data 1 (Data1) will not be updated.
Number of dimensions (p-o+1)	The stroke and current position per cycle for each point are read. Set the array element to "2". If the array element is set to "3" or more, a read is possible, but as the intended result of the read may not be obtained due to data being moved forward, make sure to set the array element to "2".

*2 The PROFILE_CAM ELEMENT array is set as follows.
PROFILE_CAM_ELEMENT[m..n]

Item	Description
Number of elements (n-m+1)	Set the array elements so that they are equal to or greater than the number of points that are to be read. When reading the entire operation profile data (Offset (Offset), Number of read data (Points) are set to "0"), set this to the cam data Number of sections (NumberOfSections) or more. However, an area bigger than the Number of sections (NumberOfSections) in Read data 1 (Data1) will not be updated.

*3 Read as stroke data (a pair of current value per cycle and stroke) opened to the open area. X of 1st point data is the value set in start point, and Y of 1st point data is the value set in initial stroke amount.
Data after the 2nd point is stroke data divided by the resolution. Due to this, the read points in the open area is "resolution+1".

*4 Only used by the end point and stroke.

*5 Set the PROFILE_POSITIONING_DATA_ELEMENT[m..n] array as follows.

Item	Description
Number of elements (n-m+1)	Set the array elements so that they are equal to or greater than the number of points that are to be read. When reading the entire operation profile data (Offset (Offset) and Number of read data (Points) are set to "0"), ensure that the number of array elements is set to the number of positioning data settings (PROFILE_POSITIONING_DATA.TotalNumberOfData) or more. If the number of array elements is less, "Number of read/write data mismatch (error code: 1BACH)" will occur.

*6 Set the PROFILE_CAMSWITCH_ELEMENT[m..n] array as follows.

Item	Description
Number of elements (n-m+1)	Set a positive number so that the array elements are equal to or greater than the number of points to be read. When reading the entire operation profile data (Offset (Offset) and Number of read data (Points) are set to "0"), ensure that the value is set to the number of switches (PROFILE_CAMSWITCH_DATA.NumberOfSwitches) of digital cam switch data or more. However, an area larger than the number of switches read by Data1 is not updated.

Ex.

When setting LREAL type with Specifying interpolation method (Interpolate)

Specifying interpolation method (Interpolate)	Setting example
0: Linear interpolation	The following shows examples of setting the two-dimensional label coordinate number "100" to Target (Target) of the TARGET_REF structure. Data2.Target := "[VAR]CamData2" *: CamData2 declares the "LREAL[1..100,1..2]" label to data type and label name of the global label to "CamData2", and specifies the string for Target (Target). The read data is as shown below. · CamData2[n,1]: nth point current position per cycle · CamData2[n,2]: nth point stroke
1: Specifying for each section 2: Spline interpolation	The following shows examples of setting the two-dimensional label resolution "100" to Target (Target) of the TARGET_REF structure. TARGET_REF.Target := "[VAR]CamData2" *: CamData2 declares the "LREAL[1..101,1..2]" label to data type and label name of the global label to "CamData2", and specifies the string for Target (Target). The read data is as shown below. · CamData2[1,1]: Value set for the start point · CamData2[1,2]: Value set for the initial stroke amount · CamData2[n+1,1]: nth point current position per cycle · CamData2[n+1,2]: nth point stroke

Ex.

For PROFILE_CAM_ELEMENT type

Setting example
The following shows examples of setting the two-dimensional label number of sections "10" to Target (Target) of the TARGET_REF structure. Data2.Target := "[VAR]CamData2" *: CamData2 declares the "PROFILE_CAM_ELEMENT[1..10]" label to data type and label name of the global label to "CamData2", and specifies the string for Target (Target). The read data is as shown below. · CamData2[n]: nth point section data (PROFILE_CAM_ELEMENT type)

Point

- If the type of the specified data structure for reading does not match the type that is required for each operation profile data, Read/Write Data Type Incorrect (error code: 1BAAH) occurs.
- For details on the cam data for the operation profile data set in Read data 1 (Data1)/Read data 2 (Data2), refer to the following.

User's manual for the controller used

- Using [VAR] may cause the read processing to take longer. Reading with [DEV] will be faster than with [VAR]. Therefore, use [DEV] to reduce read processing time.

■Read destination (Target)

The read destination of the operation profile data is set.

Setting value	Description
0: Open area	• The Profile ID (ID) value of the PROFILE_DATA structure is referred to, and the read is performed for the operation profile data that Profile ID No. (Number) of the PROFILE_ID structure indicates in the open area. • The read is performed at the cycle of the executed program. • If the operation profile data does not exist (unopened), No Operation Profile Data (error code: 1B8EH) occurs.
1: File	• The Operation profile data storage location (Location) value of the PROFILE_DATA structure is referred, and the read is performed for the operation profile data that the file name and the path indicate. • The read is performed at the cycle of the motion service processing. • If the operation profile data does not exist, No Operation Profile Data (error code: 1B8EH) occurs.
2: Open area (motion service processing)	• The Profile ID (ID) value of the PROFILE_DATA structure is referred to, and the read is performed for the operation profile data that Profile ID No. (Number) of the PROFILE_ID structure indicates in the open area. • The read is performed at the cycle of the motion service processing. • If the operation profile data does not exist (unopened), No Operation Profile Data (error code: 1B8EH) occurs.

Point

- When Read destination (Target) is "1: File" or "2: Open area (motion service processing)", as motion service processing is used in that case, it may take much time for the processing to complete. Use Done (Execution completion) for the interlock as needed.
- When Target is set to "0: Open area", the processing is completed immediately as it is performed within the task where the FB is called. The program execution time increases when the number of read data points is too large. Therefore, execute the read multiple times with smaller data quantities as needed, or use the setting with Target set to "2: Open area (motion service processing)".

Precautions

- The number of data points that can be read at the same time are limited. If the read is not completed in a single FB execution, execute the read over several FB executions.

Program example

The following is a program example. "LREAL type (two-dimensional array) format" cam data 1 (ProfileData0001) and "PROFILE_CAM_ELEMENT format" cam data 2 (ProfileData0002) are read from the file.

■Operation profile data

Profile ID	Label name	Data type	Comment
1	ProfileData0001	MC_CAM_REF	Cam data 1
2	ProfileData0002	MC_CAM_REF	Cam data 2

• Operation profile data (cam data) settings

Item		ProfileData0001	ProfileData0002
Specifying interpolation method		Linear interpolation	Specifying for each section
Resolution/number of coordinates		5	256
Length per cycle setting	Unit	degree	degree
	Length per cycle	360.00000	360.00000
Stroke amount	Unit	pulse	%
	Stroke amount	100.0	100.0000000
Cam 1 cycle time		—	1.000 [s]
Initial stroke value		—	0.0000000
Cam 1 stroke minimum value		0.00000	—
Cam 1 stroke maximum value		360.00000	—

• ProfileData0001

Point No.	Input value [degree]	Output value [pulse]
1	0.00000	0.0
2	90.00000	60.0
3	180.00000	100.0
4	270.00000	30.0
5	360.00000	0.0

• ProfileData0002

Section No.	Start point [degree]	End point [degree]	Stroke [%]	Cam curve type
1	0.00000	90.00000	60.0000000	Single hypotenuse
2	90.00000	180.00000	100.0000000	Single hypotenuse
3	180.00000	270.00000	30.0000000	Single hypotenuse
4	270.00000	0.00000	0.0000000	Single hypotenuse

■Labels used

Label name	Data type	Comment
MCv_ReadProfileData_1	MCv_ReadProfileData	Profile Read FB 1
ReadData1_1	TARGET_REF	Read data 1 Profile 1
ReadData2_1	TARGET_REF	Read data 2 Profile 1
bReadProfile1	Bit	Profile read command 1
bDone1	Bit	Completion 1
bBusy1	Bit	Executing 1
bError1	Bit	Error 1
uwErrorID1	Word [unsigned]/bit string [16-bit]	Error code 1
MCv_ReadProfileData_2	MCv_ReadProfileData	Profile Read FB 2
ReadData1_2	TARGET_REF	Read data 1 Profile 2
ReadData2_2	TARGET_REF	Read data 2 Profile 2
bReadProfile2	Bit	Profile read command 2
bDone2	Bit	Completion 2
bBusy2	Bit	Executing 2
bError2	Bit	Error 2
uwErrorID2	Word [unsigned]/bit string [16-bit]	Error code 2

■Global labels used

Label name	Data type	Comment
G_ReadData1_Lreal*1	PROFILE_CAM_DATA	Profile data Cam data 1
G_ReadData2_Lreal*1	Double-precision real number (0..4,0..1)	Input value/Output value data LREAL type
G_ReadData1_Element*1	PROFILE_CAM_DATA	Profile data Cam data 2
G_ReadData2_Element*1	PROFILE_CAM_ELEMENT (0..3)	Input value/Output value data PROFILE_CAM_ELEMENT type

*1 Select "Access from External Device" for global labels to be used.

■ST program

```
//-----Read data 1-----
ReadData1_1.Target := "[VAR]G_ReadData1_Lreal";
ReadData2_1.Target := "[VAR]G_ReadData2_Lreal";

//-----Profile Read 1-----
//-----Reads cam data in the LREAL type (two-dimensional array) format)-----
MCv_ReadProfileData_1(
   ProfileData:= ProfileData0001.ProfileData ,
   Data1:= ReadData1_1 ,
   Data2:= ReadData2_1 ,
   Execute:= bReadProfile1 ,
   Offset:= 0 ,
   Points:= 0 ,
   Target:= 1 ,
   Done=> bDone1 ,
   Busy=> bBusy1 ,
   Error=> bError1 ,
   ErrorID=> uwErrorID1
);

//-----Read Data 2-----
ReadData1_2.Target := "[VAR]G_ReadData1_Element";
ReadData2_2.Target := "[VAR]G_ReadData2_Element";

//-----Profile Read 2-----
//-----Reads cam data in the PROFILE_CAM_ELEMENT type format-----
MCv_ReadProfileData_2(
   ProfileData:= ProfileData0002.ProfileData ,
   Data1:= ReadData1_2 ,
   Data2:= ReadData2_2 ,
   Execute:= bReadProfile2 ,
   Offset:= 0 ,
   Points:= 0 ,
   Target:= 1 ,
   Done=> bDone2 ,
   Busy=> bBusy2 ,
   Error=> bError2 ,
   ErrorID=> uwErrorID2
);
```

Point

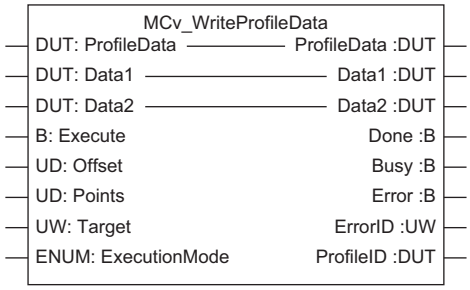
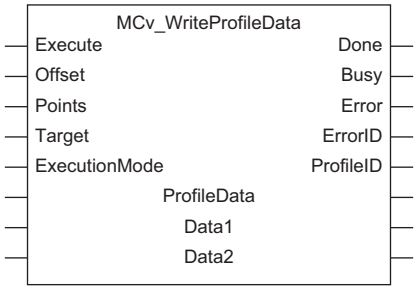
For program examples in which the operation profile data format is "multiple axes positioning data" or "digital cam switch data", refer to the program examples of MCv_WriteProfileData (Profile Write).

☞ Page 1922 MCv_WriteProfileData

43.2 Profile Write

MCv_WriteProfileData

This FB writes the specified operation profile data to the open area or a file.

Ladder	FBD/LD	ST
		MCv_WriteProfileData(ProfileData:= ?PROFILE_DATA? , Data1:= ?TARGET_REF? , Data2:= ?TARGET_REF? , Execute:= ?BOOL? , Offset:= ?DWORD? , Points:= ?DWORD? , Target:= ?WORD? , ExecutionMode:= ?INT? , Done=> ?BOOL? , Busy=> ?BOOL? , Error=> ?BOOL? , ErrorID=> ?WORD? , ProfileID=> ?PROFILE_ID?);

Setting data

■I/O variables

I/O variable	Name	Data type	Input	Setting range	Default value	Description
ProfileData	Profile data	PROFILE_DATA	At start	—	Omission not possible	The operation profile data to be written is set. The operation profile data is set with a PROFILE_DATA structure. For the PROFILE_DATA structure, refer to the following. ☞ Page 1447 PROFILE_DATA The operation profile data in which the write data structure has been defined can be written.
Data1	Write data 1	TARGET_REF	At start	—	Omission not possible	The write data structure suitable for the operation profile data format to be written is set. For details, refer to the following. ☞ Page 1928 Write data 1 (Data1)/Write data 2 (Data2)
Data2	Write data 2	TARGET_REF	At start	—	Omission not possible	The write data structure suitable for the operation profile data format to be written is set. For details, refer to the following. ☞ Page 1928 Write data 1 (Data1)/Write data 2 (Data2)

■Input variables

Input variable	Name	Data type	Input	Setting range	Default value	Description
Execute	Start	BOOL	At start	TRUE, FALSE	FALSE	When this variable is set to TRUE, MCv_WriteProfileData (Profile Write) is executed.
Offset	Offset	DWORD(UDINT)	At start	0 to resolution (2 to 65535)	0	The offset from the start of the operation profile data is set. If Offset (Offset) exceeds the range of the operation profile data, Out of Offset Range (error code: 1B92H) occurs and the data will not be written. When a value other than "0" is set, set a value other than "0" to Number of write data (Points). (If "0" is set to Number of write data (Points), Number of Offset/Read/Write Data Incorrect (error code: 1BADH) occurs.) *: The write operation differs depending on the values set to Offset (Offset) and Number of write data (Points). For details, refer to the following. ☞ Page 1924 Offset (Offset) and Number of write data (Points) setting

Input variable	Name	Data type	Input	Setting range	Default value	Description
Points	Number of write data	DWORD(UDINT)	At start	0 to 4294967295	0	The number of write data points is set. When Number of write data (Points) exceeds the number of elements in the operation profile data, the operation profile data for the number of elements is written. When "1: File" is set to Write destination (Target), set "0". (When a value other than "0" is set, Number of Offset/Read/Write Data Incorrect (error code: 1BADH) occurs.)
Target	Write destination	WORD(UINT)	At start	0, 1, 2	0	The write destination of the operation profile data is set. • 0: Open area • 1: File • 2: Open area (motion service processing) *: When "1: File" is set for Write destination (Target), set "0" to Offset (Offset) and Number of write data (Points). (When a value other than "0" is set, Number of Offset/Read/Write Data Incorrect (error code: 1BADH) occurs.) For details, refer to the following. ☞ Page 1931 Write destination (Target)
ExecutionMode	Execution mode	INT (MC_EXECUTION_MODE)	At start	0, 1, 3	0	The execution timing of MCv_WriteProfileData (Profile Write) is set. • 0: Execute immediately (mcImmediately) • 1: Execute at completion (mcQueued) • 3: Execute speculatively (mcSpeculatively) For details, refer to the following. ☞ Page 1932 Execution mode (ExecutionMode)

■Output variables

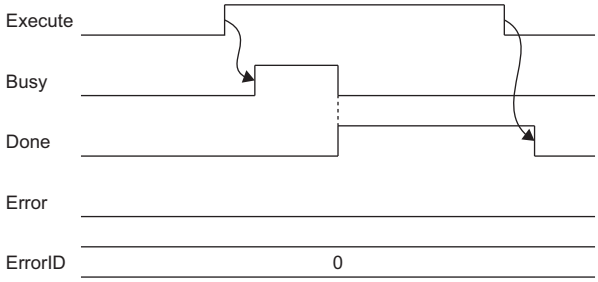
Output variable	Name	Data type	Default value	Description
Done	Completion	BOOL	FALSE	This variable becomes TRUE when the control is completed. The value differs depending on the status of Start (Execute) at operation completion. ■When Start (Execute) is TRUE This variable remains TRUE until Start (Execute) is set to FALSE. ■When Start (Execute) is FALSE This variable becomes TRUE only for one cycle.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_WriteProfileData (Profile Write) is executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details on error codes, refer to the following. 📖 User's manual for the controller used
ProfileID	Profile ID	PROFILE_ID	0	The profile ID is output.

Processing details

- This FB writes the operation profile data.
- The file specified in Profile data (ProfileData) and Write destination (Target), or the operation profile data in the open area are written for the number of points specified in Number of write data (Points) from the data specified with Offset (Offset).
- To write the whole operation profile data, specify "0" in both Offset (Offset) and Number of write data (Points).
- Write data is stored in variables specified in Write data 1 (Data1) and Write data 2 (Data2).

■Timing chart

- When the processing is completed successfully



- When the processing is completed with an error

  For the timing chart, refer to the following.

  ☞ Page 1388 Basic operation of Execution command (Execute) type motion control FBs

■Offset (Offset) and Number of write data (Points) setting

The following operations are based on the values set in Offset (Offset) and Number of write data (Points).

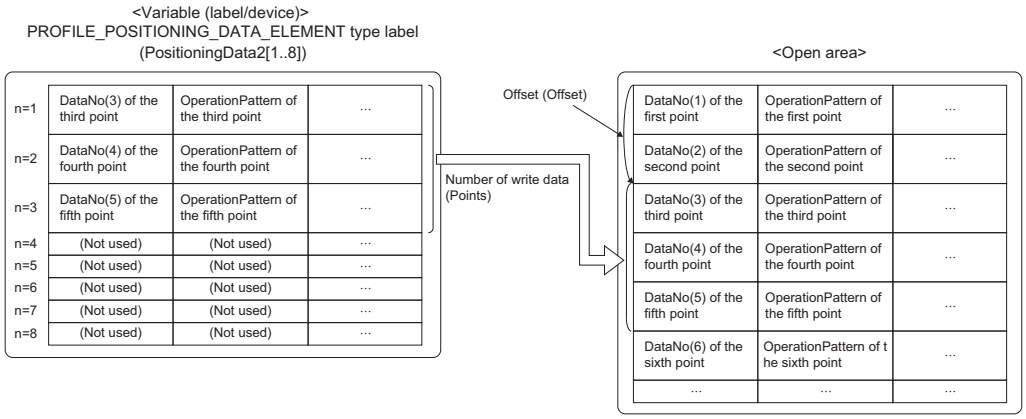
The contents of the values set for Offset (Offset) and Number of write data (Points) vary depending on the operation profile data format.

- When the operation profile data format is "cam data"

Specifying interpolation method (Interpolate)	Description
0: Linear interpolation	Set the top of the coordinate for write in Offset (Offset) and the number of coordinates in Number of write data (Points). The data stored in Write data 1 (Data1) is the PROFILE_CAM_DATA type data when Specifying interpolation method (Interpolate) is "0: Linear interpolation". The data stored in Write data 2 (Data2) is data defined by using set coordinate data (a pair of input value (current value per cycle) and output value (stroke)) in LREAL type two-dimensional array. Example: Write result of Write data 2 (Data2) when Offset (Offset) is set to "2" and Number of write data (Points) is set to "3". (When the variable set to Write data 2 (Data2) is a "LREAL[1..8,1..2]" data type "CamData2" label.) <Variable (label/device)> Two-dimensional array label of LREAL type (CamData2[1..8,1..2]) <Open area/file>*1 CamData[n][1] / CamData[n][2] n=1: Input value at the 2nd point / Output value at the 2nd point n=2: Input value at the 3rd point / Output value at the 3rd point n=3: Input value at the 4th point / Output value at the 4th point n=4 to n=8: Not updating / Not updating Offset Number of write data (Points) Input value at the 0th point / Output value at the 0th point Input value at the 1st point / Output value at the 1st point Input value at the 2nd point / Output value at the 2nd point Input value at the 3rd point / Output value at the 3rd point Input value at the 4th point / Output value at the 4th point Input value at the 5th point / Output value at the 5th point Input value at the 6th point / Output value at the 6th point Input value at the 7th point / Output value at the 7th point *1: Depends on Read destination (Target).
1: Specifying for each section	Specifying a section for write is not possible. Offset (Offset) and Number of write data (Points) are set to "0", and writing is performed for the entire operation profile data. When a value other than "0" is set, Number of Offset/Read/Write Data Incorrect (error code: 1BADH) occurs. The data stored in Write data 1 (Data1) is the PROFILE_CAM_DATA type data when Specifying interpolation method (Interpolate) is "1: Specifying for each section" or "2: Spline interpolation". The data stored in Write data 2 (Data2) is section data defined in PROFILE_CAM_ELEMENT type array.
2: Spline interpolation	

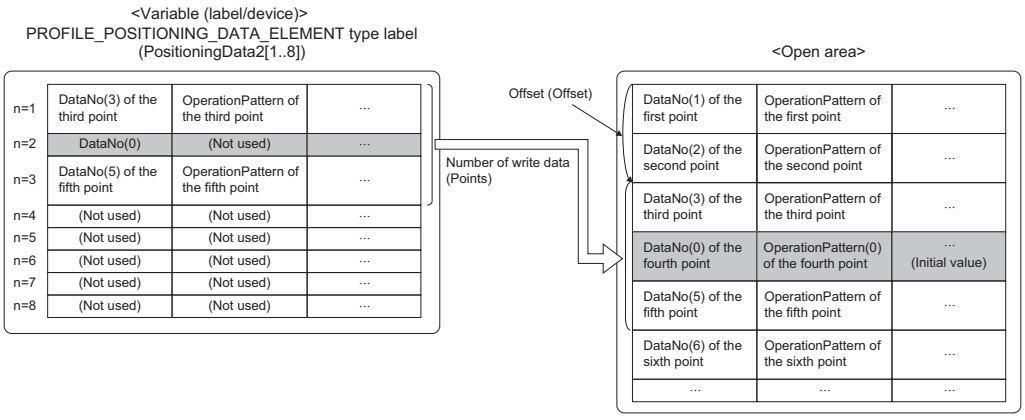
- When the operation profile data format is "rotary cutter"
  Specifying a section for write is not possible. This format sets the Offset (Offset) and Number of write data (Points) to "0", and writes the entire operation profile data. The setting input manually for Offset (Offset) and the Number of write data (Points) is ignored. The data stored in Write data 1 (Data1) is data where the operation profile data format is PROFILE_ROTARY_CUTTER type "rotary cutter". Write data 2 (Data2) is not used.
- When the operation profile data format is "Multiple Axes Positioning Data"

Write destination (Target)	Description
0: Open area 2: Open area (motion service processing)	Specify the top of the section to which positioning data is written in Offset (Offset), and the Number of sections (NumberOfSections) in Number of write data (Points). Data is written to a PROFILE_POSITIONING_DATA type variable specified in Data1 (TARGET_REF type). When a PROFILE_POSITIONING_DATA_ELEMENT type array variable is specified in Data2 (TARGET_REF type), positioning data is written from the specified variable. Example: Write result of Write data 2 (Data2) when Offset (Offset) is set to "2" and Number of write data (Points) is set to "3". (When the variable set to Write data 2 (Data2) is a "PROFILE_POSITIONING_DATA_ELEMENT[1..8]" data type "PositioningData2" label)

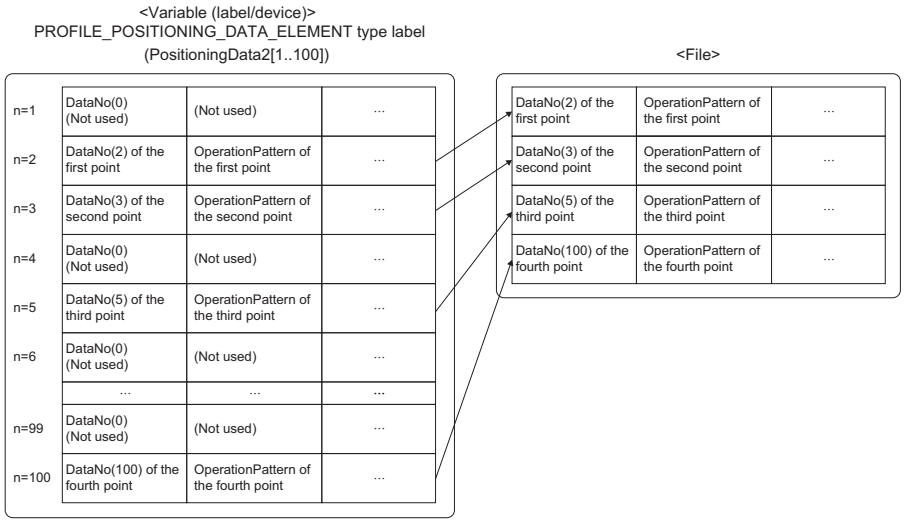


Specify DataNo of each positioning data in the specified variable so that it matches the write destination DataNo.
- When there is positioning data whose DataNo is "0", it is considered as unset, and the initial value is stored in the section data with the corresponding DataNo. (DataNo will be 0.)

Example:
When DataNo with variable n = 2 is set to 0 in the above example



If there is positioning data whose DataNo is "0" or does not match any of the write destination DataNo, Number of Offset/Read/Write Data Incorrect (error code: 1BADH) occurs.
When a PROFILE_POSITIONING_DATA_EXTENDED type variable is specified in Data2 (TARGET_REF type)
The condition signal, skip signal, and positioning data are written from their corresponding variables specified as the WSTRING type in variable members ConditionSignal, SkipSignal, and PositioningData.
For the condition signal and skip signal, specify the PROFILE_SIGNAL_SELECT_ELEMENT type array data.
For positioning data, specify the PROFILE_POSITIONING_DATA_ELEMENT type array data. (This behavior is the same as specifying a PROFILE_POSITIONING_DATA_ELEMENT type array variable as Data2 (TARGET_REF type).)

Write destination (Target)	Description
1: File	Specify Offset (Offset) and Number of write data (Points) to 0, and write the data to the entire operation profile data. (When a value other than 0 is set, Number of Offset/Read/Write Data Incorrect (error code: 1BADH) occurs.) The PROFILE_POSITIONING_DATA type data is written from the variable specified in Data1 (TARGET_REF type). When a PROFILE_POSITIONING_DATA_ELEMENT type array variable is specified in Data2 (TARGET_REF type), positioning data is written from the specified variable. When there is positioning data whose positioning data number is "0" is present in the write destination section, the section data with the corresponding DataNo is not written. Example: Write result of Write data 2 (Data2) when Offset (Offset) is set to "0" and Number of write data (Points) is set to "0" (When the variable set to Write data 2 (Data2) is a "PROFILE_POSITIONING_DATA_ELEMENT[1..100]" data type "PositioningData2" label)  When a PROFILE_POSITIONING_DATA_EXTENDED type variable is specified in Data2 (TARGET_REF type) The condition signal, skip signal, and positioning data are written from their corresponding variables specified as the WSTRING type in variable members ConditionSignal, SkipSignal, and PositioningData. For the condition signal and skip signal, specify the PROFILE_SIGNAL_SELECT_ELEMENT type array data. For positioning data, specify the PROFILE_POSITIONING_DATA_ELEMENT type array data. (This behavior is the same as specifying a PROFILE_POSITIONING_DATA_ELEMENT type array variable as Data2 (TARGET_REF type).)

• When the operation profile data format is "Digital Cam Switch Data"

The top of the section to be written is set in Offset (Offset), and the Number of sections (NumberOfSections) in Number of write data (Points).

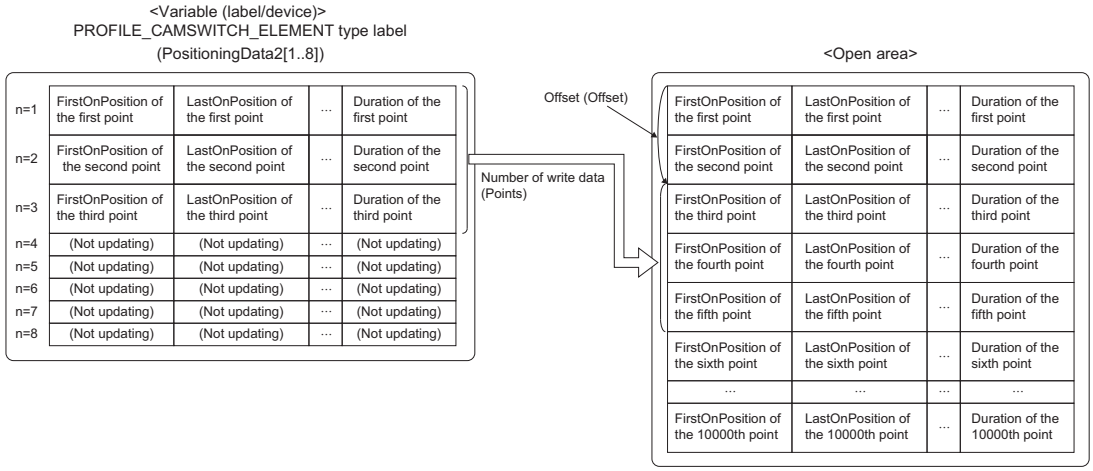
Specify PROFILE_CAMSWITCH_DATA type data in Write data 1 (Data1).

Specify the data for which the data of switch settings (FirstOnPosition, LastOnPosition, Direction, CamSwitchMode, and Duration) is defined as the PROFILE_CAMSWITCH_ELEMENT type array in Write data 2 (Data2).

Example:

Write result of Write data 2 (Data2) when Offset (Offset) is set to "0" and Number of write data (Points) is set to "3"

(When the variable set to Write data 2 (Data2) is a "PROFILE_CAMSWITCH_ELEMENT[1..8]" data type "DigitalCamSwitchData2" label)



■Write data 1 (Data1)/Write data 2 (Data2)

Set write data of the operation profile data with a TARGET_REF structure. The [VAR] and [DEV] data types can be used.

For details on the TARGET_REF structure, refer to the following.

☞ Page 1444 TARGET_REF (Input signal)

For data types, write data structures of each operation profile data format can be used.

• Write data 1 (Data1)

The operation profile data formats and data types set in Write data 1 (Data1) are shown below.

Operation profile data format	Data type	Reference
Cam data	PROFILE_CAM_DATA	☞ Page 1449 PROFILE_CAM_DATA
Cam data for rotary cutter	PROFILE_ROTARY_CUTTER	☞ Page 1451 PROFILE_ROTARY_CUTTER
Multiple axes positioning data	PROFILE_POSITIONING_DATA	☞ Page 1454 PROFILE_POSITIONING_DATA
Digital cam switch data	PROFILE_CAMSWITCH_DATA	☞ Page 1453 PROFILE_CAMSWITCH_DATA

Ex.

When writing a PROFILE_CAM_DATA type label (CamData1) using a device or label, describe as follows.

```
Data1.Target := "[VAR]CamData1"
```

• Write data 2 (Data2)

The operation profile data formats and data types set in Write data 2 (Data2) are shown below. The data type of the structure that is set varies depending on the Write destination (Target) and Specifying interpolation method (Interpolate) of the target cam data.

If the number of elements of the specified array is less than (the resolution + 1/number of coordinates) when the entire operation profile data is written, Number of Read/Write Data Mismatch (error code: 1BACH) occurs.

○: Supported, ×: Not supported

Operation profile data format	Write destination (Target)	Data type	Specifying interpolation method (Interpolate)			Reference
			0: Linear interpolation	1: Specifying for each section	2: Spline interpolation	
Cam data	0: Open area 2: Open area (motion service processing)	LREAL[ ]*1	○	×	×	—
		PROFILE_CAM_ELEMENT[ ]*2	×	○	○*3	☞ Page 1450 PROFILE_CAM_ELEMENT
	1: File	LREAL[ ]*1	○	×	×	—
		PROFILE_CAM_ELEMENT[ ]*2	×	○	○*3	☞ Page 1450 PROFILE_CAM_ELEMENT
Cam data for rotary cutter	Setting is unnecessary. If set, the setting is ignored.					—
Multiple axes positioning data	—	PROFILE_POSITIONING_DATA_ELEMENT[ ]*4	—			☞ Page 1454 PROFILE_POSITIONING_DATA
Digital cam switch data	—	PROFILE_CAMSWITCH_ELEMENT*5	—			☞ Page 1453 PROFILE_CAMSWITCH_ELEMENT

*1 The LREAL type two-dimensional array is set as follows.
LREAL[m..n,o..p]

Item	Description
Number of elements (n-m+1)	Set the array elements so that they are equal to or greater than the number of points that are to be read. When writing the entire operation profile data (Offset (Offset), Number of write data (Points) are set to "0"), set this to the cam data resolution +1/number of coordinates or more. However, an area bigger than the resolution +1/number of coordinates in Write data 1 (Data1) will not be updated.
Number of dimensions (p-o+1)	The stroke and current position per cycle for each point are written. Set the array element to "2". If the array element is set to "3" or more, a write is possible, but as the intended result of the write may not be obtained due to data being moved forward, make sure to set the array element to "2".

*2 The PROFILE_CAM ELEMENT array is set as follows.
PROFILE_CAM_ELEMENT[m..n]

Item	Description
Number of elements (n-m+1)	Set the array elements so that they are equal to or greater than the number of points that are to be read. When reading the entire operation profile data (Offset (Offset), Number of read data (Points) are set to "0"), set this to the cam data Number of sections (NumberOfSections) or more. However, an area bigger than the Number of sections (NumberOfSections) in Write data 1 (Data1) will not be updated.

*3 Only used by the end point and stroke.

*4 Set the PROFILE_POSITIONING_DATA_ELEMENT[m..n] array as follows.

Item	Description
Number of elements (n-m+1)	Set the array elements so that they are equal to or greater than the number of points that are to be read. When reading the entire operation profile data (Offset (Offset) and Number of read data (Points) are set to "0"), ensure that the number of array elements is set to the number of positioning data settings (PROFILE_POSITIONING_DATA.TotalNumberOfData) or more. If the number of array elements is less, "Number of read/write data mismatch (error code: 1BACH)" will occur.

*5 Set the PROFILE_CAMSWITCH_ELEMENT[m..n] array as follows.

Item	Description
Number of elements (n-m+1)	Set a positive number so that the array elements are equal to or greater than the number of points to be read. When reading the entire operation profile data (Offset (Offset) and Number of read data (Points) are set to "0"), ensure that the value is set to the number of switches (PROFILE_CAMSWITCH_DATA.NumberOfSwitches) of digital cam switch data or more. Data of an area larger than the number of switches specified in Data1 cannot be used.

Ex.

For LREAL type

Specifying interpolation method (Interpolate)	Setting example
0: Linear interpolation	The following shows examples of setting the two-dimensional device/label coordinate number "100" to Target (Target) of the TARGET_REF structure. ■Device specification TARGET_REF.Target := "[DEV](LREAL[1..100,1..2])D0" ■Label specification Data2.Target := "[VAR]CamData2" *: CamData2 declares the "LREAL[1..100,1..2]" label to data type and label name of the global label to "CamData2", and specifies the string for Target (Target). The write data is as shown below. · CamData2[n,1]: nth point current position per cycle · CamData2[n,2]: nth point stroke
1: Specifying for each section 2: Spline interpolation	The following shows examples of setting the two-dimensional device/label resolution "100" to Target (Target) of the TARGET_REF structure. ■Device specification TARGET_REF.Target := "[DEV](LREAL[1..101,1..2])D0" ■Label specification TARGET_REF.Target := "[VAR]CamData2" *: CamData2 declares the "LREAL[1..101,1..2]" label to data type and label name of the global label to "CamData2", and specifies the string for Target (Target). The write data is as shown below. · CamData2[1,1]: Value set for the start point · CamData2[1,2]: Value set for the initial stroke amount · CamData2[n+1,1]: nth point current position per cycle · CamData2[n+1,2]: nth point stroke

Ex.

For PROFILE_CAM_ELEMENT type

Setting example
The following shows examples of setting the two-dimensional device/label number of sections "10" to Target (Target) of the TARGET_REF structure. ■Device specification TARGET_REF.Target := "[DEV](PROFILE_CAM_ELEMENT[1..10])D0" ■Label specification Data2.Target := "[VAR]CamData2" *: CamData2 declares the "PROFILE_CAM_ELEMENT[1..10]" label to data type and label name of the global label to "CamData2", and specifies the string for Target (Target). The write data is as shown below. · CamData2[n]: nth point section data (PROFILE_CAM_ELEMENT type)

Point

- If the type of the specified data structure for writing does not match the type that is required for each operation profile data, Read/Write Data Type Incorrect (error code: 1BAAH) occurs.
- For details on the cam data for the operation profile data set in Write data 1 (Data1)/Write data 2 (Data2), refer to the following.
  User's manual for the controller used
- If the Write destination (Target) of MCv_WriteProfileData (Profile Write) is set to "1: File" with an LREAL type array and then data is written, the data is output to the file with Specifying interpolation method (Interpolate) for the cam data format set to "0: Linear Interpolation".
- Using [VAR] may cause the write processing to take longer. Writing with [DEV] will be faster than with [VAR]. Therefore, use [DEV] to reduce write processing time.

■Write destination (Target)

The write destination of the operation profile data is set and writing is performed for Write Data 1 (Data1)/Write Data 2 (Data2) and the Profile ID (PROFILE_ID) of the operation profile data open FB.

Setting value	Description
0: Open area	• The Profile ID (ID) value of the PROFILE_DATA structure is referred to, and the write is performed for the operation profile data that Profile ID No. (Number) of the PROFILE_ID structure indicates in the open area. • The write is performed at the cycle of the executed program. ■When "0" is set in Profile ID No. (Number) of the PROFILE_ID structure An available ID is assigned automatically, and the ID is stored in profile ID No. (Number). If an FB with Profile ID No. (Number) specified as "0" is executed multiple times to the same operation profile data, the data will be opened in multiple different open areas. When no ID is available, Insufficient Operation Profile Data ID (error code: 1BA0H) will occur and open processing will not be performed. ■When "1 to 60000" is set in Profile ID No. (Number) of the PROFILE_ID structure The set ID is opened without changing the ID. If setting an ID that has been already opened, the open data will be overwritten.
1: File	• The Profile ID (ID) value of the PROFILE_DATA structure is referred, and the write is performed for the operation profile data that Profile ID No. (Number) of the PROFILE_ID structure indicates in the open area. Following this, the Operation profile data storage location (Location) value of the PROFILE_DATA structure is referred, and the write is performed for the operation profile data that the File name (Filename) and Folder specification (Path) indicate within the FILE_LOCATION structure. With the Profile ID of the operation profile data, the Profile ID (ProfileID) value of the output variable will be written in the file. • The write is performed at the cycle of the motion service processing. • If Operation profile data storage location (Location) of the PROFILE_DATA structure is not set, No Operation Profile Data (error code: 1B8EH) occurs and the write processing is not performed. • If the specified file exists, the file is overwritten. • If the specified file does not exist, a new file will be created and written in "Unicode (UTF-16 Little Endian with BOM)" if the value in both Offset (Offset) and Number of write data (Points) is "0". • When creating a new file, both Offset (Offset) and Number of write data (Points) must be "0", otherwise Number of Offset/Read/Write Data Incorrect (error code: 1BADH) occurs. (Write to the open area is executed.) • When set to "1: File", set Offset (Offset) and Number of write data (Points) to "0". (When a value other than 0 is set, Number of Offset/Read/Write Data Incorrect (error code: 1BADH) occurs.)
2: Open area (motion service processing)	• The Profile ID (ID) value of the PROFILE_DATA structure is referred to, and the write is performed for the operation profile data that Profile ID No. (Number) of the PROFILE_ID structure indicates in the open area. • The write is performed at the cycle of the motion service processing. ■When "0" is set in Profile ID No. (Number) of the PROFILE_ID structure An available ID is assigned automatically, and the ID is stored in profile ID No. (Number). If an FB with Profile ID No. (Number) specified as "0" is executed multiple times to the same operation profile data, the data will be opened in multiple different open areas. When no ID is available, Insufficient Operation Profile Data ID (error code: 1BA0H) will occur and open processing will not be performed. ■When "1 to 60000" is set in Profile ID No. (Number) of the PROFILE_ID structure The set ID is opened without changing the ID. If setting an ID that has been already opened, the open data will be overwritten.

Point

- When Read destination (Target) is "1: File" or "2: Open area (motion service processing)", as motion service processing is used in that case, it may take much time for the processing to complete. Use Done (Execution completion) for the interlock as needed.
- When Target is set to "0: Open area", the processing is completed immediately as it is performed within the task where the FB is called. The program execution time increases when the number of write data points is too large. Therefore, execute the write multiple times with smaller data quantities as needed, or use the setting with Target set to "2: Open area (motion service processing)".

■Execution mode (ExecutionMode)

The timing for executing MCv_WriteProfileData (Profile Write) is set.

Setting value	Description
0: Execute immediately (mcImmediately)	The contents of the open area are immediately written. The writing may affect the control in execution. Note that if the operation profile data format and resolution do not match when performing a write during FB execution, Operation Profile Data Being Operated (error code: 1B90H) occurs.
1: Execute at completion (mcQueued)	Writing is performed after waiting for the FB being executed to complete. (Done = TRUE) When multiple FBs are waiting, the FBs will be executed in order of task priority. FBs of the same priority are executed according to the start order.
3: Execute speculatively (mcSpeculatively)	Operation profile data being operated (error code: 1B90H) occurs and the open area is not changed.

Point

When no FB is in execution, the FB is executed immediately regardless of the setting of Execution mode (ExecutionMode).
When FBs in execution are accessing the open area, Executing (Busy) of each FB accessing the open area becomes TRUE. Use Execution mode (ExecutionMode) for the interlock as needed.

Precautions

- The number of data points that can be written at the same time are limited. If the write is not completed in a single FB execution, execute the write over several FB executions.

Program example

<Program example for writing operation profile data in which the operation profile data format is cam data>

- The following is a program example. "LREAL type (two-dimensional array) format" cam data 1 (ProfileData0001) and "PROFILE_CAM_ELEMENT format" cam data 2 (ProfileData0002) are written to the file.

■Operation profile data

Profile ID	Label name	Data type	Comment
1	ProfileData0001	MC_CAM_REF	Cam data 1
2	ProfileData0002	MC_CAM_REF	Cam data 2

- Operation profile data (cam data) settings

Item		ProfileData0001	ProfileData0002
Specifying interpolation method		Linear interpolation	Specifying for each section
Resolution/number of coordinates		5	256
Length per cycle setting	Unit	degree	degree
	Length per cycle	360.00000	360.00000
Stroke amount	Unit	pulse	%
	Stroke amount	100.0	100.0000000
Cam 1 cycle time		—	1.000 [s]
Initial stroke value		—	0.0000000
Cam 1 stroke minimum value		0.00000	—
Cam 1 stroke maximum value		360.00000	—

- ProfileData0001

Point No.	Input value [degree]	Output value [pulse]
1	0.00000	0.0
2	90.00000	60.0
3	180.00000	100.0
4	270.00000	30.0
5	360.00000	0.0

• ProfileData0002

Section No.	Start point [degree]	End point [degree]	Stroke [%]	Cam curve type
1	0.00000	90.00000	60.0000000	Single hypotenuse
2	90.00000	180.00000	100.0000000	Single hypotenuse
3	180.00000	270.00000	30.0000000	Single hypotenuse
4	270.00000	0.00000	0.0000000	Single hypotenuse

■Labels used

Label name	Data type	Comment
MCv_WriteProfileData_1	MCv_WriteProfileData	Profile Write FB 1
WriteData1_1	TARGET_REF	Write data 1 Profile 1
WriteData2_1	TARGET_REF	Write data 2 Profile 1
bWriteProfile1	Bit	Profile write command 1
bDone1	Bit	Completion 1
bBusy1	Bit	Executing 1
bError1	Bit	Error 1
uwErrorID1	Word [unsigned]/bit string [16-bit]	Error code 1
ProfileID1	PROFILE_ID	Profile ID 1
MCv_WriteProfileData_2	MCv_WriteProfileData	Profile Write FB 2
WriteData1_2	TARGET_REF	Write data 1 Profile 2
WriteData2_2	TARGET_REF	Write data 2 Profile 2
bWriteProfile2	Bit	Profile write command 2
bDone2	Bit	Completion 2
bBusy2	Bit	Executing 2
bError2	Bit	Error 2
uwErrorID2	Word [unsigned]/bit string [16-bit]	Error code 2
ProfileID2	PROFILE_ID	Profile ID 2

■Global labels used

Label name	Data type	Comment
G_WriteData1_Lreal*1	PROFILE_CAM_DATA	Profile data 1 Cam data 1
G_WriteData2_Lreal*1	Double-precision real number (0..4,0..1)	Profile data 1 LREAL type
G_WriteData1_Element*1	PROFILE_CAM_DATA	Profile data 2 Cam data 2
G_WriteData2_Element*1	PROFILE_CAM_ELEMENT (0..3)	Profile data 2 PROFILE_CAM_ELEMENT type

*1 Select "Access from External Device" for global labels to be used.

■ST program

```
//-----Write data 1-----
WriteData1_1.Target := "[VAR]G_WriteData1_Lreal";
WriteData1_2.Target := "[VAR]G_WriteData2_Lreal";

//-----Profile Write 1-----
//-----Writes cam data in the LREAL type (two-dimensional array) format-----
MCv_WriteProfileData_1(
   ProfileData:= ProfileData0001.ProfileData ,
   Data1:= WriteData1_1 ,
   Data2:= WriteData1_2 ,
   Execute:= bWriteProfile1 ,
   Offset:= 0 ,
   Points:= 0 ,
   Target:= 1 ,
   ExecutionMode:= MC_EXECUTION_MODE__mcImmediately ,
   Done=> bDone1 ,
   Busy=> bBusy1 ,
   Error=> bError1 ,
   ErrorID=> uwErrorID1 ,
   ProfileID=> ProfileID1
);

//-----Write data 2-----
WriteData2_1.Target := "[VAR]G_WriteData1_Element";
WriteData2_2.Target := "[VAR]G_WriteData2_Element";

//-----Profile Write 2-----
//-----Writes cam data in the PROFILE_CAM_ELEMENT type format-----
MCv_WriteProfileData_2(
   ProfileData:= ProfileData0002.ProfileData ,
   Data1:= WriteData2_1 ,
   Data2:= WriteData2_2 ,
   Execute:= bWriteProfile2 ,
   Offset:= 0 ,
   Points:= 0 ,
   Target:= 1 ,
   ExecutionMode:= MC_EXECUTION_MODE__mcImmediately ,
   Done=> bDone2 ,
   Busy=> bBusy2 ,
   Error=> bError2 ,
   ErrorID=> uwErrorID2 ,
   ProfileID=> ProfileID2
);
```

<Program example for writing operation profile data in which the operation profile data format is multiple axes positioning data>

- The following is an example of a program that reads the profile data for multiple axes positioning data 1 (PositioningData0001) from the open area. For positioning data number (DataNo) 3, the target position/movement amount/end point (interpolation axis 1) (Position1) is updated to 300.0, and the velocity (Velocity) is updated to 120.0.

■Operation profile data

Profile ID	Label name	Data type	Comment
1	PositioningData0001	MC_POSITIONING_DATA_REF	Multiple axes positioning data 1

- Operation profile data (PositioningData0001 (multiple axes positioning data)) settings

Item		Positioning data number					
		1	2	3	4	5	6
Operation pattern		Continuous Path (BlendingPrevious)	Continuous Path (BlendingPrevious)	Continuous Path (BlendingPrevious)	Continuous Path (BlendingPrevious)	Continuous Positioning	Positioning Complete
Control Method		Absolute Value Linear Interpolation	Absolute Value Circular Interpolation	Absolute Value Linear Interpolation	Relative Value Circular Interpolation	Relative Value Linear Interpolation	Absolute Value Linear Interpolation
Interpolation Axes	Interpolation Axes 1	Axis0001	Axis0001	Axis0001	Axis0001	Axis0001	Axis0001
	Interpolation Axes 2	Axis0002	Axis0002	Axis0002	Axis0002	Axis0002	Axis0002
Circular Interpolation Mode		—	Center Point Specification	—	Center Point Specification	—	—
Path Selection		—	CW	—	CW	—	—
Target Position/ Movement Amount/End Point	Interpolation Axes 1	200.0	500.0	1000.0	300.0	0.0	200.0
	Interpolation Axes 2	400.0	700.0	700.0	-300.0	-200.0	200.0
Direction Selection	Interpolation Axes 1	Positive Direction	—	Positive Direction	—	—	Positive Direction
	Interpolation Axes 2	Positive Direction	—	Positive Direction	—	—	Positive Direction
Border Point/ Center Point/ Radius	Interpolation Axes 1	—	500.0	—	0.0	—	—
	Interpolation Axes 2	—	400.0	—	-300.0	—	—
Velocity Mode		Vector Velocity Specification	—	Vector Velocity Specification	—	Vector Velocity Specification	Vector Velocity Specification
Velocity		100.0	150.0	100.0	150.0	100.0	100.0
Acceleration/Deceleration Method		Acceleration/ Deceleration Specifying Method	—	—	—	—	—
Acceleration and Acceleration/ Deceleration Time		400.0	0.0	0.0	0.0	0.0	0.0
Deceleration		500.0	0.0	0.0	0.0	0.0	0.0
Jerk		500.0	0.0	0.0	0.0	0.0	0.0
Circular Interpolation Error Tolerance		—	—	—	—	—	—
Target Position Specification Exceeding the Ring Counter		Do Not Allow	—	Do Not Allow	—	—	Do Not Allow
Dwell Time		500.0	0.0	0.0	0.0	0.3	0.0
M Code		0	0	0	0	0	0
M Code Output Timing Override		—	—	—	—	—	—

■Labels used

Label name	Data type	Comment
MCv_ReadProfileData_1	MCv_ReadProfileData	Profile read
stReadData1	TARGET_REF	Read data 1
stReadData2	TARGET_REF	Read data 2
bReadProfile	Bit	Profile read command
bDoneReadProfile	Bit	Profile read completion
bBusyReadProfile	Bit	Executing profile read
bErrorReadProfile	Bit	Profile read error
bSetWriteData	Bit	Write data setting command
MCv_WriteProfileData_1	MCv_WriteProfileData	Profile write
stWriteData1	TARGET_REF	Write data 1
stWriteData2	TARGET_REF	Write data 2
bWriteProfile	Bit	Profile write command
bDoneWriteProfile	Bit	Profile write completion
bBusyWriteProfile	Bit	Executing profile write
bErrorWriteProfile	Bit	Profile write error

■Global labels used

Label name	Data type	Comment
PositioningData0001	MC_POSITIONING_DATA_REF	Positioning data1*1
G_stReadProfilePositioningData	PROFILE_POSITIONING_DATA	A variable specified by read data 1
G_st6ReadProfilePositioningDataElement	PROFILE_POSITIONING_DATA_ELEMENT(1..6)	A variable specified by read data 2
G_stWriteProfilePositioningData	PROFILE_POSITIONING_DATA	A variable specified by write data 1
G_st6WriteProfilePositioningDataElement	PROFILE_POSITIONING_DATA_ELEMENT(1..6)	A variable specified by write data 2

*1 A global label of the profile data type that is automatically added by the engineering tool when creating an operation profile data.

■ST program

```
//-----Read Data-----
stReadData1.Target:= "[VAR]G_stReadProfilePositioningData";
stReadData2.Target:= "[VAR]G_st6ReadProfilePositioningDataElement";

//-----Profile Read-----
//Profile Read (MCv_ReadProfileData_1) is performed by setting Profile read command (bReadProfile) to TRUE.
MCv_ReadProfileData_1(
   ProfileData:= PositioningData0001.ProfileData ,
   Data1:= stReadData1 ,
   Data2:= stReadData2 ,
   Execute:= bReadProfile ,
   Offset:= 0 ,
   Points:= 0 ,
   Target:= 0 ,
   Done=> bDoneReadProfile ,
   Busy=> bBusyReadProfile ,
   Error=> bErrorReadProfile
);

//-----Write Data Setting-----
//Read data is partially changed by setting Write data setting command (bSetWriteProfile) to TRUE.
IF bSetWriteData THEN
   G_stWriteProfilePositioningData := G_stReadProfilePositioningData;
   G_st6WriteProfilePositioningDataElement := G_st6ReadProfilePositioningDataElement;
   G_st6WriteProfilePositioningDataElement[3].Position1 := 300.0;
   G_st6WriteProfilePositioningDataElement[3].Velocity := 120.0;
bSetWriteData:= FALSE;
END_IF;

//-----Write Data-----
stWriteData1.Target:= "[VAR]G_stWriteProfilePositioningData";
stWriteData2.Target:= "[VAR]G_st6WriteProfilePositioningDataElement";

//-----Profile Write-----
//Profile Write (MCv_WriteProfileData_1) is performed by setting Profile write command (bWriteProfile) to TRUE.
MCv_WriteProfileData_1(
   ProfileData:= PositioningData0001.ProfileData ,
   Data1:= stWriteData1 ,
   Data2:= stWriteData2 ,
   Execute:= bWriteProfile ,
   Offset:= 0 ,
   Points:= 0 ,
   Target:= 0 ,
   ExecutionMode:= MC_EXECUTION_MODE__mcQueued ,
   Done=> bDoneWriteProfile ,
   Busy=> bBusyWriteProfile ,
   Error=> bErrorWriteProfile
);
```

<Program example for writing operation profile data (including condition signal, skip signal) in which the operation profile data format is multiple axes positioning data>

- The following is an example of a program that reads the profile data for multiple axes positioning data 2 (PositioningData0002) from the open area. For positioning data number (DataNo) 5, the skip signal number (SkipSignalNo) is updated to "3," the signal detection method (Detection) for condition signal number 1 in the condition signal settings is updated to "1: Detection at FALSE (LowLevel)," and the filter time for skip signal number 1 in the skip signal settings is updated to "filter time (FilterTime) = 0.001 [s]."

■Operation profile data

Profile ID	Label name	Data type	Comment
2	PositioningData0002	MC_POSITIONING_DATA_REF	Multiple axes positioning data 2

- Operation profile data (PositioningData0002 (multiple axes positioning data)) settings

Item		Positioning data number						
		1	2	3	4	5	6	7
Operation pattern		Continuous Path (BlendingPrevious)	Continuous Path (BlendingPrevious)	Continuous Path (BlendingPrevious)	Continuous Path (BlendingPrevious)	Continuous Positioning	—	Positioning Complete
Control Method		Absolute Value Linear Interpolation	Absolute Value Circular Interpolation	Absolute Value Linear Interpolation	Relative Value Circular Interpolation	Relative Value Linear Interpolation	JUMP	Absolute Value Linear Interpolation
Interpolation Axes	Interpolation Axes 1	Axis0001	Axis0001	Axis0001	Axis0001	Axis0001	—	Axis0001
	Interpolation Axes 2	Axis0002	Axis0002	Axis0002	Axis0002	Axis0002	—	Axis0002
Circular Interpolation Mode		—	Center Point Specification	—	Center Point Specification	—	—	—
Path Selection		—	CW	—	CW	—	—	—
Target Position/ Movement Amount/End Point	Interpolation Axes 1	200.0	500.0	1000.0	300.0	0.0	—	200.0
	Interpolation Axes 2	400.0	700.0	700.0	-300.0	-200.0	—	200.0
Direction Selection	Interpolation Axes 1	Positive Direction	—	Positive Direction	—	—	—	Positive Direction
	Interpolation Axes 2	Positive Direction	—	Positive Direction	—	—	—	Positive Direction
Border Point/ Center Point/ Radius	Interpolation Axes 1	—	500.0	—	0.0	—	—	—
	Interpolation Axes 2	—	400.0	—	-300.0	—	—	—
Velocity Mode		Vector Velocity Specification	—	Vector Velocity Specification	—	Vector Velocity Specification	—	Vector Velocity Specification
Velocity		100.0	150.0	100.0	150.0	100.0	—	100.0
Acceleration/Deceleration Method		Acceleration/ Deceleration Specifying Method	—	—	—	—	—	—
Acceleration and Acceleration/ Deceleration Time		400.0	0.0	0.0	0.0	0.0	—	0.0
Deceleration		500.0	0.0	0.0	0.0	0.0	—	0.0
Jerk		500.0	0.0	0.0	0.0	0.0	—	0.0
Circular Interpolation Error Tolerance		—	—	—	—	—	—	—
Target Position Specification Exceeding the Ring Counter		Do Not Allow	—	Do Not Allow	—	—	—	Do Not Allow
Dwell Time		500.0	0.0	0.0	0.0	0.3	—	0.0
M Code		0	0	0	0	0	—	0
M Code Output Timing Override		—	—	—	—	—	—	—

Item	Positioning data number						
	1	2	3	4	5	6	7
JUMP Destination Positioning Data No.	—	—	—	—	—	1	—
Condition Signal No.	—	—	—	—	—	1	—
Number of Loop Repetitions	—	—	—	—	—	—	—
Skip signal No.	0	1	0	0	2	—	0

■Condition Signal Setting

Signal No.	Signal			
	Target	Signal Detection Method	Compensation Time	Filter Time
1	[VAR](BOOL)G_bConditionSignal1	0: Detection at TRUE	0.0s	0.0s

■Skip signal setting

Signal No.	Signal			
	Target	Signal Detection Method	Compensation Time	Filter Time
1	[VAR](BOOL)G_bSkipSignal1	0: Detection at TRUE	0.0s	0.0s
2	[VAR](BOOL)G_bSkipSignal2	0: Detection at TRUE	0.0s	0.0s
3	[VAR](BOOL)G_bSkipSignal3	0: Detection at TRUE	0.0s	0.0s

■Labels used

Label name	Data type	Comment
MCv_ReadProfileData_1	MCv_ReadProfileData	Profile read
stReadData1	TARGET_REF	Read data 1
stReadData2	TARGET_REF	Read data 2
bReadProfile	Bit	Profile read command
bDoneReadProfile	Bit	Profile read completion
bBusyReadProfile	Bit	Executing profile read
bErrorReadProfile	Bit	Profile read error
bSetWriteData	Bit	Write data setting command
MCv_WriteProfileData_1	MCv_WriteProfileData	Profile write
stWriteData1	TARGET_REF	Write data 1
stWriteData2	TARGET_REF	Write data 2
bWriteProfile	Bit	Profile write command
bDoneWriteProfile	Bit	Profile write completion
bBusyWriteProfile	Bit	Executing profile write
bErrorWriteProfile	Bit	Profile write error

■Global labels used

Label name	Data type	Comment
PositioningData0002	MC_POSITIONING_DATA_REF	Positioning data 2*1
G_stReadProfilePositioningData	PROFILE_POSITIONING_DATA	A variable specified by read data 1
G_stReadProfilePositioningDataExtended	PROFILE_POSITIONING_DATA_EXTENDED	A variable specified by read data 2
G_st10ReadProfileConditionSignal	PROFILE_SIGNAL_SELECT_ELEMENT(1..10)	A variable specified by G_stReadProfilePositioningData.ConditionSignal
G_st10ReadProfileSkipSignal	PROFILE_SIGNAL_SELECT_ELEMENT(1..10)	A variable specified by G_stReadProfilePositioningData.SkipSignal
G_st8ReadProfilePositioningDataElement	PROFILE_POSITIONING_DATA_ELEMENT(1..8)	A variable specified by G_stReadProfilePositioningData.PositioningData
G_stWriteProfilePositioningData	PROFILE_POSITIONING_DATA	A variable specified by write data 1
G_stWriteProfilePositioningDataExtended	PROFILE_POSITIONING_DATA_EXTENDED	A variable specified by write data 2
G_st10WriteProfileConditionSignal	PROFILE_SIGNAL_SELECT_ELEMENT(1..10)	A variable specified by G_stWriteProfilePositioningData.ConditionSignal
G_st10WriteProfileSkipSignal	PROFILE_SIGNAL_SELECT_ELEMENT(1..10)	A variable specified by G_stWriteProfilePositioningData.SkipSignal
G_st8WriteProfilePositioningDataElement	PROFILE_POSITIONING_DATA_ELEMENT(1..8)	A variable specified by G_stWriteProfilePositioningData.PositioningData
G_bConditionSignal1	Bit	A signal used as ConditionSignal1 of positioning data 2
G_bConditionSignal1Override	Bit	A signal used as ConditionSignal1 of positioning data 2 (after writing the profile)
G_bSkipSignal1	Bit	A signal used as SkipSignal1 of positioning data 2
G_bSkipSignal1Override	Bit	A signal used as SkipSignal1 of positioning data 2 (after writing the profile)
G_bSkipSignal2	Bit	A signal used as SkipSignal2 of positioning data 2
G_bSkipSignal3	Bit	A signal used as SkipSignal3 of positioning data 2

*1 A global label of the profile data type that is automatically added by the engineering tool when creating an operation profile data.

■ST program

```
//-----Read Data-----
G_stReadProfilePositioningDataExtended.ConditionSignal := "[VAR]G_st10ReadProfileConditionSignal";
G_stReadProfilePositioningDataExtended.SkipSignal := "[VAR]G_st10ReadProfileSkipSignal";
G_stReadProfilePositioningDataExtended.PositioningData := "[VAR]G_st8ReadProfilePositioningDataElement";
stReadData1.Target:= "[VAR]G_stReadProfilePositioningData";
stReadData2.Target:= "[VAR]G_stReadProfilePositioningDataExtended";

//-----Profile Read-----
//Profile Read (MCv_ReadProfileData_1) is performed by setting Profile read command (bReadProfile) to TRUE.
MCv_ReadProfileData_1(
   ProfileData:= PositioningData0002.ProfileData ,
   Data1:= stReadData1 ,
   Data2:= stReadData2 ,
   Execute:= bReadProfile ,
   Offset:= 0 ,
   Points:= 0 ,
   Target:= 2 ,
   Done=> bDoneReadProfile ,
   Busy=> bBusyReadProfile ,
   Error=> bErrorReadProfile
);

//-----Write Data Setting-----
//Read data is partially changed by setting Write data setting command (bSetWriteProfile) to TRUE.
IF bSetWriteData THEN
   G_stWriteProfilePositioningData := G_stReadProfilePositioningData;
   G_st10WriteProfileConditionSignal := G_st10ReadProfileConditionSignal;
   G_st10WriteProfileConditionSignal[1].Target := "[VAR]G_bConditionSignal1Override";
   G_st10WriteProfileConditionSignal[1].Detection := MC_SIGNAL_LOGIC__LowLevel;
   G_st10WriteProfileSkipSignal := G_st10ReadProfileSkipSignal;
   G_st10WriteProfileSkipSignal[1].Target := "[VAR]G_bSkipSignal1Override";
   G_st10WriteProfileSkipSignal[1].FilterTime := 0.001;
   G_st8WriteProfilePositioningDataElement := G_st8ReadProfilePositioningDataElement;
   G_st8WriteProfilePositioningDataElement[5].SkipSignalNo := 3;
   bSetWriteData := FALSE;
END_IF;

//-----Write Data-----
G_stWriteProfilePositioningDataExtended.ConditionSignal := "[VAR]G_st10WriteProfileConditionSignal";
G_stWriteProfilePositioningDataExtended.SkipSignal := "[VAR]G_st10WriteProfileSkipSignal";
G_stWriteProfilePositioningDataExtended.PositioningData := "[VAR]G_st8WriteProfilePositioningDataElement";
stWriteData1.Target := "[VAR]G_stWriteProfilePositioningData";
stWriteData2.Target := "[VAR]G_stWriteProfilePositioningDataExtended";

//-----Profile Write-----
//Profile Write (MCv_WriteProfileData_1) is performed by setting Profile write command (bWriteProfile) to TRUE.
MCv_WriteProfileData_1(
   ProfileData:= PositioningData0002.ProfileData ,
   Data1:= stWriteData1 ,
   Data2:= stWriteData2 ,
   Execute:= bWriteProfile ,
```

```
    Offset:= 0 ,
    Points:= 0 ,
    Target:= 2 ,
    ExecutionMode:= MC_EXECUTION_MODE__mcQueued ,
    Done=> bDoneWriteProfile ,
    Busy=> bBusyWriteProfile ,
    Error=> bErrorWriteProfile
);
```

<Program example for writing operation profile data in which the operation profile data format is digital cam switch data>

- The following is an example of a program that reads the switch setting data 1 (CamSwitchData0001), which has the operation profile format of "digital cam switch," from the open area and partially changes.

■Operation profile data

Profile ID	Label name	Format	Comment
1	CamSwitchData0001	Digital cam switch	Digital cam switch

- Operation profile data (CamSwitchData0001 (digital cam switch)) settings

Item			Description
Input Data Information	Position Unit		pulse
	Velocity Unit		U/s
	Ring Counter Lower Limit Value		-10000000000.0
	Ring Counter Upper Limit Value		10000000000.0
Switch Setting	Switch Mode	Switch No.1	1: Position reference
		Switch No.2	
		Switch No.3	
	Lower Limit Position	Switch No.1	-8000000000.0
		Switch No.2	-3000000000.0
		Switch No.3	-1000000000.0
	Upper Limit Position	Switch No.1	-6000000000.0
		Switch No.2	-2000000000.0
		Switch No.3	0.0
	Movement Direction	Switch No.1	0: Both directions
		Switch No.2	1: Positive direction
		Switch No.3	2: Negative direction

■Labels used

Label name	Data type	Comment
MCv_ReadProfileData_1	MCv_ReadProfileData	Profile read
stReadData1	TARGET_REF	Read data 1
stReadData2	TARGET_REF	Read data 2
bReadProfile	Bit	Profile read command
bDoneReadProfile	Bit	Profile read completion
bBusyReadProfile	Bit	Executing profile read
bErrorReadProfile	Bit	Profile read error
bSetWriteData	Bit	Write data setting command
MCv_WriteProfileData_1	MCv_WriteProfileData	Profile write
stWriteData1	TARGET_REF	Write data 1
stWriteData2	TARGET_REF	Write data 2
bWriteProfile	Bit	Profile write command
bDoneWriteProfile	Bit	Profile write completion
bBusyWriteProfile	Bit	Executing profile write
bErrorWriteProfile	Bit	Profile write error

■Global labels used

Label name	Data type	Comment
CamSwitchData0001	MC_CAMSWITCH_REF	Digital cam switch data 1*1
G_stReadCamSwitchData	PROFILE_CAMSWITCH_DATA	A variable specified by read data 1
G_st3ReadCamSwitchElement	PROFILE_CAMSWITCH_ELEMENT(1..3)	A variable specified by read data 2
G_stWriteCamSwitchData	PROFILE_CAMSWITCH_DATA	A variable specified by write data 1
G_st3WriteCamSwitchElement	PROFILE_CAMSWITCH_ELEMENT(1..3)	A variable specified by write data 2

*1 A global label of the profile data type that is automatically added by the engineering tool when creating an operation profile data.

■ST program

```
//-----Read Data-----
stReadData1.Target := "[VAR]G_stReadCamSwitchData";
stReadData2.Target := "[VAR]G_st3ReadCamSwitchElement";

//-----Profile Read-----
//Profile Read (MCv_ReadProfileData_1) is performed by setting Profile read command (bReadProfile) to TRUE.
MCv_ReadProfileData_1(
   ProfileData:= CamSwitchData0001.ProfileData ,
   Data1:= stReadData1 ,
   Data2:= stReadData2 ,
   Execute:= bReadProfile ,
   Offset:= 0 ,
   Points:= 0 ,
   Target:= 2 ,
   Done=> bDoneReadProfile ,
   Busy=> bBusyReadProfile ,
   Error=> bErrorReadProfile
);

//-----Write Data Setting-----
//Read data is partially changed by setting the write data setting command (bSetWriteData) to TRUE.
IF bSetWriteData THEN
   G_stWriteCamSwitchData:= G_stReadCamSwitchData;
   G_st3WriteCamSwitchElement:= G_st3ReadCamSwitchElement;
   G_st3WriteCamSwitchElement[1].FirstOnPosition:= -6000000000.0;
   G_st3WriteCamSwitchElement[3].LastOnPosition:= 4000000000.0;
   bSetWriteData := FALSE;
END_IF;

//-----Write Data-----
stWriteData1.Target:= "[VAR]G_stWriteCamSwitchData";
stWriteData2.Target:= "[VAR]G_st3WriteCamSwitchElement";

//-----Profile Write-----
//Profile Write (MCv_WriteProfileData_1) is performed by setting Profile write command (bWriteProfile) to TRUE.
MCv_WriteProfileData_1(
   ProfileData:= CamSwitchData0001.ProfileData ,
   Data1:= stWriteData1 ,
   Data2:= stWriteData2 ,
   Execute:= bWriteProfile ,
   Offset:= 0 ,
   Points:= 0 ,
   Target:= 2 ,
   ExecutionMode:= MC_EXECUTION_MODE__mcQueued ,
   Done=> bDoneWriteProfile ,
   Busy=> bBusyWriteProfile ,
   Error=> bErrorWriteProfile
);
```

PART 9 NETWORK FUNCTION BLOCKS

This part consists of the following chapters.

44 FUNCTION BLOCKS FOR Ethernet

45 FUNCTION BLOCKS FOR CC-Link IE TSN

44 FUNCTION BLOCKS FOR Ethernet

Point

The function blocks in this chapter are controller module FBs. Use them from [Module] ⇨ [Module FB] of the Element Selection window of the engineering tool.

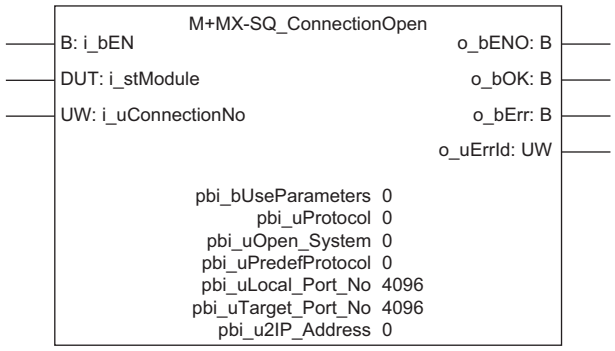
44.1 Opening/Disconnecting a Connection

M+MX-SQ_ConnectionOpen

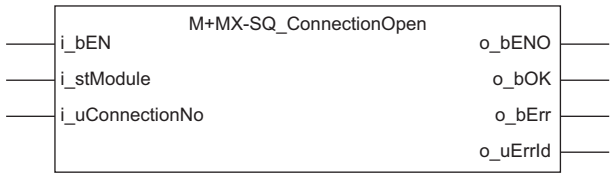
Opens (establishes) a connection.

To specify the encrypted predefined protocols (TLS and DTLS), the external device must also support them.

Ladder



FBD/LD



ST

```
M_MX_SQ_ConnectionOpen(
   i_bEN:= ?BOOL? ,
   i_stModule:= ?M+MX_SQ? ,
   i_uConnectionNo:= ?WORD? ,
   o_bENO=> ?BOOL? ,
   o_bOK=> ?BOOL? ,
   o_bErr=> ?BOOL? ,
   o_uErrID=> ?WORD?
);
```

Setting data

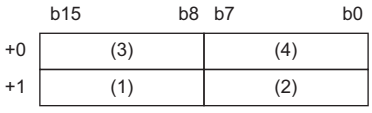
■Input arguments

Variable name	Name	Data type	Range	Description
i_bEN	Execution command	Bit	—	On: Start FB. Off: Do not start FB.
i_stModule	Module label	Structure	—	Specify the module label of the controller executing the FB. (Example: MX_SQ)
i_uConnectionNo	Connection No.	Word [unsigned]/bit string [16-bit]	1 to 64	Specify the number of the connection to be opened.

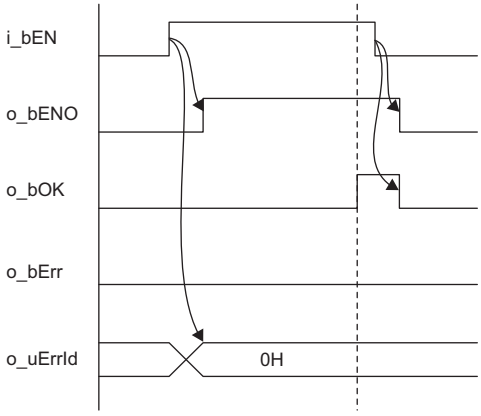
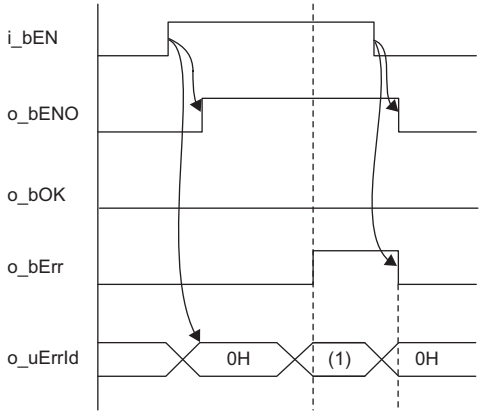
■Output arguments

Variable name	Name	Data type	Description	Default value
o_bENO	Execution status	Bit	On: The execution command is turned on. Off: The execution command is turned off.	Off
o_bOK	Normal completion	Bit	The FB has been processed normally when this argument is on.	Off
o_bErr	Error completion	Bit	The FB has been processed abnormally when this argument is on.	Off
o_uErrId	Error code	Word [unsigned]/bit string [16-bit]	An error code is stored at error completion.	0

■Operation parameters

Variable name	Name	Data type	Range	Description	Default value
pbi_bUseParameters	Parameter used	Bit	On or off	Specify whether to use the parameter values set by the engineering tool or the following operation parameter values when processing for opening a connection. • Off: Performs open processing according to the external device configuration setting made by the engineering tool. (The following operation parameters need not be set. Any settings are ignored if made.) • On: Performs open processing according to the following operation parameters.	Off
pbi_uProtocol	Protocol	Word [unsigned]/ bit string [16-bit]	0 to 3	Select the protocol to be used for the connection to be opened. • pbi_uPredefProtocol (predefined protocol setting) is 0 0: TCP 1: UDP 2: TLS 3: DTLS • pbi_uPredefProtocol (predefined protocol setting) is 1 0: TCP 1: UDP	0
pbi_uOpen_System	Open method	Word [unsigned]/ bit string [16-bit]	0 to 2	Select the connection open method. • 0: Active open or UDP or DTLS • 1: Unpassive open • 2: Fullpassive open	0
pbi_uPredefProtocol	Communication protocol setting	Word [unsigned]/ bit string [16-bit]	0 to 1	Select whether to use the predefined protocol function. • 0: Do not use the predefined protocol support function. (Use the socket communications function.) • 1: Use the predefined protocol support function.	0
pbi_uLocal_Port_No	Own node port number	Word [unsigned]/ bit string [16-bit]	1 to 4999, 5010 to 65534	Specify the port number of the own node. Port numbers 1 to 1023 are generally reserved port numbers (WELL KNOWN PORT NUMBERS), and therefore port numbers 1024 to 4999 and 5010 to 65534 should be used.	4096
pbi_uTarget_Port_No	Destination port number	Word [unsigned]/ bit string [16-bit]	1 to 65534, 65535	Specify the destination port number. With the connection that is assigned port No.65535 (only when the UDP or DTLS protocol is selected), data is received through all port numbers. Data cannot be sent with the connection which is assigned port No.65535 and therefore a port number from 1 to 65534 should be specified to send data.	4096
pbi_u2IP_Address	IP address of external device	Word [unsigned]/ bit string [16-bit] (0..1)	0.0.0.1 to 255.255.255.255 (00000001H to FFFFFFFFH)	Specify the IP address of an external device. Specify the third and fourth octets to the 1st word, and first and second octets to the 2nd word. Specify 255.255.255.255 (FFFFFFFFH) when performing simultaneous broadcast.  (1) to (4): IP address octet	192.168.1.1 (C0A80101H)

Processing details

Item	Description	
Available device	Target module	MELSEC MX Controller (Ethernet port)
	CPU module	MELSEC MX Controller
	Engineering tool	GX Works3
Language	Ladder diagram	
Usage of labels	Label: 36 points Latched label: 0 points The usage of labels embedded in a program depends on the controller used, the devices specified for arguments, and the option settings of GX Works3. For the option settings of GX Works3, refer to the following. GX Works3 Operating Manual	
Processing	When i_bEN (execution command) is turned on, this function opens (establishes) a connection for data communication with an external device.	
FB compilation method	Macro type	
FB operation	Pulse type (multiple-scan execution type)	
Timing chart of I/O signals	• For normal completion  • For error completion (same as in the case of a module error)  (1) Error code	

Item	Description
Precautions	■Specifications of the FBs • This module FB cannot be executed for the connection that is being used by another module FB or a dedicated instruction. An error occurs if this module FB is executed for the connection in use. • When open processing is performed according to the content of the operation parameter with pbi_bUseParameters (parameter used) set to on, the available communication means is the socket communications and predefined protocols only. • If this FB is executed for the connection for which parameters are already set by "External Device Connection Configuration Setting", conduct open processing based on the operation parameters specified by this FB. • This FB uses the label initial value by each program. When the program file using this FB is specified to boot file setting for the boot operation in the controller, specify the initial label value file by each program to the boot file setting as well. If an error code that is not described in Page 1949 Error codes appears, the initial label value files by each program may not be set correctly. In this case, specify the initial label value files by each program to the boot file setting. • When this FB is executed, an operation error may occur in the controller. In this case, check the detailed information of the operation error in event history and correct the input argument of the module FB. ■Operations of the FBs • Turn off i_bEN (execution command) after o_bOK (normal completion) or o_bErr (error completion) is turned on. By turning off i_bEN (execution command), o_bOK (normal completion) or o_bErr (error completion) is turned off and o_uErrId (error code) is cleared to 0.

■Error codes

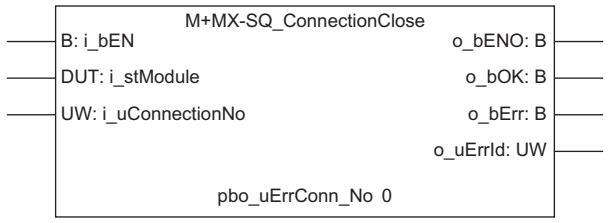
The error code occurred by this FB is the same as the SOCOPEN instruction.

☞ Page 1110 GP.SOCOPEN, SP.SOCOPEN

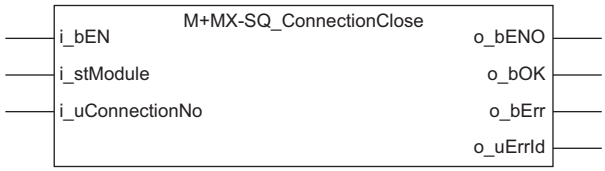
M+MX-SQ_ConnectionClose

Closes (disconnects) the connection.

Ladder



FBD/LD



ST

```
M_MX_SQ_ConnectionClose(
   i_bEN:= ?BOOL? ,
   i_stModule:= ?M+MX_SQ? ,
   i_uConnectionNo:= ?WORD? ,
   o_bENO=> ?BOOL? ,
   o_bOK=> ?BOOL? ,
   o_bErr=> ?BOOL? ,
   o_uErrID=> ?WORD?
);
```

Setting data

■Input arguments

Variable name	Name	Data type	Range	Description
i_bEN	Execution command	Bit	—	On: Start FB. Off: Do not start FB.
i_stModule	Module label	Structure	—	Specify the module label of the controller executing the FB. (Example: MX_SQ)
i_uConnectionNo	Connection No.	Word [unsigned]/bit string [16-bit]	1 to 64, 65535	Specify the number of the connection to be closed. This function closes all connections if 65535 (FFFFH) is specified.

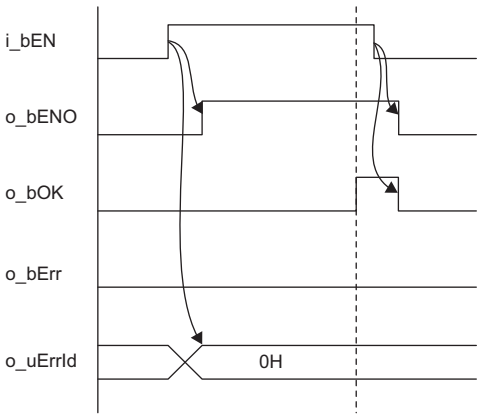
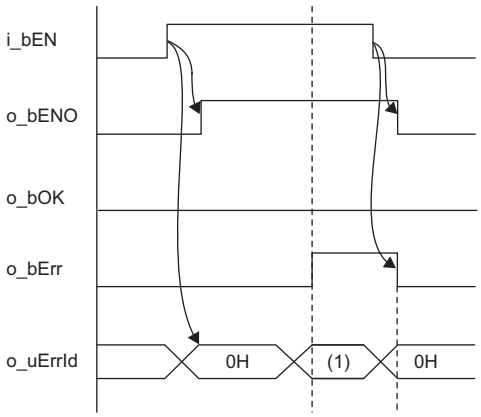
■Output arguments

Variable name	Name	Data type	Description	Default value
o_bENO	Execution status	Bit	On: The execution command is turned on. Off: The execution command is turned off.	Off
o_bOK	Normal completion	Bit	The FB has been processed normally when this argument is on.	Off
o_bErr	Error completion	Bit	The FB has been processed abnormally when this argument is on.	Off
o_uErrId	Error code	Word [unsigned]/bit string [16-bit]	An error code is stored at error completion.	0

■Public variables

Variable name	Name	Data type	Description	Default value
pbo_uErrConn_No	Error connection No.	Word [unsigned]/bit string [16-bit]	The number of the connection for which close processing was completed with an error is stored. If 65535 (FFFFH) is specified in "Connection No." (i_uConnectionNo), the number of the connection for which close processing was first completed with an error is stored.	0

Processing details

Item	Description	
Available device	Target module	MELSEC MX Controller (Ethernet port)
	CPU module	MELSEC MX Controller
	Engineering tool	GX Works3
Language	Ladder diagram	
Usage of labels	Label: 28 points Latched label: 0 points The usage of labels embedded in a program depends on the controller used, the devices specified for arguments, and the option settings of GX Works3. For the option settings of GX Works3, refer to the following. 📖GX Works3 Operating Manual	
Processing	• When i_bEN (execution command) is turned on, this function closes a connection for data communication with an external device. • The function closes all connections if 65535 (FFFFH) is specified for "Connection No." (i_uConnectionNo) in the input argument. • If the function fails to close even one connection among those specified to be closed, it is completed with an error.	
FB compilation method	Macro type	
FB operation	Pulse type (multiple-scan execution type)	
Timing chart of I/O signals	• For normal completion  • For error completion (same as in the case of a module error)  (1) Error code	
Precautions	• This module FB cannot be executed for the connection that is being used by another module FB or a dedicated instruction. An error occurs if this module FB is executed for the connection in use. • Turn off i_bEN (execution command) after o_bOK (normal completion) or o_bErr (error completion) is turned on. By turning off i_bEN (execution command), o_bOK (normal completion) or o_bErr (error completion) is turned off and o_uErrId (error code) is cleared to 0. • When this FB is executed, an operation error may occur in the controller. In this case, check the detailed information of the operation error in event history and correct the input argument of the module FB.	

Operation error

The error code occurred by this FB is the same as the SOCCLOSE instruction.

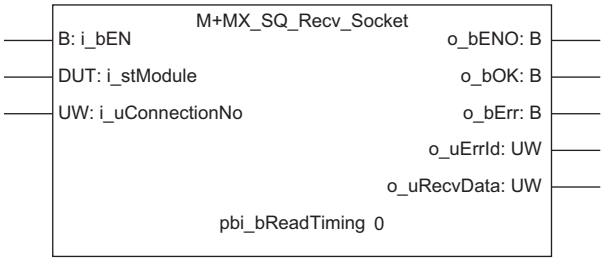
☞ Page 1115 GP.SOCCLOSE, SP.SOCCLOSE

44.2 Socket Communications

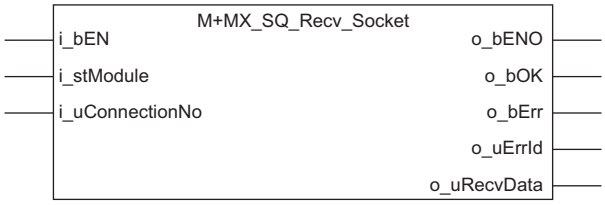
M+MX-SQ_Recv_Socket

Reads the data received from the external device through socket communications.

Ladder



FBD/LD



ST

```
M_MX_SQ_Recv_Socket(
   i_bEN:= ?BOOL? ,
   i_stModule:= ?M+MX_SQ? ,
   i_uConnectionNo:= ?WORD? ,
   o_bENO=> ?BOOL? ,
   o_bOK=> ?BOOL? ,
   o_bErr=> ?BOOL? ,
   o_uErrID=> ?WORD? ,
   o_uRecvData=> ?WORD?
);
```

Setting data

■Input arguments

Variable name	Name	Data type	Range	Description
i_bEN	Execution command	Bit	—	On: Start FB. Off: Do not start FB.
i_stModule	Module label	Structure	—	Specify the module label of the controller executing the FB. (Example: MX_SQ)
i_uConnectionNo	Connection No.	Word [unsigned]/bit string [16-bit]	1 to 64	Specify the number of the connection to be received.

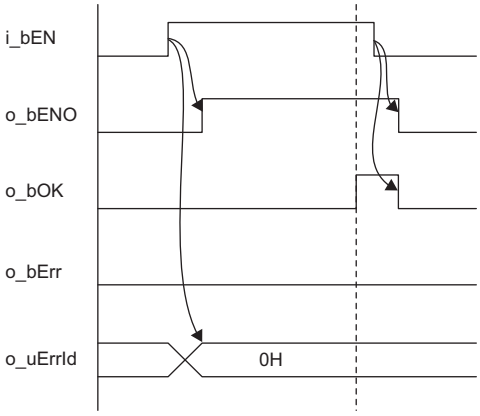
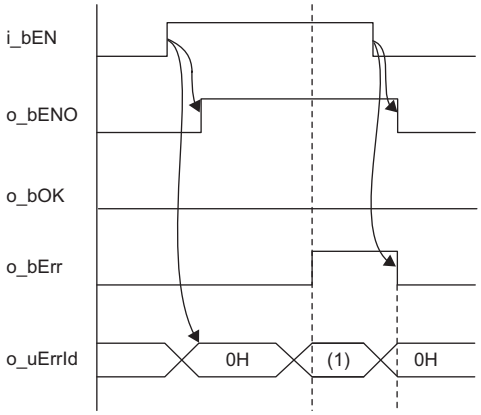
■Output arguments

Variable name	Name	Data type	Description	Default value
o_bENO	Execution status	Bit	On: The execution command is turned on. Off: The execution command is turned off.	Off
o_bOK	Normal completion	Bit	The FB has been processed normally when this argument is on.	Off
o_bErr	Error completion	Bit	The FB has been processed abnormally when this argument is on.	Off
o_uErrId	Error code	Word [unsigned]/bit string [16-bit]	An error code is stored at error completion.	0
o_uRecvData	Receive data storage destination	Word [unsigned]/bit string [16-bit]	Specify the receive data length and the start number of the device for storing received data. The data that has been read is stored sequentially in ascending order of addresses as shown below. • When the data unit is byte 1st word: Receive data length (unit: byte) 2nd to nth word: b15···b8 b7···b0 (2) (1) ⋮ (4) (3) (1) Receive data 1 (2) Receive data 2 (3) Receive data m-1 (4) Receive data m The data format, unit, and data length range of receive data vary depending on the module type and connection number. Receive data is stored in the word area in order from the first half (b0 to b7) to the second half (b8 to b15). The following cannot be specified as an argument. Specifying any of the following may cause an error (2820H: Device/label/buffer memory specification incorrect). • Dynamically specified array elements (Example: wLabel[D0]) • Digit-specified labels (Example: K4bLabel) • Indirectly specified devices (Example: @W0) • Local devices (Example: #D0)	0

■Operation parameters

Variable name	Name	Data type	Range	Description	Default value
pbi_bReadTiming	Read timing	Bit	On or off	Specify the timing of executing data read processing. • Off: Start reading soon after the module FB starts. • On: Start reading in the first END processing after the module FB starts.	Off

Processing details

Item	Description	
Available device	Target module	MELSEC MX Controller (Ethernet port)
	CPU module	MELSEC MX Controller
	Engineering tool	GX Works3
Language	Ladder diagram	
Usage of labels	Label: 24 points Latched label: 0 points The usage of labels embedded in a program depends on the controller used, the devices specified for arguments, and the option settings of GX Works3. For the option settings of GX Works3, refer to the following. GX Works3 Operating Manual	
Processing	When i_bEN (execution instruction) is turned on, this function reads the data received to the connection specified by the input argument.	
FB compilation method	Macro type	
FB operation	Pulse type (multiple-scan execution type)	
Timing chart of I/O signals	• For normal completion  • For error completion (same as in the case of a module error)  (1) Error code	
Precautions	• This module FB cannot be executed for the connection that is being used by another module FB or a dedicated instruction. An error occurs if this module FB is executed for the connection in use. • Turn off i_bEN (execution command) after o_bOK (normal completion) or o_bErr (error completion) is turned on. By turning off i_bEN (execution command), o_bOK (normal completion) or o_bErr (error completion) is turned off and o_uErrId (error code) is cleared to 0. • The execution command of this FB can be executed at any timing. However, when executing it after receiving data, G7889 to G7892 (Socket communications reception status signal) or corresponding module label must be added to conditions of the execution. • When the module FB is executed by specifying on (start reading in the first END processing after the FB starts) in operation parameter "Read timing", the module FB extends the scan time to complete data read processing within one END processing. • This FB uses the label initial value by each program. When the program file using this FB is specified to boot file setting for the boot operation in the controller, specify the initial label value file by each program to the boot file setting as well. If an error code that is not described in Page 1955 Error codes appears, the initial label value files by each program may not be set correctly. In this case, specify the initial label value files by each program to the boot file setting. • When this FB is executed, an operation error may occur in the controller. In this case, check the detailed information of the operation error in event history and correct the input argument of the module FB.	

■Error codes

The error code occurred by this FB is the same as the SOCRCV instruction or SOCRCVS instruction.

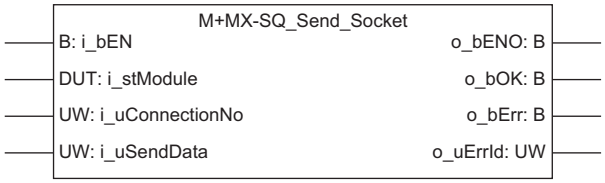
☞ Page 1117 GP.SOCRCV, SP.SOCRCV

☞ Page 1121 G.SOCRCVS, S.SOCRCVS

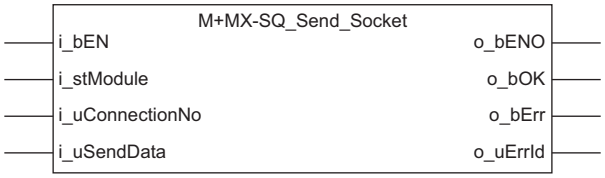
M+MX-SQ_Send_Socket

Sends data to the external device through socket communications.

Ladder



FBD/LD

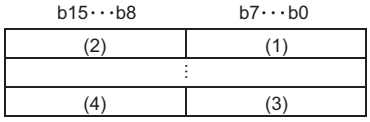


ST

```
M_MX_SQ_Send_Socket(
    i_bEN:= ?BOOL? ,
    i_stModule:= ?M+MX_SQ? ,
    i_uConnectionNo:= ?WORD? ,
    i_uSendData:= ?WORD? ,
    o_bENO=> ?BOOL? ,
    o_bOK=> ?BOOL? ,
    o_bErr=> ?BOOL? ,
    o_uErrID=> ?WORD?
);
```

Setting data

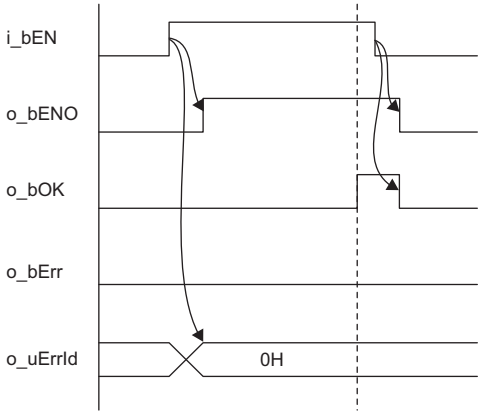
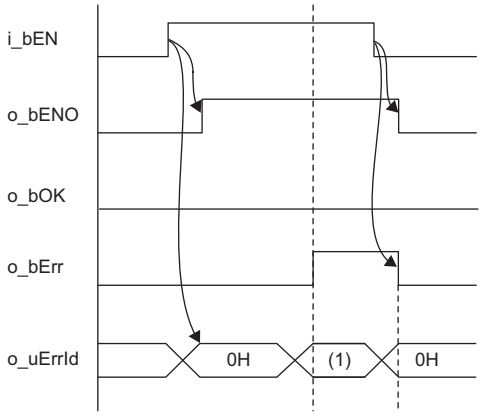
■Input arguments

Variable name	Name	Data type	Range	Description
i_bEN	Execution command	Bit	—	On: Start FB. Off: Do not start FB.
i_stModule	Module label	Structure	—	Specify the module label of the controller executing the FB. (Example: MX_SQ)
i_uConnectionNo	Connection No.	Word [unsigned]/bit string [16-bit]	1 to 64	Specify the number of the connection to be sent.
i_uSendData	Send data storage destination	Word [unsigned]/bit string [16-bit]	—	Specify the start number of the device containing the send data length and data to be sent. • When the data unit is byte 1st word: Send data length (unit: byte) 2nd to nth word:  (1) Send data 1 (2) Send data 2 (3) Send data m-1 (4) Send data m The data format and data length range of send data vary depending on the module type and the setting of the connection used. Data is sent in the word area in order from the first half (b0 to b7) to the second half (b8 to b15). The following cannot be specified as an argument. Specifying any of the following may cause an error (2820H: Device/label/buffer memory specification incorrect). • Dynamically specified array elements (Example: wLabel[D0]) • Digit-specified labels (Example: K4bLabel) • Indirectly specified devices (Example: @W0) • Local devices (Example: #D0)

■Output arguments

Variable name	Name	Data type	Description	Default value
o_bENO	Execution status	Bit	On: The execution command is turned on. Off: The execution command is turned off.	Off
o_bOK	Normal completion	Bit	The FB has been processed normally when this argument is on.	Off
o_bErr	Error completion	Bit	The FB has been processed abnormally when this argument is on.	Off
o_uErrId	Error code	Word [unsigned]/bit string [16-bit]	An error code is stored at error completion.	0

Processing details

Item	Description	
Available device	Target module	MELSEC MX Controller (Ethernet port)
	CPU module	MELSEC MX Controller
	Engineering tool	GX Works3
Language	Ladder diagram	
Usage of labels	Label: 24 points Latched label: 0 points The usage of labels embedded in a program depends on the controller used, the devices specified for arguments, and the option settings of GX Works3. For the option settings of GX Works3, refer to the following. 📖GX Works3 Operating Manual	
Processing	When i_bEN (execution instruction) is turned on, this function sends the data to the external device of the connection specified by the input argument.	
FB compilation method	Macro type	
FB operation	Pulse type (multiple-scan execution type)	
Timing chart of I/O signals	• For normal completion  • For error completion (same as in the case of a module error)  (1) Error code	
Precautions	• This module FB cannot be executed for the connection that is being used by another module FB or a dedicated instruction. An error occurs if this module FB is executed for the connection in use. • Turn off i_bEN (execution command) after o_bOK (normal completion) or o_bErr (error completion) is turned on. By turning off i_bEN (execution command), o_bOK (normal completion) or o_bErr (error completion) is turned off and o_uErrId (error code) is cleared to 0. • When this FB is executed, an operation error may occur in the controller. In this case, check the detailed information of the operation error in event history and correct the input argument of the module FB.	

Operation error

The error code occurred by this FB is the same as the SOCSND instruction.

☞ Page 1124 GP.SOCSND, SP.SOCSND

44.3 SLMP

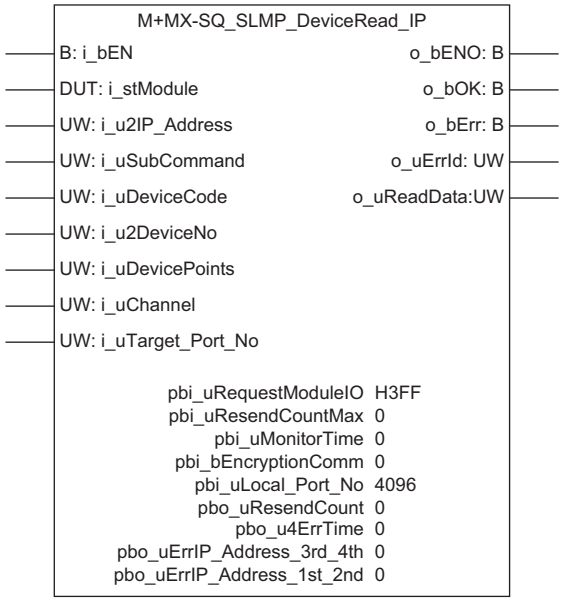
M+MX-SQ_SLMP_DeviceRead_IP

Reads device data from the SLMP-compatible device specified by an IP address.

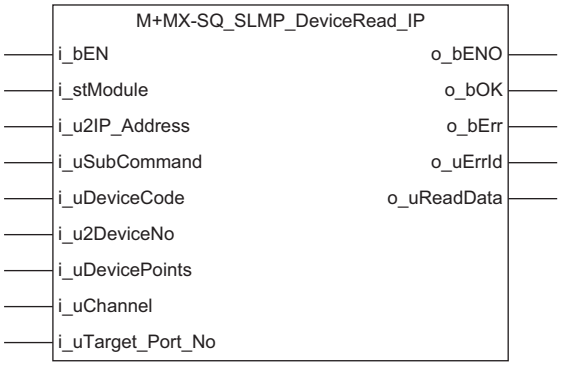
The external device must support SLMP command (Device Read).

For encrypted communications, the external device must support the encrypted predefined protocol (DTLS) in addition to the above.

Ladder



FBD/LD



ST

```
M_MX_SQ_SLMP_DeviceRead_IP(
   i_bEN:= ?BOOL? ,
   i_stModule:= ?M+MX_SQ? ,
   i_u2IP_Address:= ?WORD(0..1)? ,
   i_uSubCommand:= ?WORD? ,
   i_uDeviceCode:= ?WORD? ,
   i_u2DeviceNo:= ?WORD(0..1)? ,
   i_uDevicePoints:= ?WORD? ,
   i_uChannel:= ?WORD? ,
   i_uTarget_Port_No:= ?WORD? ,
   o_bENO=> ?BOOL? ,
   o_bOK=> ?BOOL? ,
   o_bErr=> ?BOOL? ,
   o_uErrId=> ?WORD? ,
   o_uReadData=> ?WORD?
);
```

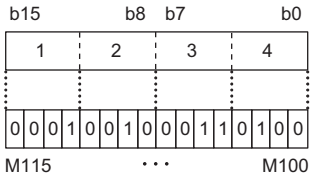
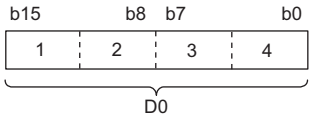
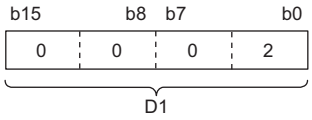
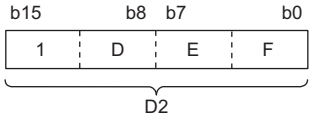
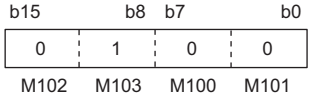
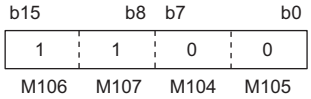
■Input arguments

Variable name	Name	Data type	Range	Description
i_bEN	Execution command	Bit	—	On: Start FB. Off: Do not start FB.
i_stModule	Module label	Structure	—	Specify the module label of the controller executing the FB. (Example: MX_SQ)
i_u2IP_Address	IP address of external device	Word [unsigned]/bit string [16-bit] (0..1)	0.0.0.1 to 253.255.255.254 (00000001H to DFFFFFFEH)	Specify the IP address of an external device. Specify the third and fourth octets to the 1st word, and first and second octets to the 2nd word. Note that the fourth octet cannot be set to 0 or 255 (FFH). b15 b8 b7 b0 +0 (3) (4) +1 (1) (2) (1) to (4): IP address octet
i_uSubCommand	Sub command	Word [unsigned]/bit string [16-bit]	—	Specify the read unit and specification method of a device. ■0th bit: Read unit 0: In units of words 1: In units of bits ■1st bit: Specification method of the device is read 0: Specify the device code in 2 digits and the start device number in 6 digits (for MELSEC-Q/L series). 1: Specify the device code in 4 digits and the start device number in 8 digits (for MELSEC iQ-R series).
i_uDeviceCode	Device code*1	Word [unsigned]/bit string [16-bit]	—	Specify the device code of the device to be read in binary code. • When the 1st bit of the subcommand is 0: 2 digits • When the 1st bit of the subcommand is 1: 4 digits
i_u2DeviceNo	Head device No.	Word [unsigned]/bit string [16-bit] (0..1)	—	Specify the start device number of the device to be read in binary code. • When the 1st bit of the subcommand is 0: 6 digits • When the 1st bit of the subcommand is 1: 8 digits
i_uDevicePoints	Number of device points	Word [unsigned]/bit string [16-bit]	—	Specify the number of device points of the device to be read in binary code. • When the 0th bit of the subcommand is 0: 1 to 960 • When the 0th bit of the subcommand is 1: 1 to 3972
i_uChannel	Own station channel	Word [unsigned]/bit string [16-bit]	1 to 9	Specify the channel to be used by the own station. Since whether or not a serial number*2 is given to the request message depends on the channel, specify the channel as follows according to the application. • 1: No serial number is given (3E frame used) • 2 to 9: Serial number is given (4E frame used)
i_uTarget_Port_No	Destination port number	Word [unsigned]/bit string [16-bit]	1 to 65534	Specify the UDP port number or DTLS port number of an external device.

*1 For details on each device code, refer to the following.
SLMP Reference Manual

*2 Give the serial numbers when sending several request messages to the same SLMP-compatible device. Serial numbers to be given are automatically numbered by the system. For the serial number, refer to the following.
SLMP Reference Manual

■Output arguments

Variable name	Name	Data type	Description	Default value
o_bENO	Execution status	Bit	On: The execution command is turned on. Off: The execution command is turned off.	Off
o_bOK	Normal completion	Bit	The FB has been processed normally when this argument is on.	Off
o_bErr	Error completion	Bit	The FB has been processed abnormally when this argument is on.	Off
o_uErrId	Error code	Word [unsigned]/bit string [16-bit]	An error code is stored at error completion.	0
o_uReadData	Read data storage destination	Word [unsigned]/bit string [16-bit]	Specify the start device number of the device for storing the read data. The read data is stored in binary code. ■When the 0th bit of the subcommand is 0 The device data is read in units of words. Example: When reading the bit device M100 to M115 (one word) in units of words 1st word:  Example: When reading the word device D0 to D2 in units of words 1st word:  2nd word:  3rd word:  ■When the 0th bit of the subcommand is 1 The device data is read in units of bits. Example: When reading the bit device M100 to M107 in units of bits 1st word:  2nd word:  	0

■Operation parameters

Variable name	Name	Data type	Range	Description	Default value
pbi_uRequestModuleIO	Requested module I/O No.	Word [unsigned]/ bit string [16-bit]	03E0H to 03E3H, 03FFH	Specify the module of the access destination. • 03FFH: Own station, control CPU • 03E0H: Multiple CPU No.1 • 03E1H: Multiple CPU No.2 • 03E2H: Multiple CPU No.3 • 03E3H: Multiple CPU No.4	03FFH
pbi_uResendCountMax	Maximum number of resends	Word [unsigned]/ bit string [16-bit]	0 to 15	Specify the number of resends to be performed if the data transfer is not completed within the monitoring time specified by pbi_uMonitorTime (Arrival monitoring time). • 0 to 15	5
pbi_uMonitorTime	Arrival monitoring time	Word [unsigned]/ bit string [16-bit]	0, 1 to 32767	Specify the monitoring time until completion of processing. If the processing is not completed within the monitoring time, data is resent until the value specified in pbi_uResendCountMax (Maximum number of resends) is reached. • 0: 10s • 1 to 32767: 1 to 32767s	0
pbi_bEncryptionComm	Encrypted communication enable/disable	Bit string	On or off	Specify whether SLMP frames are encrypted or not. • On: Enable (Encrypt) • Off: Disable (Do not encrypt)	Off
pbi_uLocal_Port_No	Own node port number	Word [unsigned]/ bit string [16-bit]	1 to 4999, 5010 to 65534	Specify the port number of the own node. Port numbers 1 to 1023 are generally reserved port numbers (WELL KNOWN PORT NUMBERS), and therefore port numbers 1024 to 4999 and 5010 to 65534 should be used. The specified value is valid only when pbi_bEncryptionComm (whether to use the encrypted communications) is on.	4096

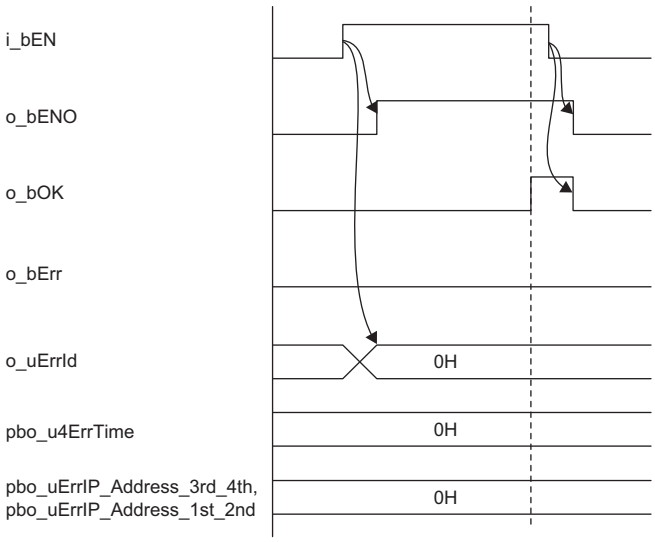
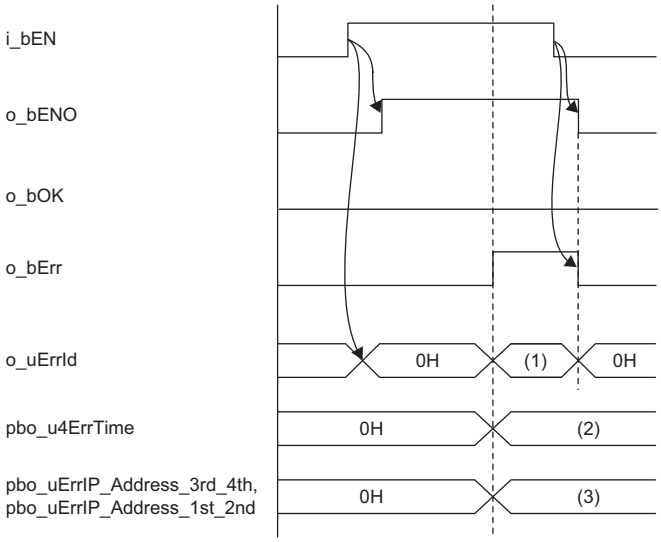
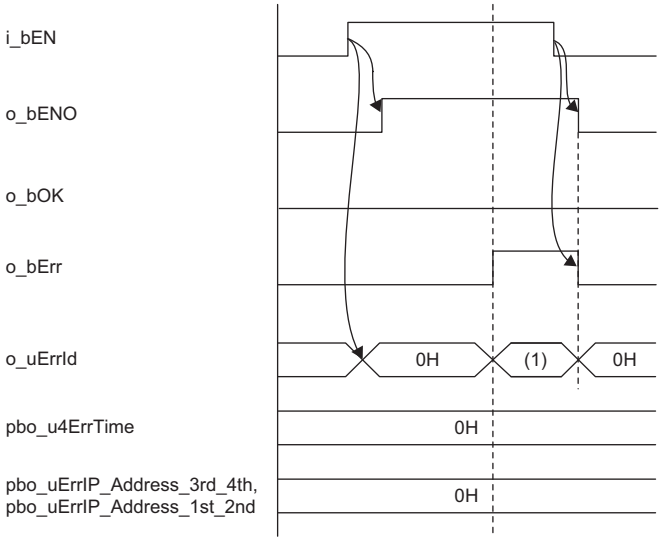
■Public variables

Variable name	Name	Data type	Description	Default value
pbo_uResendCount	Number of resends	Word [unsigned]/ bit string [16-bit]	The number of resends performed (result) is stored. The number of resends performed (result) between error detection and resend stop is stored.	0
pbo_u4ErrTime	Error occurrence time	Word [unsigned]/ bit string [16-bit] (0..3)	Clock data at the time of error occurrence is stored. 1st word • Upper 8 bits: Month (01H to 12H) • Lower 8 bits: Lower 2 digits of year (00H to 99H) 2nd word • Upper 8 bits: Hour (00H to 23H) • Lower 8 bits: Day (01H to 31H) 3rd word • Upper 8 bits: Second (00H to 59H) • Lower 8 bits: Minute (00H to 59H) 4th word • Upper 8 bits: Upper 2 digits of year (00H to 99H) • Lower 8 bits: Day of week (00H (Sunday) to 06H (Saturday))	0
pbo_uErrIP_Address_3rd_4th	IP address of error detected station (third and fourth octets)	Word [unsigned]/ bit string [16-bit] (0..1)	The IP addresses (the third and fourth octets) of the station in which an error was detected are stored.*1 Example: 0102H when the IP address is 192.168.1.2	0
pbo_uErrIP_Address_1st_2nd	IP address of error detected station (first and second octets)	Word [unsigned]/ bit string [16-bit] (0..1)	The IP addresses (the first and second octets) of the station in which an error was detected are stored.*1 Example: C0A8H when the IP address is 192.168.1.2	0

*1 The value is stored only when the dedicated instruction was completed with an error.
The value set to the target station address of the input argument is stored in the error-detected device IP addresses (the third and fourth octets) and the error-detected device IP addresses (the first and second octets).

Processing details

Item	Description	
Available device	Target module	MELSEC MX Controller (Ethernet port)
	CPU module	MELSEC MX Controller
	Engineering tool	GX Works3
Language	Ladder diagram	
Usage of labels	Label: 1096 points Latched label: 0 points The usage of labels embedded in a program depends on the controller used, the devices specified for arguments, and the option settings of GX Works3. For the option settings of GX Works3, refer to the following. GX Works3 Operating Manual	
Processing	• When i_bEN (execution command) is turned on, this function reads device data from the SLMP-compatible device. • Execute this FB while the IP address for an external device is specified. • This FB uses Read command (command: 0401) of the SLMP. The message of the SLMP command is a binary code. (SLMP Reference Manual) • When pbi_bEncryptionComm (whether to use the encrypted communications) is turned on and this FB is executed, the port number specified in pbi_uLocal_Port_No (own node port number) is used for communication by SLMP frame encrypted with DTLS.	
FB compilation method	Macro type	
FB operation	Pulse type (multiple-scan execution type)	

Item	Description
Timing chart of I/O signals	• For normal completion  • For error completion (When the dedicated instruction was completed with an error)  (1) Error code (2) Error occurrence time (3) Error-detected device IP address • For error completion (When the dedicated instruction was completed successfully but the end code indicates an error)  (1) Error code

Item	Description
Precautions	• This FB does not include error recovery processing. Please create error recovery processing separately according to the system and required operations. • This FB uses the SLMPSND instruction or SLMPSNDC instruction. The SLMPSND instruction and SLMPSNDC instruction are successfully completed even if the target device returns an abnormal response. Therefore, in this FB, the instruction is determined to be completed successfully or completed with an error by the end code of the response frame in addition to the completion status of the dedicated instruction. If the instruction is determined as completed with an error by the end code, check that the end code is stored in o_uErrId (error code), and 0 (initial value) is stored in pbo_u4ErrTime (error occurrence time), pbo_uErrIP_Address_3rd_4th (error detection device IP address (3rd and 4th octets)) and pbo_uErrIP_Address_1st_2nd (error detection device IP address (1st and 2nd octets)). For the cause of the error, refer the manual for the SLMP-compatible device being used, and take corrective action. (📖SLMP Reference Manual) • Turn off i_bEN (execution command) after o_bOK (normal completion) or o_bErr (error completion) is turned on. By turning off i_bEN (execution command), o_bOK (normal completion) or o_bErr (error completion) is turned off and o_uErrId (error code) is cleared to 0. • In this FB, access devices (such as link direct device) that are accessed by the extension specification of the SLMP cannot be read. • In this FB, stations in other network cannot be set as the target station. • When this FB is executed for the port of an external device where the remote password is set, execute this FB after performing the unlock processing of the remote password. When this FB is executed for the port of an external device where the remote password is set, an error will occur. • The target station must support "Read (command: 0401)" of the SLMP command. • This FB is for communications in binary code only. (Communications using ASCII code cannot be performed.) • This FB uses UDP or DTLS communications. The external device must have the same settings. • This FB uses the label initial value by each program. When the program file using this FB is specified to boot file setting for the boot operation in the controller, specify the initial label value file by each program to the boot file setting as well. If an error code that is not described in Page 1964 Error codes appears, the initial label value files by each program may not be set correctly. In this case, specify the initial label value files by each program to the boot file setting. • When pbi_bEncryptionComm (whether to use the encrypted communications) is on (enabled), the value set to pbi_uLocal_Port_No (own node port number) must not be the same as the value set to "PLC Port No." set in "External Device Configuration" in the module parameter (Port 3: Ethernet). If the configured port numbers are overlapped, this FB may complete with an error. • When this FB is executed, an operation error may occur in the controller. In this case, check the detailed information of the operation error in event history and correct the input argument of the module FB.

■Error codes

Error code	Description	Action
100H	A value out of the range is set in the number of device points (i_uDevicePoints) of the argument.	Set the value within the setting range in the number of device points (i_uDevicePoints).
Except 100H	Check the error code of the SLMPSND instruction and take actions. ☞ Page 1147 G(P).SLMPSND, J(P).SLMPSND, SP.SLMPSND For encrypted communication, check the error code of the SLMPSNDC instruction, and take actions. ☞ Page 1156 G(P).SLMPSNDC	

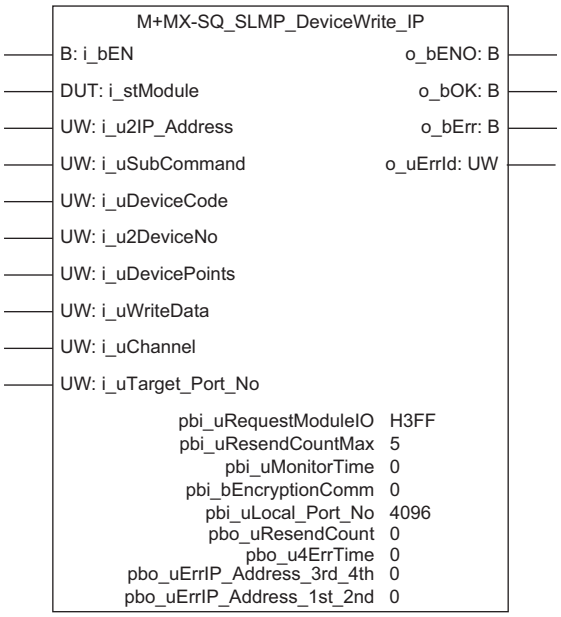
M+MX-SQ_SLMP_DeviceWrite_IP

Writes device data to the SLMP-compatible device specified by an IP address.

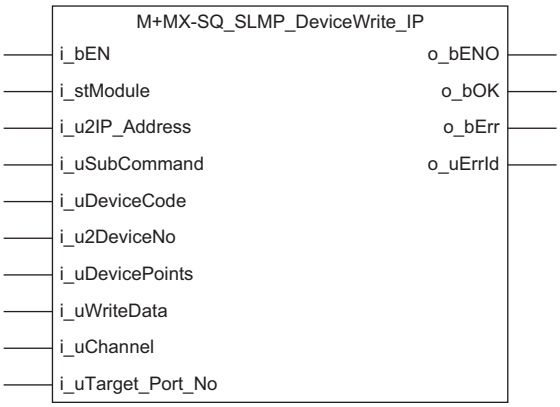
The external device must support SLMP command (Device Write).

For encrypted communications, the external device must support the encrypted predefined protocol (DTLS) in addition to the above.

Ladder



FBD/LD

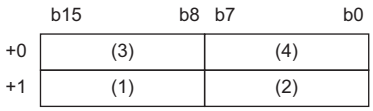


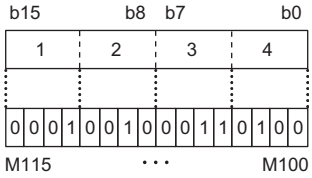
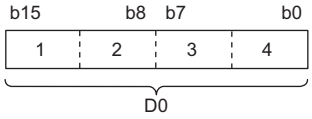
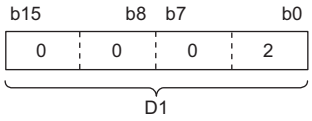
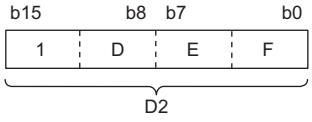
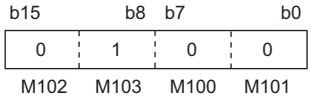
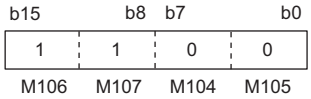
ST

```
M_MX_SQ_SLMP_DeviceWrite_IP(
  i_bEN:= ?BOOL? ,
  i_stModule:= ?M+MX_SQ? ,
  i_u2IP_Address:= ?WORD(0..1)? ,
  i_uSubCommand:= ?WORD? ,
  i_uDeviceCode:= ?WORD? ,
  i_u2DeviceNo:= ?WORD(0..1)? ,
  i_uDevicePoints:= ?WORD? ,
  i_uWriteData:= ?WORD? ,
  i_uChannel:= ?WORD? ,
  i_uTarget_Port_No:= ?WORD? ,
  o_bENO=> ?BOOL? ,
  o_bOK=> ?BOOL? ,
  o_bErr=> ?BOOL? ,
  o_uErrId=> ?WORD?
);
```

Setting data

■Input arguments

Variable name	Name	Data type	Range	Description
i_bEN	Execution command	Bit	—	On: Start FB. Off: Do not start FB.
i_stModule	Module label	Structure	—	Specify the module label of the controller executing the FB. (Example: MX_SQ)
i_u2IP_Address	IP address of external device	Word [unsigned]/ bit string [16-bit] (0..1)	0.0.0.1 to 253.255.255.254 (00000001H to DFFFFFFEH)	Specify the IP address of the target station. Specify the third and fourth octets to the 1st word, and first and second octets to the 2nd word. Note that the fourth octet cannot be set to 0 or 255 (FFH).  (1) to (4): IP address octet
i_uSubCommand	Sub command	Word [unsigned]/ bit string [16-bit]	—	Specify the write unit and specification method of a device. ■0th bit: Write unit 0: In units of words 1: In units of bits ■Specification method of the device is written 0: Specify the device code in 2 digits and the start device number in 6 digits (for MELSEC-Q/L series). 1: Specify the device code in 4 digits and the start device number in 8 digits (for MELSEC iQ-R series).
i_uDeviceCode	Device code[*1]	Word [unsigned]/ bit string [16-bit]	—	Specify the device code of the device to be written in binary code. • When the 1st bit of the subcommand is 0: 2 digits • When the 1st bit of the subcommand is 1: 4 digits
i_u2DeviceNo	Head device No.	Word [unsigned]/ bit string [16-bit] (0..1)	—	Specify the start device number of the device to be written in binary code. • When the 1st bit of the subcommand is 0: 6 digits • When the 1st bit of the subcommand is 1: 8 digits
i_uDevicePoints	Number of device points	Word [unsigned]/ bit string [16-bit]	—	Specify the number of device points of the device to be written in binary code. • When the 0th bit of the subcommand is 0: 1 to 960 • When the 0th bit of the subcommand is 1: 1 to 3972

Variable name	Name	Data type	Range	Description
i_uWriteData	Write data storage destination	Word [unsigned]/ bit string [16-bit]	—	Specify the start device number of the device for storing the write data. ■When the 0th bit of the subcommand is 0 The device data is written in units of words. Example: When writing the bit device M100 to M115 (one word) in units of words 1st word:  Example: When writing the word device D0 to D2 in units of words 1st word:  2nd word:  3rd word:  ■When the 0th bit of the subcommand is 1 The device data is written in units of bits. Example: When writing the bit device M100 to M107 in units of bits 1st word:  2nd word: 
i_uChannel	Own station channel	Word [unsigned]/ bit string [16-bit]	1 to 9	Specify the channel to be used by the own station. Since whether or not a serial number*2 is given to the request message depends on the channel, specify the channel as follows according to the application. • 1: No serial number is given (3E frame used) • 2 to 9: Serial number is given (4E frame used)
i_uTarget_Port_No	Destination port number	Word [unsigned]/ bit string [16-bit]	1 to 65534	Specify the UDP port number or DTLS port number of an external device.

*1 For details on each device code, refer to the following.
SLMP Reference Manual

*2 Give the serial numbers when sending several request messages to the same SLMP-compatible device. Serial numbers to be given are automatically numbered by the system. For the serial number, refer to the following.
SLMP Reference Manual

■Output arguments

Variable name	Name	Data type	Description	Default value
o_bENO	Execution status	Bit	On: The execution command is turned on. Off: The execution command is turned off.	Off
o_bOK	Normal completion	Bit	The FB has been processed normally when this argument is on.	Off
o_bErr	Error completion	Bit	The FB has been processed abnormally when this argument is on.	Off
o_uErrId	Error code	Word [unsigned]/bit string [16-bit]	An error code is stored at error completion.	0

■Operation parameters

Variable name	Name	Data type	Range	Description	Default value
pbi_uRequestModuleIO	Requested module I/O No.	Word [unsigned]/bit string [16-bit]	03E0H to 03E3H, 03FFH	Specify the module of the access destination. • 03FFH: Own station, control CPU • 03E0H: Multiple CPU No.1 • 03E1H: Multiple CPU No.2 • 03E2H: Multiple CPU No.3 • 03E3H: Multiple CPU No.4	03FFH
pbi_uResendCountMax	Maximum number of resends	Word [unsigned]/bit string [16-bit]	0 to 15	Specify the number of resends to be performed if the data transfer is not completed within the monitoring time specified by pbi_uMonitorTime (Arrival monitoring time). • 0 to 15	5
pbi_uMonitorTime	Arrival monitoring time	Word [unsigned]/bit string [16-bit]	0, 1 to 32767	Specify the monitoring time until completion of processing. If the processing is not completed within the monitoring time, data is resent until the value specified in pbi_uResendCountMax (Maximum number of resends) is reached. • 0: 10s • 1 to 32767: 1 to 32767s	0
pbi_bEncryptionComm	Encrypted communication enable/disable	Bit string	On or off	Specify whether SLMP frames are encrypted or not. • On: Enable (Encrypt) • Off: Disable (Do not encrypt)	Off
pbi_uLocal_Port_No	Own node port number	Word [unsigned]/bit string [16-bit]	1 to 4999, 5010 to 65534	Specify the port number of the own node. Port numbers 1 to 1023 are generally reserved port numbers (WELL KNOWN PORT NUMBERS), and therefore port numbers 1024 to 4999 and 5010 to 65534 should be used. The specified value is valid only when pbi_bEncryptionComm (whether to use the encrypted communications) is on.	4096

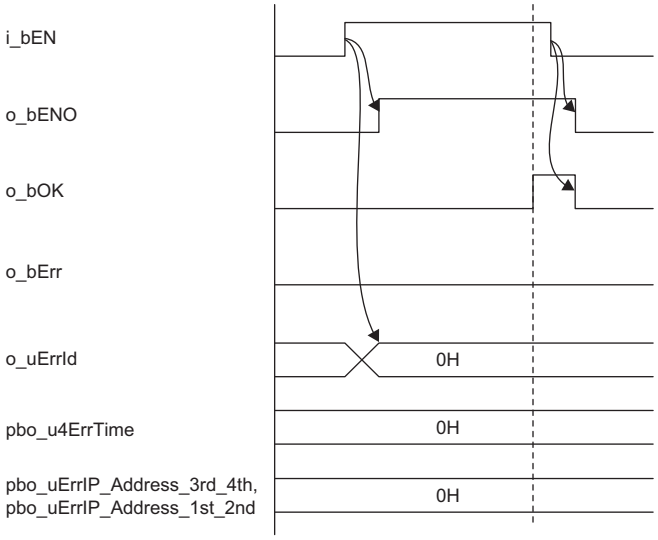
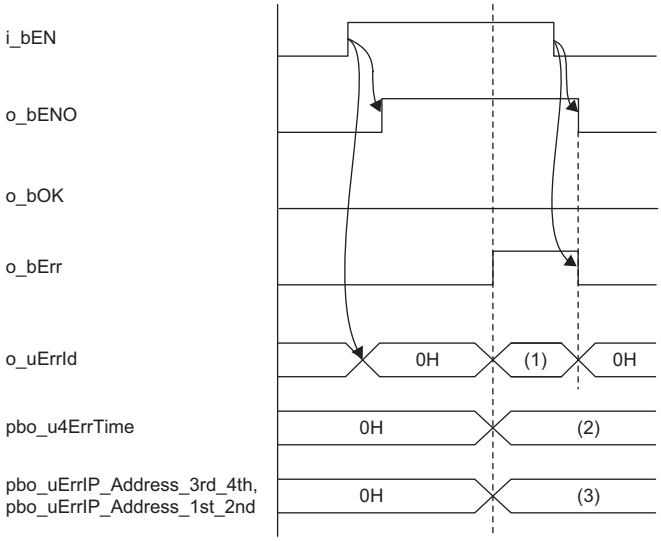
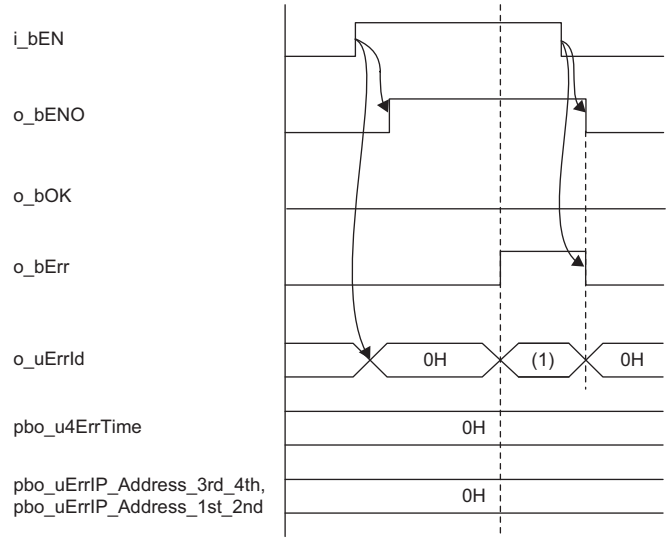
■Public variables

Variable name	Name	Data type	Description	Default value
pbo_uResendCount	Number of resends	Word [unsigned]/bit string [16-bit]	The number of resends performed (result) is stored.	0
pbo_u4ErrTime	Error occurrence time	Word [unsigned]/bit string [16-bit] (0..3)	Clock data at the time of error occurrence is stored. 1st word • Upper 8 bits: Month (01H to 12H) • Lower 8 bits: Lower 2 digits of year (00H to 99H) 2nd word • Upper 8 bits: Hour (00H to 23H) • Lower 8 bits: Day (01H to 31H) 3rd word • Upper 8 bits: Second (00H to 59H) • Lower 8 bits: Minute (00H to 59H) 4th word • Upper 8 bits: Upper 2 digits of year (00H to 99H) • Lower 8 bits: Day of week (00H (Sunday) to 06H (Saturday))	0
pbo_uErrIP_Address_3rd_4th	IP address of error detected station (third and fourth octets)	Word [unsigned]/bit string [16-bit] (0..1)	The IP addresses (the third and fourth octets) of the station in which an error was detected are stored.*1 Example: 0102H when the IP address is 192.168.1.2 0102H	0
pbo_uErrIP_Address_1st_2nd	IP address of error detected station (first and second octets)	Word [unsigned]/bit string [16-bit] (0..1)	The IP addresses (the first and second octets) of the station in which an error was detected are stored.*1 Example: C0A8H when the IP address is 192.168.1.2 C0A8H	0

*1 The value is stored only when the dedicated instruction was completed with an error.
The value set to the target station address of the input argument is stored in the error-detected device IP addresses (the third and fourth octets) and the error-detected device IP addresses (the first and second octets).

Processing details

Item	Description	
Available device	Target module	MELSEC MX Controller (Ethernet port)
	CPU module	MELSEC MX Controller
	Engineering tool	GX Works3
Language	Ladder diagram	
Usage of labels	Label: 1096 points Latched label: 0 points The usage of labels embedded in a program depends on the controller used, the devices specified for arguments, and the option settings of GX Works3. For the option settings of GX Works3, refer to the following. 📖GX Works3 Operating Manual	
Processing	• When i_bEN (execution command) is turned on, this function writes device data of the SLMP-compatible device. • Execute this FB while the IP address for an external device is specified. • This FB uses Write command (command: 1401) of the SLMP. The message of the SLMP command is a binary code. (📖SLMP Reference Manual) • When pbi_bEncryptionComm (whether to use the encrypted communications) is turned on and this FB is executed, the port number specified in pbi_uLocal_Port_No (own node port number) is used for communication by SLMP frame encrypted with DTLS.	
FB compilation method	Macro type	
FB operation	Pulse type (multiple-scan execution type)	

Item	Description
Timing chart of I/O signals	• For normal completion  • For error completion (When the dedicated instruction was completed with an error)  (1) Error code (2) Error occurrence time (3) Error-detected device IP address • For error completion (When the dedicated instruction was completed successfully but the end code indicates an error)  (1) Error code

Item	Description
Precautions	• This FB does not include error recovery processing. Please create error recovery processing separately according to the system and required operations. • This FB uses the SLMPSND instruction or SLMPSNDC instruction. The SLMPSND instruction and SLMPSNDC instruction are successfully completed even if the target device returns an abnormal response. Therefore, in this FB, the instruction is determined to be completed successfully or completed with an error by the end code of the response frame in addition to the completion status of the dedicated instruction. If the instruction is determined as completed with an error by the end code, check that the end code is stored in o_uErrId (error code), and 0 (initial value) is stored in pbo_u4ErrTime (error occurrence time), pbo_uErrIP_Address_3rd_4th (error detection device IP address (3rd and 4th octets)) and pbo_uErrIP_Address_1st_2nd (error detection device IP address (1st and 2nd octets)). For the cause of the error, refer the manual for the SLMP-compatible device being used, and take corrective action. (📖SLMP Reference Manual) • Turn off i_bEN (execution command) after o_bOK (normal completion) or o_bErr (error completion) is turned on. By turning off i_bEN (execution command), o_bOK (normal completion) or o_bErr (error completion) is turned off and o_uErrId (error code) is cleared to 0. • In this FB, access devices (such as link direct device) that are accessed by the extension specification of the SLMP cannot be written. • In this FB, stations in other network cannot be set as the target station. • When this FB is executed for the port of an external device where the remote password is set, execute this FB after performing the unlock processing of the remote password. When this FB is executed for the port of an external device where the remote password is set, an error will occur. • The target station must support "Write (command: 1401)" of the SLMP command. • This FB is for communications in binary code only. (Communications using ASCII code cannot be performed.) • This FB uses UDP or DTLS communications. The external device must have the same settings. • This FB uses the label initial value by each program. When the program file using this FB is specified to boot file setting for the boot operation in the controller, specify the initial label value file by each program to the boot file setting as well. If an error code that is not described in Page 1971 Error codes appears, the initial label value files by each program may not be set correctly. In this case, specify the initial label value files by each program to the boot file setting. • When pbi_bEncryptionComm (whether to use the encrypted communications) is on (enabled), the value set to pbi_uLocal_Port_No (own node port number) must not be the same as the value set to "PLC Port No." set in "External Device Configuration" in the module parameter (Port 3: Ethernet). If the configured port numbers are overlapped, this FB may complete with an error. • When this FB is executed, an operation error may occur in the controller. In this case, check the detailed information of the operation error in event history and correct the input argument of the module FB.

■Error codes

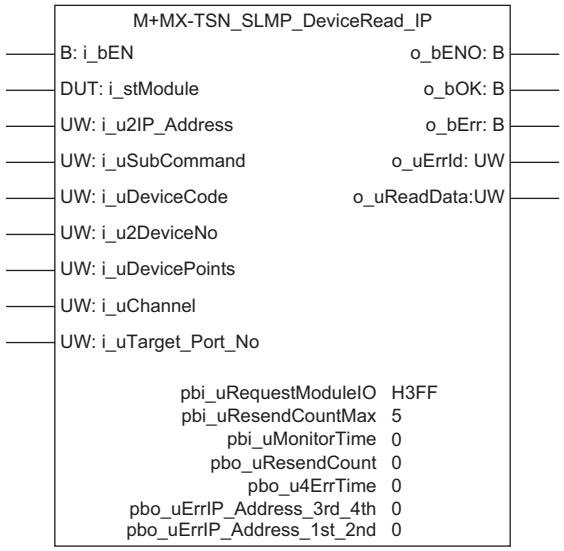
Error code	Description	Action
100H	A value out of the range is set in the number of device points (i_uDevicePoints) of the argument.	Set the value within the setting range in the number of device points (i_uDevicePoints).
Except 100H	Check the error code of the SLMPSND instruction and take actions. ☞ Page 1147 G(P).SLMPSND, J(P).SLMPSND, SP.SLMPSND For encrypted communication, check the error code of the SLMPSNDC instruction, and take actions. ☞ Page 1156 G(P).SLMPSNDC	

M+MX-TSN_SLMP_DeviceRead_IP

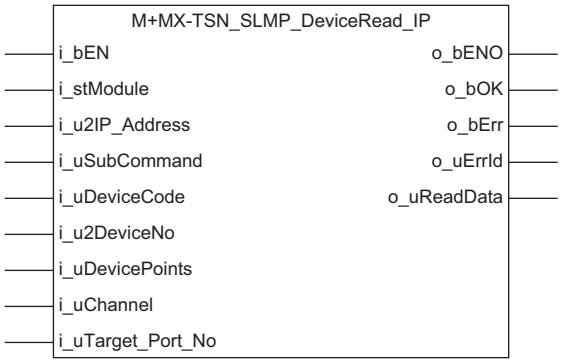
Reads device data from the SLMP-compatible device specified by an IP address.

The external device must support SLMP command (Device Read).

Ladder



FBD/LD



ST

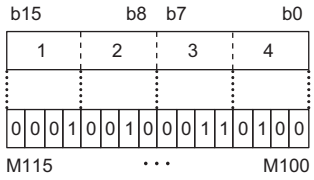
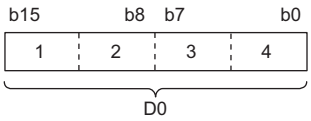
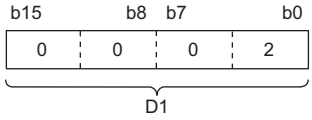
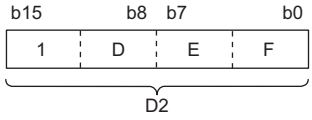
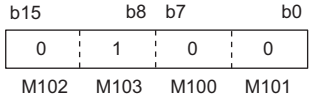
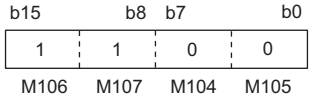
```
M_MX_TSN_SLMP_DeviceRead_IP(
  i_bEN:= ?BOOL? ,
  i_stModule:= ?M+MX_TSN? ,
  i_u2IP_Address:= ?WORD(0..1)? ,
  i_uSubCommand:= ?WORD? ,
  i_uDeviceCode:= ?WORD? ,
  i_u2DeviceNo:= ?WORD(0..1)? ,
  i_uDevicePoints:= ?WORD? ,
  i_uChannel:= ?WORD? ,
  i_uTarget_Port_No:= ?WORD? ,
  o_bENO=> ?BOOL? ,
  o_bOK=> ?BOOL? ,
  o_bErr=> ?BOOL? ,
  o_uErrId=> ?WORD? ,
  o_uReadData=> ?WORD?
);
```

■Input arguments

Variable name	Name	Data type	Range	Description
i_bEN	Execution command	Bit	—	On: Start FB. Off: Do not start FB.
i_stModule	Module label	Structure	—	Specify the module label of the controller executing the FB. (Example: MX_TSN)
i_u2IP_Address	IP address of external device	Word [unsigned]/bit string [16-bit] (0..1)	0.0.0.1 to 253.255.255.254 (00000001H to DFFFFFFEH)	Specify the IP address of an external device. Specify the third and fourth octets to the 1st word, and first and second octets to the 2nd word. Note that the fourth octet cannot be set to 0 or 255 (FFH). b15 b8 b7 b0 +0 (3) (4) +1 (1) (2) (1) to (4): IP address octet
i_uSubCommand	Sub command	Word [unsigned]/bit string [16-bit]	—	Specify the read unit and specification method of a device. ■0th bit: Read unit 0: In units of words 1: In units of bits ■1st bit: Specification method of the device is read 0: Specify the device code in 2 digits and the start device number in 6 digits (for MELSEC-Q/L series). 1: Specify the device code in 4 digits and the start device number in 8 digits (for MELSEC iQ-R series).
i_uDeviceCode	Device code*1	Word [unsigned]/bit string [16-bit]	—	Specify the device code of the device to be read in binary code. • When the 1st bit of the subcommand is 0: 2 digits • When the 1st bit of the subcommand is 1: 4 digits
i_u2DeviceNo	Head device No.	Word [unsigned]/bit string [16-bit] (0..1)	—	Specify the start device number of the device to be read in binary code. • When the 1st bit of the subcommand is 0: 6 digits • When the 1st bit of the subcommand is 1: 8 digits
i_uDevicePoints	Number of device points	Word [unsigned]/bit string [16-bit]	—	Specify the number of device points of the device to be read in binary code. • When the 0th bit of the subcommand is 0: 1 to 960 • When the 0th bit of the subcommand is 1: 1 to 3972
i_uChannel	Own station channel	Word [unsigned]/bit string [16-bit]	1 to 9	Specify the channel to be used by the own station. • 1: No serial number is given (3E frame used) • 2 to 9: Serial number is given (4E frame used) • 10 to 17: Station number extension frame
i_uTarget_Port_No	Destination port number	Word [unsigned]/bit string [16-bit]	1 to 65534	Specify the UDP port number of an external device.

*1 For details on each device code, refer to the following.
SLMP Reference Manual

■Output arguments

Variable name	Name	Data type	Description	Default value
o_bENO	Execution status	Bit	On: The execution command is turned on. Off: The execution command is turned off.	Off
o_bOK	Normal completion	Bit	The FB has been processed normally when this argument is on.	Off
o_bErr	Error completion	Bit	The FB has been processed abnormally when this argument is on.	Off
o_uErrId	Error code	Word [unsigned]/bit string [16-bit]	An error code is stored at error completion.	0
o_uReadData	Read data storage destination	Word [unsigned]/bit string [16-bit]	Specify the start device number of the device for storing the read data. The read data is stored in binary code. ■When the 0th bit of the subcommand is 0 The device data is read in units of words. Example: When reading the bit device M100 to M115 (one word) in units of words 1st word:  Example: When reading the word device D0 to D2 in units of words 1st word:  2nd word:  3rd word:  ■When the 0th bit of the subcommand is 1 The device data is read in units of bits. Example: When reading the bit device M100 to M107 in units of bits 1st word:  2nd word:  	0

■Operation parameters

Variable name	Name	Data type	Range	Description	Default value
pbi_uRequestModuleIO	Requested module I/O No.	Word [unsigned]/bit string [16-bit]	03E0H to 03E3H, 03FFH	Specify the module of the access destination. • 03FFH: Own station, control CPU • 03E0H: Multiple CPU No.1 • 03E1H: Multiple CPU No.2 • 03E2H: Multiple CPU No.3 • 03E3H: Multiple CPU No.4	03FFH
pbi_uResendCountMax	Maximum number of resends	Word [unsigned]/bit string [16-bit]	0 to 15	Specify the number of resends to be performed if the data transfer is not completed within the monitoring time specified by pbi_uMonitorTime (Arrival monitoring time). • 0 to 15	5
pbi_uMonitorTime	Arrival monitoring time	Word [unsigned]/bit string [16-bit]	0, 1 to 32767	Specify the monitoring time until completion of processing. If the processing is not completed within the monitoring time, data is resent until the value specified in pbi_uResendCountMax (Maximum number of resends) is reached. • 0: 10s • 1 to 32767: 1 to 32767s	0

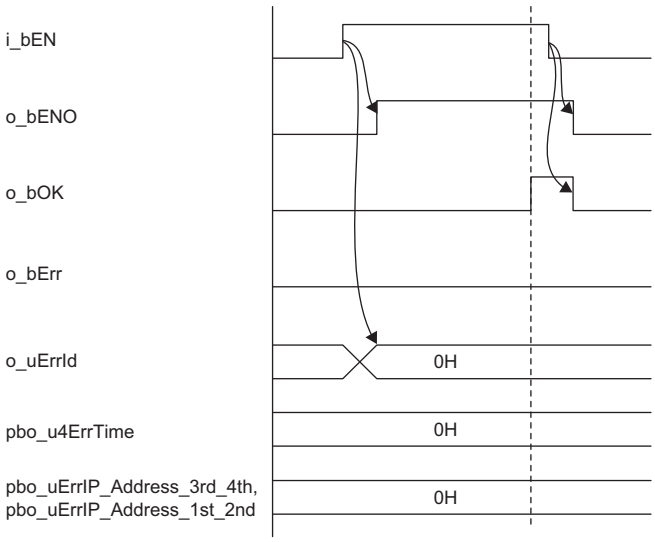
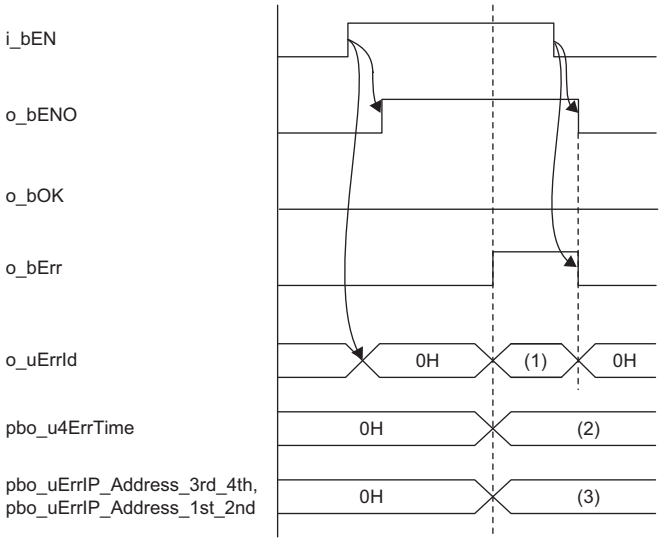
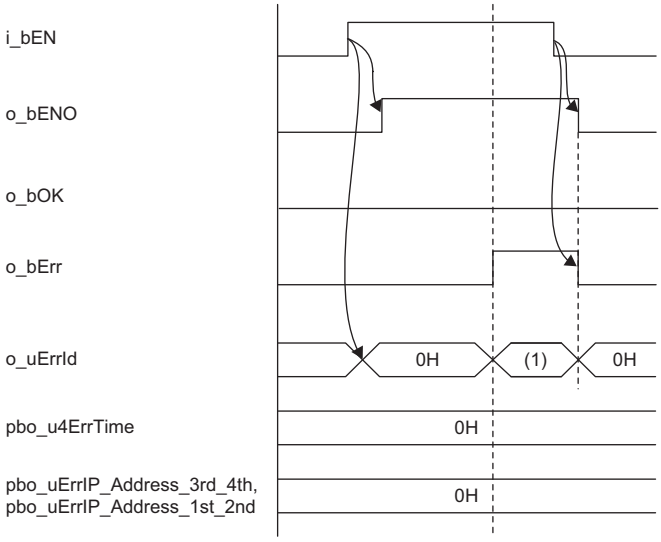
■Public variables

Variable name	Name	Data type	Description	Default value
pbo_uResendCount	Number of resends	Word [unsigned]/bit string [16-bit]	The number of resends performed (result) is stored. The number of resends performed (result) between error detection and resend stop is stored.	0
pbo_u4ErrTime	Error occurrence time	Word [unsigned]/bit string [16-bit] (0..3)	Clock data at the time of error occurrence is stored. 1st word • Upper 8 bits: Month (01H to 12H) • Lower 8 bits: Lower 2 digits of year (00H to 99H) 2nd word • Upper 8 bits: Hour (00H to 23H) • Lower 8 bits: Day (01H to 31H) 3rd word • Upper 8 bits: Second (00H to 59H) • Lower 8 bits: Minute (00H to 59H) 4th word • Upper 8 bits: Upper 2 digits of year (00H to 99H) • Lower 8 bits: Day of week (00H (Sunday) to 06H (Saturday))	0
pbo_uErrIP_Address_3rd_4th	IP address of error detected station (third and fourth octets)	Word [unsigned]/bit string [16-bit] (0..1)	The IP addresses (the third and fourth octets) of the station in which an error was detected are stored.*1 Example: 0102H when the IP address is 192.168.1.2 0102H	0
pbo_uErrIP_Address_1st_2nd	IP address of error detected station (first and second octets)	Word [unsigned]/bit string [16-bit] (0..1)	The IP addresses (the first and second octets) of the station in which an error was detected are stored.*1 Example: C0A8H when the IP address is 192.168.1.2 C0A8H	0

*1 The value is stored only when the dedicated instruction was completed with an error.
The value set to the target station address of the input argument is stored in the error-detected device IP addresses (the third and fourth octets) and the error-detected device IP addresses (the first and second octets).

Processing details

Item	Description	
Available device	Target module	MELSEC MX Controller (port for CC-Link IE TSN)
	CPU module	MELSEC MX Controller
	Engineering tool	GX Works3
Language	Ladder diagram	
Usage of labels	Label: 1076 points Latched label: 0 points The usage of labels embedded in a program depends on the controller used, the devices specified for arguments, and the option settings of GX Works3. For the option settings of GX Works3, refer to the following. GX Works3 Operating Manual	
Processing	• When i_bEN (execution command) is turned on, this function reads device data from the SLMP-compatible device. • Execute this FB while the IP address for an external device is specified. • This FB uses Read command (command: 0401) of the SLMP. The message of the SLMP command is a binary code. (SLMP Reference Manual)	
FB compilation method	Macro type	
FB operation	Pulse type (multiple-scan execution type)	

Item	Description
Timing chart of I/O signals	• For normal completion  • For error completion (When the dedicated instruction was completed with an error)  (1) Error code (2) Error occurrence time (3) Error-detected device IP address • For error completion (When the dedicated instruction was completed successfully but the end code indicates an error)  (1) Error code

Item	Description
Precautions	• This FB does not include error recovery processing. Please create error recovery processing separately according to the system and required operations. • This FB uses the SLMPSND instruction. The SLMPSND instruction is successfully completed even if the target device returns an abnormal response. Therefore, in this FB, the instruction is determined to be completed successfully or completed with an error by the end code of the response frame in addition to the completion status of the dedicated instruction. If the instruction is determined as completed with an error by the end code, check that the end code is stored in o_uErrId (error code), and 0 (initial value) is stored in pbo_u4ErrTime (error occurrence time), pbo_uErrIP_Address_3rd_4th (error detection device IP address (3rd and 4th octets)) and pbo_uErrIP_Address_1st_2nd (error detection device IP address (1st and 2nd octets)). For the cause of the error, refer the manual for the SLMP-compatible device being used, and take corrective action. (📖SLMP Reference Manual) • Turn off i_bEN (execution command) after o_bOK (normal completion) or o_bErr (error completion) is turned on. By turning off i_bEN (execution command), o_bOK (normal completion) or o_bErr (error completion) is turned off and o_uErrId (error code) is cleared to 0. • In this FB, access devices (such as link direct device) that are accessed by the extension specification of the SLMP cannot be read. • In this FB, stations in other network cannot be set as the target station. • When this FB is executed for the port of an external device where the remote password is set, execute this FB after performing the unlock processing of the remote password. When this FB is executed for the port of an external device where the remote password is set, an error will occur. • The target station must support "Read (command: 0401)" of the SLMP command. • This FB is for communications in binary code only. (Communications using ASCII code cannot be performed.) • This FB uses UDP communications. The external device must have the same settings. • This FB uses the label initial value by each program. When the program file using this FB is specified to boot file setting for the boot operation in the controller, specify the initial label value file by each program to the boot file setting as well. If an error code that is not described in Page 1978 Error codes appears, the initial label value files by each program may not be set correctly. In this case, specify the initial label value files by each program to the boot file setting. • When this FB is executed, an operation error may occur in the controller. In this case, check the detailed information of the operation error in event history and correct the input argument of the module FB.

■Error codes

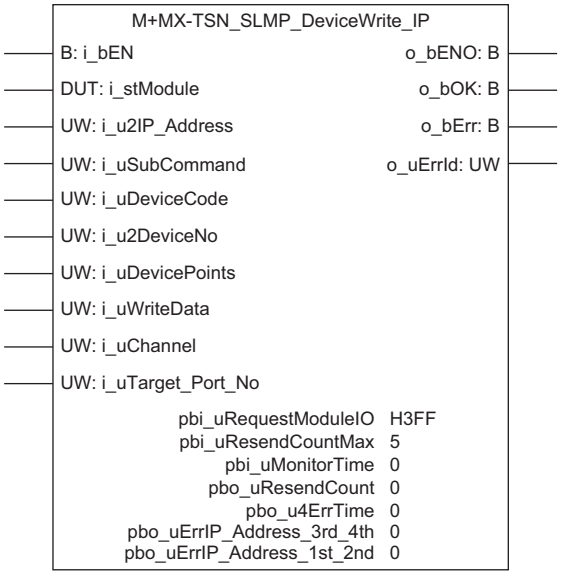
Error code	Description	Action
100H	A value out of the range is set in the number of device points (i_uDevicePoints) of the argument.	Set the value within the setting range in the number of device points (i_uDevicePoints).
Except 100H	Check the error code of the SLMPSND instruction and take actions. ☞ Page 1147 G(P).SLMPSND, J(P).SLMPSND, SP.SLMPSND	

M+MX-TSN_SLMP_DeviceWrite_IP

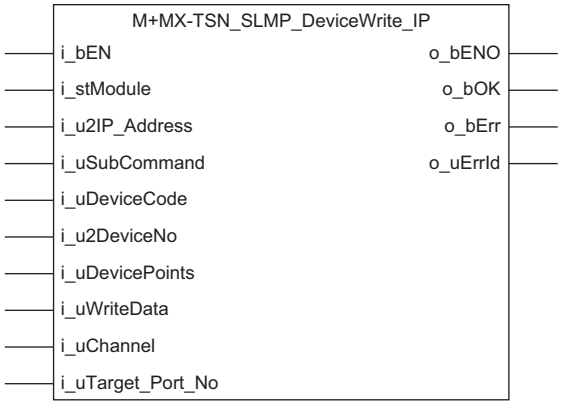
Writes device data to the SLMP-compatible device specified by an IP address.

The external device must support SLMP command (Device Write).

Ladder



FBD/LD



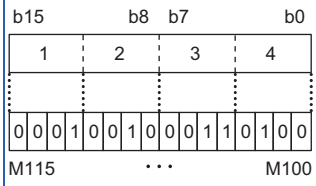
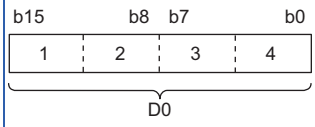
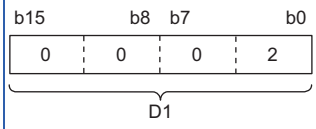
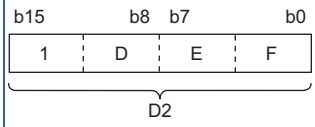
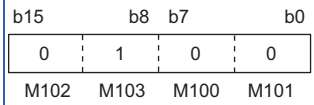
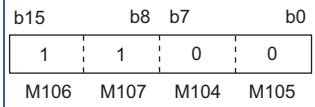
ST

```
M_MX_TSN_SLMP_DeviceWrite_IP(
    i_bEN:= ?BOOL? ,
    i_stModule:= ?M+MX_TSN? ,
    i_u2IP_Address:= ?WORD(0..1)? ,
    i_uSubCommand:= ?WORD? ,
    i_uDeviceCode:= ?WORD? ,
    i_u2DeviceNo:= ?WORD(0..1)? ,
    i_uDevicePoints:= ?WORD? ,
    i_uWriteData:= ?WORD? ,
    i_uChannel:= ?WORD? ,
    i_uTarget_Port_No:= ?WORD? ,
    o_bENO=> ?BOOL? ,
    o_bOK=> ?BOOL? ,
    o_bErr=> ?BOOL? ,
    o_uErrId=> ?WORD?
);
```

Setting data

■Input arguments

Variable name	Name	Data type	Range	Description
i_bEN	Execution command	Bit	—	On: Start FB. Off: Do not start FB.
i_stModule	Module label	Structure	—	Specify the module label of the controller executing the FB. (Example: MX_TSN)
i_u2IP_Address	IP address of external device	Word [unsigned]/bit string [16-bit] (0..1)	0.0.0.1 to 253.255.255.254 (00000001H to DFFFFFFEH)	Specify the IP address of the target station. Specify the third and fourth octets to the 1st word, and first and second octets to the 2nd word. Note that the fourth octet cannot be set to 0 or 255 (FFH). b15 b8 b7 b0 +0 (3) (4) +1 (1) (2) (1) to (4): IP address octet
i_uSubCommand	Sub command	Word [unsigned]/bit string [16-bit]	—	Specify the write unit and specification method of a device. ■0th bit: Write unit 0: In units of words 1: In units of bits ■Specification method of the device is written 0: Specify the device code in 2 digits and the start device number in 6 digits (for MELSEC-Q/L series). 1: Specify the device code in 4 digits and the start device number in 8 digits (for MELSEC iQ-R series).
i_uDeviceCode	Device code*1	Word [unsigned]/bit string [16-bit]	—	Specify the device code of the device to be written in binary code. • When the 1st bit of the subcommand is 0: 2 digits • When the 1st bit of the subcommand is 1: 4 digits
i_u2DeviceNo	Head device No.	Word [unsigned]/bit string [16-bit] (0..1)	—	Specify the start device number of the device to be written in binary code. • When the 1st bit of the subcommand is 0: 6 digits • When the 1st bit of the subcommand is 1: 8 digits
i_uDevicePoints	Number of device points	Word [unsigned]/bit string [16-bit]	—	Specify the number of device points of the device to be written in binary code. ■When the 0th bit of the subcommand is 0 • 1 to 960 ■When the 0th bit of the subcommand is 1: 1 to 3972 • Own station channel is 1 to 9: 1 to 960 • Own station channel is 10 to 17: 1 to 3960

Variable name	Name	Data type	Range	Description
i_uWriteData	Write data storage destination	Word [unsigned]/bit string [16-bit]	—	Specify the start device number of the device for storing the write data. ■When the 0th bit of the subcommand is 0 The device data is written in units of words. Example: When writing the bit device M100 to M115 (one word) in units of words 1st word:  Example: When writing the word device D0 to D2 in units of words 1st word:  2nd word:  3rd word:  ■When the 0th bit of the subcommand is 1 The device data is written in units of bits. Example: When writing the bit device M100 to M107 in units of bits 1st word:  2nd word: 
i_uChannel	Own station channel	Word [unsigned]/bit string [16-bit]	1 to 17	Specify the channel to be used by the own station. • 1: No serial number is given (3E frame used) • 2 to 9: Serial number is given (4E frame used) • 10 to 17: Station number extension frame
i_uTarget_Port_No	Destination port number	Word [unsigned]/bit string [16-bit]	1 to 65534	Specify the UDP port number of an external device.

*1 For details on each device code, refer to the following.
SLMP Reference Manual

■Output arguments

Variable name	Name	Data type	Description	Default value
o_bENO	Execution status	Bit	On: The execution command is turned on. Off: The execution command is turned off.	Off
o_bOK	Normal completion	Bit	The FB has been processed normally when this argument is on.	Off
o_bErr	Error completion	Bit	The FB has been processed abnormally when this argument is on.	Off
o_uErrId	Error code	Word [unsigned]/bit string [16-bit]	An error code is stored at error completion.	0

■Operation parameters

Variable name	Name	Data type	Range	Description	Default value
pbi_uRequestModuleIO	Requested module I/O No.	Word [unsigned]/bit string [16-bit]	03E0H to 03E3H, 03FFH	Specify the module of the access destination. • 03FFH: Own station, control CPU • 03E0H: Multiple CPU No.1 • 03E1H: Multiple CPU No.2 • 03E2H: Multiple CPU No.3 • 03E3H: Multiple CPU No.4	03FFH
pbi_uResendCountMax	Maximum number of resends	Word [unsigned]/bit string [16-bit]	0 to 15	Specify the number of resends to be performed if the data transfer is not completed within the monitoring time specified by pbi_uMonitorTime (Arrival monitoring time). • 0 to 15	5
pbi_uMonitorTime	Arrival monitoring time	Word [unsigned]/bit string [16-bit]	0, 1 to 32767	Specify the monitoring time until completion of processing. If the processing is not completed within the monitoring time, data is resent until the value specified in pbi_uResendCountMax (Maximum number of resends) is reached. • 0: 10s • 1 to 32767: 1 to 32767s	0

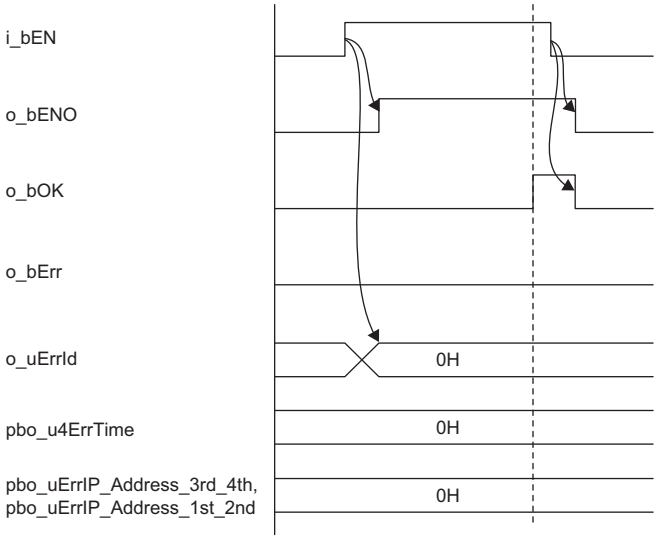
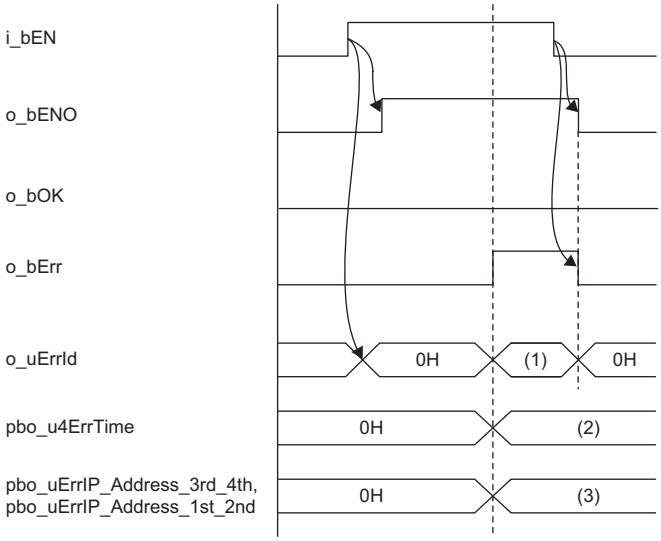
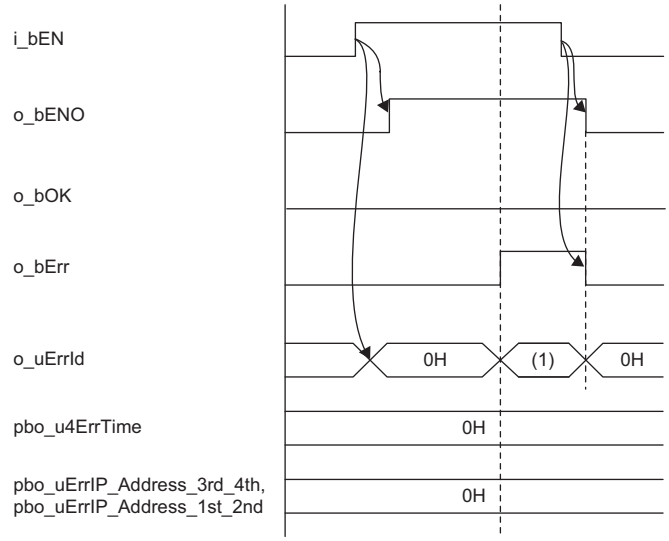
■Public variables

Variable name	Name	Data type	Description	Default value
pbo_uResendCount	Number of resends	Word [unsigned]/bit string [16-bit]	The number of resends performed (result) is stored.	0
pbo_u4ErrTime	Error occurrence time	Word [unsigned]/bit string [16-bit] (0..3)	Clock data at the time of error occurrence is stored. 1st word • Upper 8 bits: Month (01H to 12H) • Lower 8 bits: Lower 2 digits of year (00H to 99H) 2nd word • Upper 8 bits: Hour (00H to 23H) • Lower 8 bits: Day (01H to 31H) 3rd word • Upper 8 bits: Second (00H to 59H) • Lower 8 bits: Minute (00H to 59H) 4th word • Upper 8 bits: Upper 2 digits of year (00H to 99H) • Lower 8 bits: Day of week (00H (Sunday) to 06H (Saturday))	0
pbo_uErrIP_Address_3rd_4th	IP address of error detected station (third and fourth octets)	Word [unsigned]/bit string [16-bit] (0..1)	The IP addresses (the third and fourth octets) of the station in which an error was detected are stored.*1 Example: When the IP address is 192.168.1.2 0102H	0
pbo_uErrIP_Address_1st_2nd	IP address of error detected station (first and second octets)	Word [unsigned]/bit string [16-bit] (0..1)	The IP addresses (the first and second octets) of the station in which an error was detected are stored.*1 Example: When the IP address is 192.168.1.2 C0A8H	0

*1 The value is stored only when the dedicated instruction was completed with an error.
The value set to the target station address of the input argument is stored in the error-detected device IP addresses (the third and fourth octets) and the error-detected device IP addresses (the first and second octets).

Processing details

Item	Description	
Available device	Target module	MELSEC MX Controller (port for CC-Link IE TSN)
	CPU module	MELSEC MX Controller
	Engineering tool	GX Works3
Language	Ladder diagram	
Usage of labels	Label: 1076 points Latched label: 0 points The usage of labels embedded in a program depends on the controller used, the devices specified for arguments, and the option settings of GX Works3. For the option settings of GX Works3, refer to the following. GX Works3 Operating Manual	
Processing	• When i_bEN (execution command) is turned on, this function writes device data of the SLMP-compatible device. • Execute this FB while the IP address for an external device is specified. • This FB uses Write command (command: 1401) of the SLMP. The message of the SLMP command is a binary code. (SLMP Reference Manual)	
FB compilation method	Macro type	
FB operation	Pulse type (multiple-scan execution type)	

Item	Description
Timing chart of I/O signals	• For normal completion  • For error completion (When the dedicated instruction was completed with an error)  (1) Error code (2) Error occurrence time (3) Error-detected device IP address • For error completion (When the dedicated instruction was completed successfully but the end code indicates an error)  (1) Error code

Item	Description
Precautions	• This FB does not include error recovery processing. Please create error recovery processing separately according to the system and required operations. • This FB uses the SLMPSND instruction. The SLMPSND instruction is successfully completed even if the target device returns an abnormal response. Therefore, in this FB, the instruction is determined to be completed successfully or completed with an error by the end code of the response frame in addition to the completion status of the dedicated instruction. If the instruction is determined as completed with an error by the end code, check that the end code is stored in o_uErrId (error code), and 0 (initial value) is stored in pbo_u4ErrTime (error occurrence time), pbo_uErrIP_Address_3rd_4th (error detection device IP address (3rd and 4th octets)) and pbo_uErrIP_Address_1st_2nd (error detection device IP address (1st and 2nd octets)). For the cause of the error, refer the manual for the SLMP-compatible device being used, and take corrective action. (📖SLMP Reference Manual) • Turn off i_bEN (execution command) after o_bOK (normal completion) or o_bErr (error completion) is turned on. By turning off i_bEN (execution command), o_bOK (normal completion) or o_bErr (error completion) is turned off and o_uErrId (error code) is cleared to 0. • In this FB, access devices (such as link direct device) that are accessed by the extension specification of the SLMP cannot be written. • In this FB, stations in other network cannot be set as the target station. • When this FB is executed for the port of an external device where the remote password is set, execute this FB after performing the unlock processing of the remote password. When this FB is executed for the port of an external device where the remote password is set, an error will occur. • The target station must support "Write (command: 1401)" of the SLMP command. • This FB is for communications in binary code only. (Communications using ASCII code cannot be performed.) • This FB uses UDP communications. The external device must have the same settings. • This FB uses the label initial value by each program. When the program file using this FB is specified to boot file setting for the boot operation in the controller, specify the initial label value file by each program to the boot file setting as well. If an error code that is not described in Page 1985 Error codes appears, the initial label value files by each program may not be set correctly. In this case, specify the initial label value files by each program to the boot file setting. • When this FB is executed, an operation error may occur in the controller. In this case, check the detailed information of the operation error in event history and correct the input argument of the module FB.

■Error codes

Error code	Description	Action
100H	A value out of the range is set in the number of device points (i_uDevicePoints) of the argument.	Set the value within the setting range in the number of device points (i_uDevicePoints).
Except 100H	Check the error code of the SLMPSND instruction and take actions. ☞ Page 1147 G(P).SLMPSND, J(P).SLMPSND, SP.SLMPSND	

45 FUNCTION BLOCKS FOR CC-Link IE TSN

Point

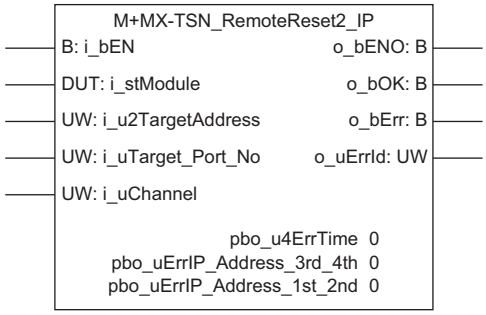
The function blocks in this chapter are controller module FBs. Use them from [Module] ⇨ [Module FB] of the Element Selection window of the engineering tool.

45.1 Remote Reset

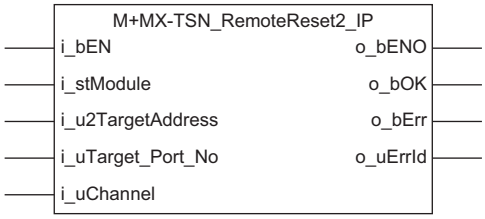
M+MX-TSN_RemoteReset2_IP

Sends the remote RESET of an SLMP request to the target station specified by an IP address.

Ladder



FBD/LD



ST

```
M_MX_TSN_RemoteReset2_IP(
   i_bEN:= ?BOOL? ,
   i_stModule:= ?M+MX_TSN? ,
   i_u2TargetAddress:= ?WORD(0..1)? ,
   i_uTarget_Port_No:= ?WORD? ,
   i_uChannel:= ?WORD? ,
   o_bENO=> ?BOOL? ,
   o_bOK=> ?BOOL? ,
   o_bErr=> ?BOOL?
   o_uErrId=> ?WORD?
);
```

Setting data

■Input arguments

Variable name	Name	Data type	Range	Description
i_bEN	Execution command	Bit	—	On: Start FB. Off: Do not start FB.
i_stModule	Module label	Structure	—	Specify the module label of the controller executing the FB. (Example: MX_TSN)
i_u2Target Address	IP address of external device	Word [unsigned]/bit string [16-bit] (0..1)	0.0.0.1 to 253.255.255.254 (00000001H to DFFFFFFEH)	Specify the IP address of an external device. When specifying the address using a label, use an array as the data type. • 00000001H to DFFFFFFEH Specify a value within the range of 1 to 254 (FEH) for the fourth octet. b15 b8 b7 b0 +0: 3 | 4 +1: 1 | 2 1 to 4: IP address octet
i_uTarget_ Port_No	Destination port number	Word [unsigned]/bit string [16-bit]	1 to 65534	Specify the UDP port number of an external device. For the port number to specify, check the manual for the external device.
i_uChannel	Own station channel	Word [unsigned]/bit string [16-bit]	1 to 17	Specify the channel to be used by the own station.*1

*1 To perform communications using the frame without the serial number on this FB, specify 1 to the own station channel. When a number of 2 to 9 is specified, this FB communicates using the frame with the serial number. When a number of 10 to 17 is specified, this FB communicates using the station number extension frame.

■Output arguments

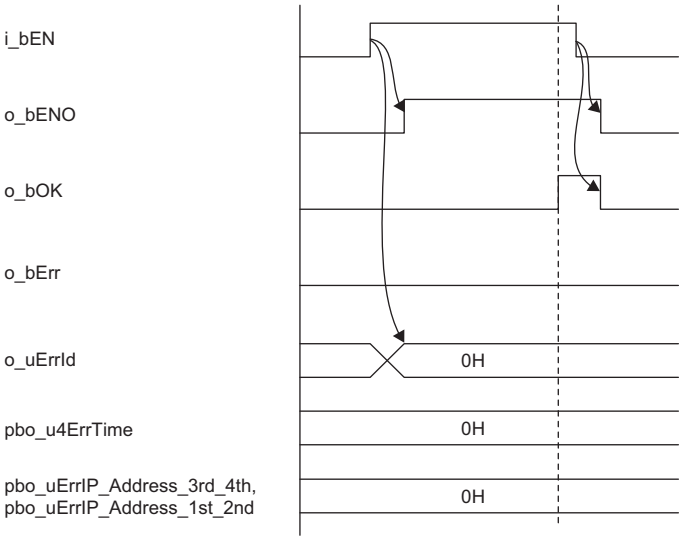
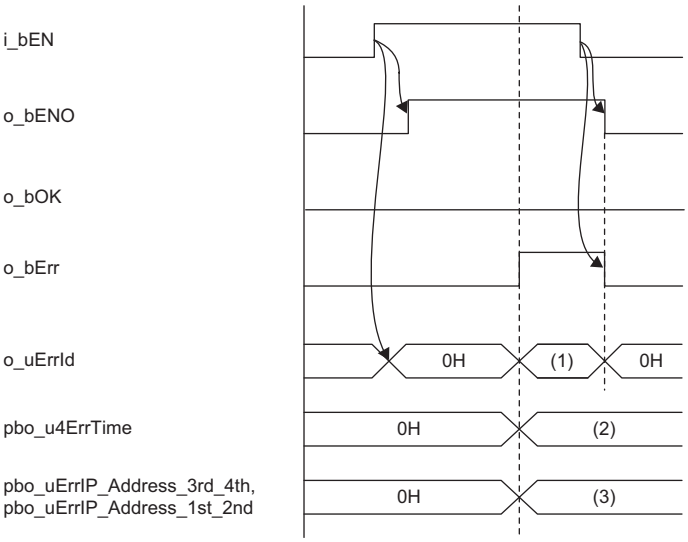
Variable name	Name	Data type	Description	Default value
o_bENO	Execution status	Bit	On: The execution command is turned on. Off: The execution command is turned off.	Off
o_bOK	Normal completion	Bit	The FB has been processed normally when this argument is on.	Off
o_bErr	Error completion	Bit	The FB has been processed abnormally when this argument is on.	Off
o_uErrId	Error code	Word [unsigned]/bit string [16-bit]	An error code is stored at error completion.	0

■Public variables

Variable name	Name	Data type	Description	Default value
pbo_u4ErrTime	Error occurrence time	Word [unsigned]/bit string [16-bit] (0..3)	Clock data at the time of error occurrence is stored.*1 1st word • Upper 8 bits: Month (01H to 12H) • Lower 8 bits: Lower 2 digits of year (00H to 99H) 2nd word • Upper 8 bits: Hour (00H to 23H) • Lower 8 bits: Day (01H to 31H) 3rd word • Upper 8 bits: Second (00H to 59H) • Lower 8 bits: Minute (00H to 59H) 4th word • Upper 8 bits: Upper 2 digits of year (00H to 99H) • Lower 8 bits: Day of week (00H (Sunday) to 06H (Saturday))	0
pbo_uErrIP_Address_3rd_4th	Error-detected device IP addresses (the third and fourth octets)	Word [unsigned]/bit string [16-bit]	The IP addresses (the third and fourth octets) of the station in which an error was detected are stored.*1 Example: When the IP address is 192.168.1.2 0102H	0
pbo_uErrIP_Address_1st_2nd	Error-detected device IP addresses (the first and second octets)	Word [unsigned]/bit string [16-bit]	The IP addresses (the first and second octets) of the station in which an error was detected are stored.*1 Example: When the IP address is 192.168.1.2 C0A8H	0

*1 The value is stored only when the dedicated instruction was completed with an error.
The value set to the target station address of the input argument is stored in the error-detected device IP addresses (the third and fourth octets) and the error-detected device IP addresses (the first and second octets).

Processing details

Item	Description	
Available device	Target module	MELSEC MX Controller (port for CC-Link IE TSN)
	CPU module	MELSEC MX Controller
	Engineering tool	GX Works3
Language	Ladder diagram	
Usage of labels	Label: 72 points Latched label: 0 points The usage of labels embedded in a program depends on the controller used, the devices specified for arguments, and the option settings of GX Works3. For the option settings of GX Works3, refer to the following. GX Works3 Operating Manual	
Processing	When i_bEN (execution command) is turned on, this function sends the remote RESET of an SLMP request to the target station.	
FB compilation method	Macro type	
FB operation	Pulse type (multiple-scan execution type)	
Input condition for FB_EN	None	
Timing chart of I/O signals	• For normal completion  • For error completion (When the dedicated instruction was completed with an error)  (1) Error code (2) Error occurrence time (3) Error-detected station address	

Item	Description
Precautions	• This FB does not include error recovery processing. Please create error recovery processing separately according to the system and required operations. • This FB uses the SLMPSND instruction. Even if the target device has sent an abnormal response, the SLMPSND instruction is completed successfully. • When the SLMPSND instruction is completed successfully, the values are not stored to the error occurrence time of the public variable, pbo_uErrIP_Address_3rd_4th (error-detected device IP addresses (the third and fourth octets)), and pbo_uErrIP_Address_1st_2nd (error-detected device IP addresses (the first and second octets)).*1 • Turn off i_bEN (execution command) after o_bOK (normal completion) or o_bErr (error completion) is turned on. By turning off i_bEN (execution command), o_bOK (normal completion) or o_bErr (error completion) is turned off and o_uErrId (error code) is cleared to 0. • When the remote RESET request is sent and completed successfully, o_bOK (normal completion) is turned on. Whether the target station is actually reset remotely or not depends on the target station status. • In this FB, stations in other network cannot be set as the target station. • The external device must support SLMP command (Remote Reset (1006H)). • This FB uses UDP communications. The external device must have the same settings.

*1 If 0 (initial value) is stored in the error occurrence time, the error-detected device IP addresses (the third and fourth octets), and the error-detected device IP addresses (the first and second octets), check and take actions using the manuals for the SLMP-compatible device used.

Operation error

The error code occurred by this FB is the same as the SLMPSND instruction.

☞ Page 1147 G(P).SLMPSND, J(P).SLMPSND, SP.SLMPSND

APPENDICES

Appendix 1 Instruction Processing Time

The following table lists the processing time of each instruction.
The processing time varies depending on the contents of the source and destination, and the processing optimized according to the program. Assume that the values in the table are reference values.

Instruction classification			Instruction name	Condition	Processing time [ns]	
Large classification	Middle classification	Small classification			MXR300-16, MXR300-32, MXR300-64	MXR500-128, MXR500-256
					Minimum	Minimum
Sequence instruction	Contact instruction	—	LD	Bit device	0.38	0.28
				Bit type label	0.40	0.29
	Output instruction	—	OUT	Bit device	15.29	11.12
				Bit type label	0.33	0.24
			OUT T	—	28.50	20.70
			OUT LT	—	98.90	73.80
Basic instruction	Data transfer instruction	—	MOV	—	0.68	0.5
			BMOV	n=1	106	76.6
				n=1024	262	191
			FMOV	n=1	64.2	46.7
				n=1024	110	79.7
	Arithmetic operation instruction	Increment	INC	—	5.05	3.61
		Four arithmetic operations instruction (integer)	+	—	0.64	0.46
			-	—	0.64	0.46
			*	—	0.63	0.46
			/	—	0.63	0.46

Instruction classification			Instruction name	Condition	Processing time [ns]	
Large classification	Middle classification	Small classification			MXR300-16, MXR300-32, MXR300-64	MXR500-128, MXR500-256
					Minimum	Minimum
Application instruction	Floating-point instruction	Four arithmetic operations instruction (single-precision real number)	E+	s1 = 2^{126}, s2 = 2^{127}	46.8	34
			E-	s1 = 2^{126}, s2 = 2^{127}	46.8	34
			E*	s1 = 2^{63}, s2 = 2^{64}	46.8	34
			E/	s1 = 2^{126}, s2 = 2^{127}	57.7	41.9
		Four arithmetic operations instruction (double-precision real number)	ED+	s1 = 2^{1022}, s2 = 2^{1023}	44.1	32.1
			ED-	s1 = 2^{1022}, s2 = 2^{1023}	44.1	32.1
			ED*	s1 = 2^{511}, s2 = 2^{512}	44.3	32.1
			ED/	s1 = 2^{1022}, s2 = 2^{1023}	57.7	41.9
		Trigonometric function (single-precision real number)	SIN	s = E0.7853982 (45 degrees)	83.3	60.5
		Trigonometric function (double-precision real number)	SIND	s = E0.7853982 (45 degrees)	62.6	45.5
		Real number data transfer (single-precision real number)	EMOV	s = Internal device, d = Internal device	0.69	0.5
				s = E1.23, d = Internal device	0.63	0.46
		Real number data transfer (double-precision real number)	EDMOV	s = Internal device, d = Internal device	0.68	0.5
				s = E1.23, d = Internal device	0.63	0.46
	PID operation instruction	—	PID	—	103	74.9
	PID control instruction	Inexact differential	S.PIDINIT	One loop	358	261
				32 loops	1250	905
			S.PIDCONT	One loop	115	83.6
				32 loops	2140	1360
	Converting multiple data for string concatenation	—	SPF	s1 = %.5d, s2 = 12345	638	467
				s1 = %.5s, s2 = "12345"	601	436
				s1 = %.5wf, s2 = 1.2345	1810	1330
				s1 = %.5d%.5s, s2+0 = 12345, s2+1 = "12345"	1230	883
	Structure creation instruction	Iteration instruction	FOR	—	0.13	0.09
			NEXT	—	0.04	0.03
		Subroutine call instruction	CALL	Local pointer	352	243
				Global pointer	395	278
			RET	Returning to the own program	235	163
				Returning to another program	264	186
	Module access instruction	—	TO	n=1	4110	3740
				n=1024	24500	24000
			FROM	n=1	5890	5470
				n=1024	43500	41400
Dedicated instruction	Built-in Ethernet instruction	Sending an SLMP message	GP.SLMPSND	Read (command: 0401H) (reading in units of words): Number of read points = One point	778	567
	Intelligent function module instruction	Reading data from the programmable controller on another station	GP.READ	—	66900	64900

Instruction classification			Instruction name	Condition	Processing time [ns]	
Large classification	Middle classification	Small classification			MXR300-16, MXR300-32, MXR300-64	MXR500-128, MXR500-256
					Minimum	Minimum
FB/FUN call	FB call	FB call	—	Local FB	286	208
				Global FB	279	205
	FUN call	FUN call	—	—	274	204
	Inline ST call	Inline ST call	—	—	2.72	1.98
ST control statement	Conditional branch	—	IF	—	0.65	0.47
	Iteration	—	CASE	—	2.52	1.84
			ST_FOR	—	7.45	5.44

Time added to instruction processing time

The following shows the processing time added to each instruction processing time according to the conditions.

Execution condition check time

For instructions whose execution condition is "rising edge" or "falling edge", the processing time shown in the following table is required for each instruction to check the execution conditions.
For example, the processing time for the MOVP instruction is "MOV instruction processing time + execution condition check addition time".

Execution condition	Processing time [ns]	
	MXR300-16, MXR300-32, MXR300-64	MXR500-128, MXR500-256
Rising edge/falling edge	28.62	20.80

Device access time

When the devices listed in the following table are specified as arguments, the processing time shown in the following table is added for each device.

Device name	Specified location	Processing time [μs]	
		MXR300-16, MXR300-32, MXR300-64	MXR500-128, MXR500-256
Module access device (U\G) and module label (ones with the label name including _D)	Source	6.37	5.45
	Destination	3.82	3.74
Link direct device (J\) and module label (link devices with the label name including _D)	Source	12.90	12.10
	Destination	10.30	9.25

Appendix 2 Module Dedicated Instruction Availability

This section describes whether module dedicated instructions can be used when a MELSEC iQ-R series module is used by the controller. (As of May 2025)

For details on the module dedicated instructions, refer to the following.

📖 MELSEC iQ-R Programming Manual (Module Dedicated Instructions)

Ethernet, CC-Link IE, MELSECNET/H common instruction

○: Applicable, ×: Not applicable

■Link dedicated instructions

Instruction symbol	Availability
JP.READ	○
GP.READ	○
JP.SREAD	○
GP.SREAD	○
JP.WRITE	○
GP.WRITE	○
JP.SWRITE	○
GP.SWRITE	○
JP.SEND	○
GP.SEND	○
JP.RECV	○
GP.RECV	○
G.RECVS	○
Z.RECVS	○
J.ZNRD	○
JP.ZNRD	○
J.ZNWR	○
JP.ZNWR	○
J.REQ	○
JP.REQ	○
G.REQ	○
GP.REQ	○
J.REQ	○
JP.REQ	○
G.REQ	○
GP.REQ	○

■CC-Link dedicated instructions

Instruction symbol	Availability
J.RIRD	○
JP.RIRD	○
G.RIRD	○
GP.RIRD	○
J.RIWT	○
JP.RIWT	○
G.RIWT	○
GP.RIWT	○

Ethernet instructions

○: Applicable, ×: Not applicable

■Open/close processing instructions

Instruction symbol	Availability
GP.CONOPEN	○
GP.CONCLOSE	○
GP.OPEN	○
ZP.OPEN	○
GP.CLOSE	○
ZP.CLOSE	○

■Socket communications instructions

Instruction symbol	Availability
GP.SOCRCV	○
G.SOCRCVS	○
GP.SOCSND	○

■Fixed buffer communications instructions

Instruction symbol	Availability
GP.BUFRCV	○
ZP.BUFRCV	○
G.BUFRCVS	○
Z.BUFRCVS	○
GP.BUFSND	○
ZP.BUFSND	○

■Reinitializing the module

Instruction symbol	Availability
G.UINI	○
GP.UINI	○
Z.UINI	○
ZP.UINI	○

■Executing the protocols registered for the predefined protocol support function

Instruction symbol	Availability
GP.ECPRTCL	○

■Clearing error information

Instruction symbol	Availability
GP.ERRCLEAR	○
ZP.ERRCLEAR	○

■Reading error information

Instruction symbol	Availability
GP.ERRRD	○
ZP.ERRRD	○

CC-Link IE TSN instructions

○: Applicable, ×: Not applicable

■Reading data from remote stations

Instruction symbol	Availability
JP.REMFR	○
ZP.REMFR	○
JP.REMFRD	○
GP.REMFRIP	○
GP.REMFRDIP	○

■Writing data to remote stations

Instruction symbol	Availability
JP.REMTO	○
ZP.REMTO	○
JP.REMTOD	○
GP.REMTOIP	○
GP.REMTODIP	○

■Parameter settings

Instruction symbol	Availability
G.CCPASETX	○
GP.CCPASETX	○
G.CCPASET	○
GP.CCPASET	○

■Own station number/IP address setting

Instruction symbol	Availability
G.UINI	○
GP.UINI	○
Z.UINI	○
ZP.UINI	○

■Sending an SLMP message

Instruction symbol	Availability
J.SLMPSND	○
JP.SLMPSND	○
G.SLMPSND	○
GP.SLMPSND	○

CC-Link IE Controller Network and MELSECNET/H common instructions

○: Applicable, ×: Not applicable

■Remote RUN

Instruction symbol	Availability
J.RRUN	○
JP.RRUN	○
G.RRUN	○
GP.RRUN	○
Z.RRUN	○
ZP.RRUN	○

■Remote STOP

Instruction symbol	Availability
J.RSTOP	○
JP.RSTOP	○
G.RSTOP	○
GP.RSTOP	○
Z.RSTOP	○
ZP.RSTOP	○

■Reading clock data from the programmable controller on another station

Instruction symbol	Availability
J.RTMRD	○
JP.RTMRD	○
G.RTMRD	○
GP.RTMRD	○
Z.RTMRD	○
ZP.RTMRD	○

■Writing clock data to the programmable controller on another station

Instruction symbol	Availability
J.RTMWR	○
JP.RTMWR	○
G.RTMWR	○
GP.RTMWR	○
Z.RTMWR	○
ZP.RTMWR	○

■Setting the station number to own station

Instruction symbol	Availability
G.UINI	○
GP.UINI	○
Z.UINI	○
ZP.UINI	○

CC-Link IE Field Network instructions

○: Applicable, ×: Not applicable

■Reading data from the intelligent device station/remote device station

Instruction symbol	Availability
JP.REMFR	○
ZP.REMFR	○
JP.REMFRD	○

■Writing data to the intelligent device station/remote device station

Instruction symbol	Availability
JP.REMTO	○
ZP.REMTO	○
JP.REMTOD	○

■Reading the module model information of an intelligent device station

Instruction symbol	Availability
J.SINFTYRD	○
JP.SINFTYRD	○
G.SINFTYRD	○
GP.SINFTYRD	○

■Reading the module status information of an intelligent device station

Instruction symbol	Availability
J.SINFSTRD	○
JP.SINFSTRD	○
G.SINFSTRD	○
GP.SINFSTRD	○

■Parameter settings

Instruction symbol	Availability
G.CCPASET	○
GP.CCPASET	○
G.CCPASETR	×
GP.CCPASETR	×

■Setting the station number to own station

Instruction symbol	Availability
G.UINI	○
GP.UINI	○
Z.UINI	○
ZP.UINI	○

■Sending an SLMP frame

Instruction symbol	Availability
J.SLMPREQ	○
JP.SLMPREQ	○
G.SLMPREQ	○
GP.SLMPREQ	○

CC-Link instructions

○: Applicable, ×: Not applicable

■Reading data from the target station

Instruction symbol	Availability
G.RIRD	○
GP.RIRD	○

■Writing data to the target station

Instruction symbol	Availability
G.RIWT	○
GP.RIWT	○

■Reading data from the buffer memory of the specified intelligent device station

Instruction symbol	Availability
G.RIRCV	○
GP.RIRCV	○

■Writing data to the buffer memory of the specified intelligent device station

Instruction symbol	Availability
G.RISEND	○
GP.RISEND	○

■Reading data from the automatic update buffer

Instruction symbol	Availability
G.RIFR	○
GP.RIFR	○

■Writing data to the automatic update buffer

Instruction symbol	Availability
G.RITO	○
GP.RITO	○

■Setting network parameters

Instruction symbol	Availability
G.RLPASET	○
GP.RLPASET	○

■Performing a message transmission to a remote device station

Instruction symbol	Availability
G.RDMSG	○
GP.RDMSG	○

Serial communication instructions

○: Applicable, ×: Not applicable

■Sending data using the on-demand function

Instruction symbol	Availability
G.ONDEMAND	○
GP.ONDEMAND	○

■Executing the protocols registered for the predefined protocol support function

Instruction symbol	Availability
G.CPRTCL	○
GP.CPRTCL	○

■Sending data using the nonprocedural protocol

Instruction symbol	Availability
G.OUTPUT	○
GP.OUTPUT	○

■Receiving data using the nonprocedural protocol

Instruction symbol	Availability
G.INPUT	○

■Sending data using the bidirectional protocol

Instruction symbol	Availability
G.BIDOUT	○
GP.BIDOUT	○

■Receiving data using the bidirectional protocol

Instruction symbol	Availability
G.BIDIN	○
GP.BIDIN	○

■Reading the data send/receive status

Instruction symbol	Availability
G.SPBUSY	○
GP.SPBUSY	○

■Receiving data using the interrupt program

Instruction symbol	Availability
Z.BUFRCVS	○

■Sending data by using user frames

Instruction symbol	Availability
G.PRR	○
GP.PRR	○

■Clearing receive data

Instruction symbol	Availability
ZP.CSET	○

■Registering/canceling the programmable controller CPU monitoring

Instruction symbol	Availability
ZP.CSET	○

■Initial setting

Instruction symbol	Availability
ZP.CSET	○

■Registering user frames

Instruction symbol	Availability
G.PUTE	○
GP.PUTE	○

■Reading user frames

Instruction symbol	Availability
G.GETE	○
GP.GETE	○

■Switching the mode

Instruction symbol	Availability
ZP.UINI	○

A/D conversion instructions

○: Applicable, ×: Not applicable

■Switching the mode

Instruction symbol	Availability
G.OFFGAN	○
GP.OFFGAN	○

■Reading the user range setting values

Instruction symbol	Availability
G.OGLOAD	○
GP.OGLOAD	○

■Restoring the user range setting values

Instruction symbol	Availability
G.OGSTOR	○
GP.OGSTOR	○

Positioning instructions

○: Applicable, ×: Not applicable

■Absolute position restoration

Instruction symbol	Availability
G.ABRST1	○
G.ABRST2	○
G.ABRST3	○
G.ABRST4	○
Z.ABRST1	○
Z.ABRST2	○
Z.ABRST3	○
Z.ABRST4	○

■Starting the positioning

Instruction symbol	Availability
GP.PSTRT1	○
GP.PSTRT2	○
GP.PSTRT3	○
GP.PSTRT4	○
ZP.PSTRT1	○
ZP.PSTRT2	○
ZP.PSTRT3	○
ZP.PSTRT4	○

■Teaching

Instruction symbol	Availability
GP.TEACH1	○
GP.TEACH2	○
GP.TEACH3	○
GP.TEACH4	○
ZP.TEACH1	○
ZP.TEACH2	○
ZP.TEACH3	○
ZP.TEACH4	○

■Backing up module data (writing data to the flash ROM)

Instruction symbol	Availability
GP.PFWRT	○
ZP.PFWRT	○

■Initializing the module

Instruction symbol	Availability
GP.PINIT	○
ZP.PINIT	○

High speed data logger module instructions

○: Applicable, ×: Not applicable

■File access instructions

Instruction symbol	Availability
Z.RCPWRITE	×
ZP.RCPWRITE	×
Z.RCPREAD	×
ZP.RCPREAD	×

C intelligent function module instructions

○: Applicable, ×: Not applicable

■User function execution instruction

Instruction symbol	Availability
G.CEXECUTE	×
GP.CEXECUTE	×

■Data analysis function execution instruction

Instruction symbol	Availability
G.DIGIFLTR	×
GP.DIGIFLTR	×
G.ENVCALC	×
GP.ENVCALC	×
G.FFTSPECT	×
GP.FFTSPECT	×

■Statistical analysis function execution instruction

Instruction symbol	Availability
G.LEASTSQ	×
GP.LEASTSQ	×
G.MTUNIT	×
GP.MTUNIT	×
G.MTMD	×
GP.MTMD	×
G.MULTIRG	×
GP.MULTIRG	×

DeviceNet instructions

○: Applicable, ×: Not applicable

■Reading master function receive data area

Instruction symbol	Availability
G.DNTMRD	○

■Writing master function send data area

Instruction symbol	Availability
G.DNTMWR	○

■Reading slave function receive data area

Instruction symbol	Availability
G.DNTSRD	○

■Writing slave function send data area

Instruction symbol	Availability
G.DNTSWR	○

Appendix 3 Determining Three PID Constants

The auto tuning function of PID operation instructions is performed in two methods: limit cycle method and step response method.

Overview of limit cycle method

This section describes the limit cycle method that is a method to determine the amplitude (a) and vibration period (τ, τ_{on}) of the input values, and calculate the proportional gain (K_P), integral time (T_I), and derivative time (T_D) according to the following expression of "Operation characteristics and three constants".

■Limit cycle method

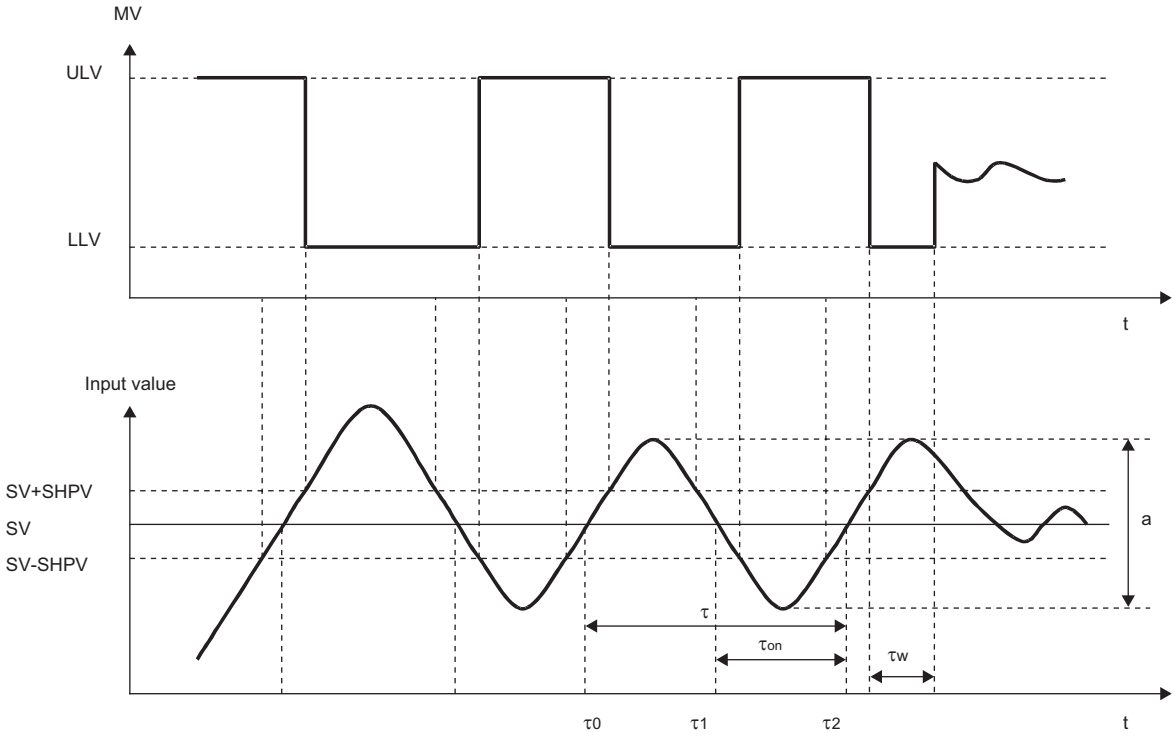
This method determines three PID constants by measuring the variations of input values while two-position control (output by switching between the output upper limit (ULV) and output lower limit (LLV) according to the deviation) is performed.

■Operation characteristics (reverse action example)

After the end of tuning cycle, the output lower limit (LLV) is retained for the manipulated value (MV) during τ_W, and a transition to the normal PID control occurs.

τ_W can be determined by $(50+K_W)/100 \times (\tau-\tau_{on})$, and the wait setting parameter (K_W) can be set in parameter (s3)+28.

(Setting range K_W = -50 to 32717 [%]; If an abnormal range is specified, operation is performed assuming τ_W = 0.)



ULV: Output upper limit value
LLV: Output lower limit value
SV: Set value
t: Time
SHPV: PV value threshold (hysteresis) width

■Operation characteristics and three constants

Control method	Proportional gain (K_P) [%]	Integral time (T_I) [×100ms]	Derivative time (T_D) [×10ms]
Proportional control (P action) only	$\frac{1}{a}$ (ULV-LLV) × 100	—	—
PI control (PI action)	$\frac{0.9}{a}$ (ULV-LLV) × 100	$33 \times \tau_{on}\left(1 - \frac{\tau_{on}}{\tau}\right)$	—
PID control (PID action)	$\frac{1.2}{a}$ (ULV-LLV) × 100	$20 \times \tau_{on}\left(1 - \frac{\tau_{on}}{\tau}\right)$	$50 \times \tau_{on}\left(1 - \frac{\tau_{on}}{\tau}\right)$

Overview of step response method

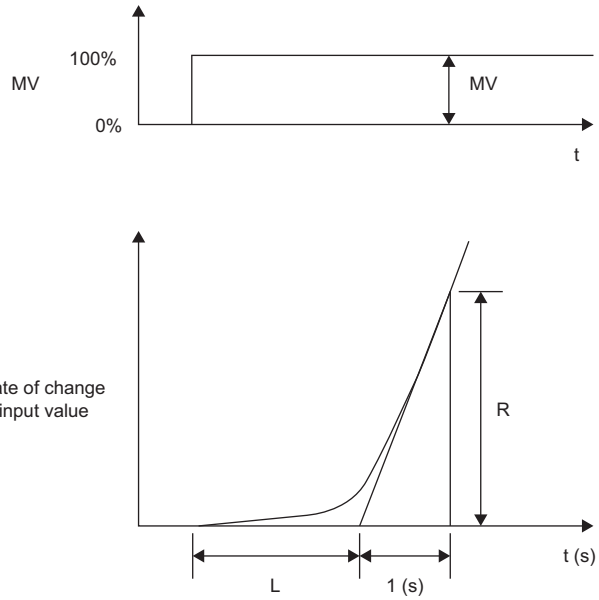
This section describes the step response method that is used to determine the optimum value of three PID constants (proportional gain (K_P), integral time (T_I), and derivative time (T_D)).

■Step response method

This method determines the three PID constants according to the operation characteristics (maximum ramp (R), dead time (L)) that can be determined from input changes by providing the control system with stepwise output of 0→100%*1.

*1 Stepwise output can also be determined by 0→75% or 0→50%.

■Operation characteristics



MV: Manipulated value
L: Dead time
R: Maximum ramp
t: Time

■Operation characteristics and three constants

Control method	Proportional gain (K_P) [%]	Integral time (T_I) [×100ms]	Derivative time (T_D) [×10ms]
Proportional control (P action) only	$\frac{1}{RL}$ × MV × 100	—	—
PI control (PI action)	$\frac{0.9}{RL}$ × MV × 100	33L	—
PID control (PID action)	$\frac{1.2}{RL}$ × MV × 100	20L	50L

INDEX

Symbols

- 93
* 93
** 93
/ 93
& 93
+ 93
< 93
<= 93
<> 93
= 93
> 93
>= 93

0 to 9

16-bit data (word data) 32
32-bit data (double word data) 35

A

AND 93
APPLICATION INSTRUCTIONS 531
ASCII 104,116
Assignment statement 95

B

Basic instruction 254
Bit data 31
Buffer memory 21

C

CASE 98
CC-Link IE Controller Network-equipped module 24
CC-Link IE TSN master/local module 24
Conversion of data type 96,109

D

Data link 24
Destination (d) 26
Device station 21
Device supporting iQSS 24
Double-precision real number data 39

E

EN 62,74
ENO 62,74
Execution condition 43
EXIT 99
External variable 67

F

FB file 66,75
FOR...DO 98,102
FUN file 61,63
Function (FUN) 59,61
Function block (FB) 59,65
Function block call statement 97
Function call statement 97

I

IF THEN 98
IF...ELSE 98
IF...ELSIF 98
Input variable 62,66
Input/output variable 66
Instance 68
Instruction configuration 26
Internal variable 67
Interrupt program 60

L

Ladder diagram 85
Link device 22
Local station 22

M

Main routine program 60
Master station 22
MOD 93

N

NOT 93
Note 90
Numerical value (n) 27

O

OR 93
Output variable 62,66

P

PID control instruction 1073
PROFILE_POSITIONING_DATA_EXTENDED 1458
Program 63,75
Program Block 60

R

Real number data (floating-point data) 38
REPEAT...UNTIL 98
Reserved word 92
RETURN 97

S

Sequence instructions 202
Shift JIS 104,116
Single-precision real number data 38
Source (s) 26

Statement 90
Station sub ID number 24
STRING 104,116
String 104,116
String [Unicode] 104,116
String data 41
Subroutine program 60
Subroutine type function block 66

T

Type specifier 105,116
Types of PID instructions 1057

U

Unicode 104

W

WHILE...DO 98
WSTRING 104,116

X

XOR 93

INSTRUCTION INDEX

Symbols

-(P)(_U) 268,270
*(P)(_U) 280
/(P)(_U) 282
+(P)(_U) 264,266
$+(P) 740,742
$MOV(P) 744
$MOV(P)_WS 746

A

ABS(_E) 1274
ACOS(_E) 1285
ACOS(P) 890
ACOSD(P) 902
ADD(_E) 1287
ADDONEXE(P) 1105
ADDON_MONI 1448
ADDON_PARAM 1448
ADD_TIME(_E) 1345
ADRSET(P) 977
ADV_CAM_MONI 1470
ADV_CAM_PARAM 1469
ADV_CLUTCH_CMD 1466
ADV_CLUTCH_MONI 1466
ADV_CLUTCH_PARAM 1463
ADV_GEAR_PARAM 1462
ADV_INTPUT_MONI 1435
ADV_INTPUT_PRM 1435
ADV_INTPUT_PRM_CONST 1434
ADV_OUTPUT_CMD 1440
ADV_OUTPUT_MONI 1440
ADV_OUTPUT_PRM 1438
ADV_OUTPUT_PRM_CONST 1437
ADV_RESTORE_PARAM 1472
ADV_SPEEDCHANGEGEAR_PARAM 1468
ANB 210
AND 202
AND(_E) 1310
AND<(_U) 254
AND<=(_U) 254
AND<>(_U) 254
AND=(_U) 254
AND>(_U) 254
AND>=(_U) 254
AND$<. 737
AND$<=. 737
AND$<>. 737
AND$=. 737
AND$>. 737
AND$>=. 737
ANDD<(_U) 256
ANDD<=(_U) 256
ANDD<>(_U) 256
ANDD=(_U) 256
ANDD>(_U) 256
ANDD>=(_U) 256
ANDD_EQ(_U) 256
ANDD_GE(_U) 256
ANDD_GT(_U) 256
ANDD_LE(_U) 256
ANDD_LT(_U) 256
ANDD_NE(_U) 256
ANDDT< 1004
ANDDT<= 1004
ANDDT<> 1004
ANDDT= 1004
ANDDT> 1004
ANDDT>= 1004
ANDDT_EQ 1004
ANDDT_GE 1004
ANDDT_GT 1004
ANDDT_LE 1004
ANDDT_LT 1004
ANDDT_NE 1004
ANDE< 829
ANDE<= 829
ANDE<> 829
ANDE= 829
ANDE> 829
ANDE>= 829
ANDED< 831
ANDED<= 831
ANDED<> 831
ANDED= 831
ANDED> 831
ANDED>= 831
ANDED_EQ 831
ANDED_GE 831
ANDED_GT 831
ANDED_LE 831
ANDED_LT 831
ANDED_NE 831
ANDE_EQ 829
ANDE_GE 829
ANDE_GT 829
ANDE_LE 829
ANDE_LT 829
ANDE_NE 829
AND_EQ(_U) 254
ANDF 204
ANDFI 207
AND_GE(_U) 254
AND_GT(_U) 254
AND_LE(_U) 254
AND_LT(_U) 254
AND_NE(_U) 254
ANDP 204
ANDPI 207
ANDSTRING_EQ 737
ANDSTRING_GE 737
ANDSTRING_GT 737
ANDSTRING_LE 737
ANDSTRING_LT 737
ANDSTRING_NE 737
ANDTM< 1008
ANDTM<= 1008
ANDTM<> 1008
ANDTM= 1008
ANDTM> 1008
ANDTM>= 1008
ANDTM_EQ 1008
ANDTM_GE 1008

ANDTM_GT 1008
ANDTM_LE 1008
ANDTM_LT 1008
ANDTM_NE 1008
ANI 202
ASC2INT(P) 474
ASIN(_E) 1284
ASIN(P) 888
ASIND(P) 900
ATAN(_E) 1286
ATAN(P) 892
ATAND(P) 904
AXES_GROUP_CMD 1429
AXES_GROUP_MONI 1426
AXES_GROUP_PRM 1424
AXES_GROUP_PRM_CONST 1423
AXES_GROUP_REF 1423
AXIS_CMD 1417
AXIS_MONI 1408
AXIS_PRM 1403
AXIS_PRM_CONST 1399
AXIS_REF 1398,1434,1436

B

B-(P) 291,292
B*(P) 300
B/(P) 302
B+(P) 288,289
BACOS(P) 914
BAND(P)(_U) 665
BASIN(P) 912
BATAN(P) 916
BCD(P) 390
BCDDA(P) 772
BCD_TO_DINT(_E) 1226
BCD_TO_INT(_E) 1224
BCD_TO_STRING(_E) 1229
BCOS(P) 908
BDIVISION(P) 302
BDSQRT(P) 940
BIN(P) 394
BINDA(P)(_U) 748
BINHA(P) 758
BITARR_TO_DINT(_E) 1263
BITARR_TO_INT(_E) 1262
BK-(P)(_U) 310
BK+(P)(_U) 308
BKAND(P) 331
BKBCD(P) 446
BKBIN(P) 448
BKCMP<(P)(_U) 258
BKCMP<=(P)(_U) 258
BKCMP<>(P)(_U) 258
BKCMP=(P)(_U) 258
BKCMP>(P)(_U) 258
BKCMP>=(P)(_U) 258
BKCMP_EQ(P)(_U) 258
BKCMP_GE(P)(_U) 258
BKCMP_GT(P)(_U) 258
BKCMP_LE(P)(_U) 258
BKCMP_LT(P)(_U) 258
BKCMP_NE(P)(_U) 258
BKMINUS(P)(_U) 310
BKOR(P) 340
BKPLUS(P)(_U) 308
BKRST(P) 368
BKXNR(P) 358
BKXOR(P) 349
BLKMOVB(P) 529
BMINUS(P) 292
BMOV(P) 510
BMOVL(P) 512
BMULTI(P) 300
BOOL_TO_DINT(_E) 1182
BOOL_TO_DWORD(_E) 1180
BOOL_TO_INT(_E) 1181
BOOL_TO_STRING(_E) 1184
BOOL_TO_TIME(_E) 1183
BOOL_TO_WORD(_E) 1178
BPLUS(P) 289
BREAK(P) 553
BRST(P) 362
BSET(P) 360
BSFL(P) 376
BSFR(P) 374
BSIN(P) 906
BSQRT(P) 938
BTAN(P) 910
BTOW(P) 498
BXCH(P) 524

C

CALL(P) 555
CJ 532
CML(P) 504
CMLB(P) 508
COM(P) 1025
CONCAT(_E) 1334
COS(_E) 1282
COS(P) 884
COSD(P) 896
COUNTER_FB_M 1369
CPY_BITARR(_E) 1266
CPY_BIT_OF_INT(_E) 1271
CRC(P) 733
CTD(_E) 1364
CTU(_E) 1362
CTUD(_E) 1366
CYCLE_MONI 1448
CYCLE_PARAM 1448

D

D-(P)(_U) 276,278
D*(P)(_U) 284
D/(P)(_U) 286
D+(P)(_U) 272,274
DABCD(P) 462
DABIN(P)(_U) 450
DAND(P) 327,329
DATE-(P) 994
DATE+(P) 992
DATE2SEC(P)(_U) 1000
DATEMINUS(P) 994
DATEPLUS(P) 992
DATERD(P) 988
DATEWR(P) 990
DB-(P) 297,298
DB*(P) 304
DB/(P) 306

DB+(P) 294,295
DBAND(P)(_U) 667
DBCD(P) 392
DBCDDA(P) 776
DBDIVISION(P) 306
DBIN(P) 396
DBINDA(P)(_U) 753
DBINHA(P) 762
DBK-(P)(_U) 315
DBK+(P)(_U) 312
DBKCMP<(P)(_U) 261
DBKCMP<=(P)(_U) 261
DBKCMP<>(P)(_U) 261
DBKCMP=(P)(_U) 261
DBKCMP>(P)(_U) 261
DBKCMP>=(P)(_U) 261
DBKCMP_EQ(P)(_U) 261
DBKCMP_GE(P)(_U) 261
DBKCMP_GT(P)(_U) 261
DBKCMP_LE(P)(_U) 261
DBKCMP_LT(P)(_U) 261
DBKCMP_NE(P)(_U) 261
DBKMINUS(P)(_U) 315
DBKPLUS(P)(_U) 312
DBL2DINT(P) 410
DBL2FLT(P) 864
DBL2INT(P) 406
DBL2UDINT(P) 412
DBL2UINT(P) 408
DBMINUS(P) 298
DBMULTI(P) 304
DBPLUS(P) 295
DCML(P) 506
DDABCD(P) 465
DDABIN(P)(_U) 453
DDEC(P)(_U) 321
DDIVISION(P)(_U) 286
DEC(P)(_U) 319
DECO(P) 481
DEG(P) 920
DEGD(P) 924
DELETE(_E) 1338
DELTA(P) 241
DFMOV(P) 518
DFMOVL(P) 520
DFROM(P) 1029
DFROMD(P) 1037
DGBIN(P)(_U) 444
DGRY(P)(_U) 440
DHABIN(P) 459
DI 536,539
DI_1 539
DINC(P)(_U) 320
DINT2DBL(P) 868
DINT2FLT(P) 860
DINT2INT(P) 426
DINT2UDINT(P) 430
DINT2UINT(P) 428
DINT_TO_BCD(_E) 1217
DINT_TO_BITARR(_E) 1265
DINT_TO_BOOL(_E) 1212
DINT_TO_DWORD(_E) 1215
DINT_TO_INT(_E) 1216
DINT_TO_LREAL(_E) 1220
DINT_TO_REAL(_E) 1219
DINT_TO_STRING(_E) 1222
DINT_TO_TIME(_E) 1221
DINT_TO_WORD(_E) 1213
DIS(P) 488
DIV(_E) 1295
DIVISION(P)(_U) 282
DIV_TIME(_E) 1351
DLIMIT(P)(_U) 663
DMAX(P)(_U) 713
DMEAN(P)(_U) 729
DMIN(P)(_U) 717
DMINUS(P)(_U) 278
DMOV(P) 502
DMULTI(P)(_U) 284
DNEG(P) 480
DOR(P) 336,338
DPLUS(P)(_U) 274
DRCL(P) 593
DRCR(P) 591
DROL(P) 593
DROR(P) 591
DSCL(P)(_U) 684
DSCL2(P)(_U) 699
DSCL2N(P)(_U) 696
DSCLN(P)(_U) 680
DSERDATA(P) 703
DSERMM(P) 707
DSFL(P) 380
DSFR(P) 378
DSORTD(_U) 721
DSQRT(P) 732
DSTR(P)(_U) 769
DSUM(P) 710
DSWAP(P) 527
DTEST(P) 366
DTO(P) 1032
DTOD(P) 1041
DUTY 1019
DVAL(P)(_U) 471
DWORD_TO_BOOL(_E) 1192
DWORD_TO_DINT(_E) 1197
DWORD_TO_INT(_E) 1195
DWORD_TO_STRING(_E) 1199
DWORD_TO_TIME(_E) 1198
DWORD_TO_WORD(_E) 1193
DWSUM(P)(_U) 725
DXCH(P) 523
DXNR(P) 354,356
DXOR(P) 345,347
DZONE(P)(_U) 671

E

E-(P) 838,840
E*(P) 850
E/(P) 852
E+(P) 834,836
ECALL(P) 568
ED-(P) 846,848
ED*(P) 854
ED/(P) 856
ED+(P) 842,844
EDDIVISION(P) 856
EDIVISION(P) 852
EDMAX(P) 952
EDMIN(P) 956
EDMINUS(P) 848

EDMOV(P) 881
EDMULTI(P) 854
EDNEG(P) 879
EDPLUS(P) 844
EDSCLN(P) 687
EDSQRT(P) 928
EFCALL(P) 575
EGF 213
EGP 213
EI 536
EMAX(P) 950
EMIN(P) 954
EMINUS(P) 840
EMOD(P) 476
EMOV(P) 880
EMULTI(P) 850
ENCO(P) 483
END 250
ENEG(P) 878
EPLUS(P) 836
EQ(_E) 1323
EREXP(P) 876
ESQRT(P) 926
ESTR(P) 781
EVAL(P) 872
EXP(_E) 1280
EXP(P) 930
EXPD(P) 932
EXPT(_E) 1299

F

FCALL(P) 562
FDEL(P) 603
FEND 249
FF 240
FIFR(P) 595
FIFW(P) 599
FILE_LOCATION 1447
FIND(_E) 1343
FINS(P) 601
FLT2DBL(P) 870
FLT2DINT(P) 402
FLT2INT(P) 398
FLT2UDINT(P) 404
FLT2UINT(P) 400
FMOV(P) 514
FMOVL(P) 516
FOR 550
FPOP(P) 597
FROM(P) 1029
FROMD(P) 1037
F_TRIG(_E) 1360

G

G.SOCRCVS 1121
G(P).SLMPSND 1147
G(P).SLMPSNDC 1156
G(P).SOCRDATA 1137
G(P)_SLMPSND 1147
G(P)_SLMPSNDC 1156
G(P)_SOCRDATA 1137
GBIN(P)(_U) 442
GE(_E) 1323
GET_BIT_OF_INT(_E) 1267
GET_BOOL_ADDR 1273
GET_INT_ADDR 1273
GET_WORD_ADDR 1273
GOEND 535
GP.ECPRTCL 1139
GP.FTPCGET 1171
GP.FTPCPUT 1165
GP.SOCCINF 1128
GP.SOCCLOSE 1115
GP.SOCCSET 1131
GP.SOCOPEN 1110
GP.SOCRCV 1117
GP.SOCRMODE 1133
GP.SOCSND 1124
GP_ECPRTCL 1139
GP_FTPCGET 1171
GP_FTPCPUT 1165
GP_SOCCINF 1128
GP_SOCCLOSE 1115
GP_SOCCSET 1131
GP_SOCOPEN 1110
GP_SOCRCV 1117
GP_SOCRMODE 1133
GP_SOCSND 1124
GRY(P)(_U) 438
G_SOCRCVS 1121
GT(_E) 1323

H

HABIN(P) 457

I

IMASK 544
INC(P)(_U) 318
INSERT(_E) 1336
INSTANCE_ID 1448
INSTR(P) 823
INT2ASC(P) 787
INT2DBL(P) 866
INT2DINT(P) 416
INT2FLT(P) 858
INT2UDINT(P) 418
INT2UINT(P) 414
INT_TO_BCD(_E) 1205
INT_TO_BITARR(_E) 1264
INT_TO_BOOL(_E) 1200
INT_TO_DINT(_E) 1204
INT_TO_DWORD(_E) 1202
INT_TO_LREAL(_E) 1208
INT_TO_REAL(_E) 1207
INT_TO_STRING(_E) 1210
INT_TO_TIME(_E) 1209
INT_TO_WORD(_E) 1201
INV 211
IRET 548

J

J(P).SLMPSND 1147
J(P)_SLMPSND 1147
JMP 532

L

LD 202
LD<(_U) 254
LD<=(_U) 254
LD<>(_U) 254
LD=(_U) 254
LD>(_U) 254
LD>=(_U) 254
LD$< 737
LD$<= 737
LD$<> 737
LD$= 737
LD$> 737
LD$>= 737
LDD<(_U) 256
LDD<=(_U) 256
LDD<>(_U) 256
LDD=(_U) 256
LDD>(_U) 256
LDD>=(_U) 256
LDD_EQ(_U) 256
LDD_GE(_U) 256
LDD_GT(_U) 256
LDD_LE(_U) 256
LDD_LT(_U) 256
LDD_NE(_U) 256
LDDT< 1004
LDDT<= 1004
LDDT<> 1004
LDDT= 1004
LDDT> 1004
LDDT>= 1004
LDDT_EQ 1004
LDDT_GE 1004
LDDT_GT 1004
LDDT_LE 1004
LDDT_LT 1004
LDDT_NE 1004
LDE< 829
LDE<= 829
LDE<> 829
LDE= 829
LDE> 829
LDE>= 829
LDED< 831
LDED<= 831
LDED<> 831
LDED= 831
LDED> 831
LDED>= 831
LDED_EQ 831
LDED_GE 831
LDED_GT 831
LDED_LE 831
LDED_LT 831
LDED_NE 831
LDE_EQ 829
LDE_GE 829
LDE_GT 829
LDE_LE 829
LDE_LT 829
LDE_NE 829
LD_EQ(_U) 254
LDF 204
LDFI 207
LD_GE(_U) 254
LD_GT(_U) 254
LDI 202
LD_LE(_U) 254
LD_LT(_U) 254
LD_NE(_U) 254
LDP 204
LDPI 207
LDSTRING_EQ 737
LDSTRING_GE 737
LDSTRING_GT 737
LDSTRING_LE 737
LDSTRING_LT 737
LDSTRING_NE 737
LDTM< 1008
LDTM<= 1008
LDTM<> 1008
LDTM= 1008
LDTM> 1008
LDTM>= 1008
LDTM_EQ 1008
LDTM_GE 1008
LDTM_GT 1008
LDTM_LE 1008
LDTM_LT 1008
LDTM_NE 1008
LE(_E) 1323
LEDR 735
LEFT(_E) 1329
LEFT(P) 816
LEN(_E) 1327
LEN(P) 812
LIMIT(_E) 1318
LIMIT(P)(_U) 661
LN(_E) 1277
LOG(_E) 1278
LOG(P) 934
LOG10(P) 946
LOG10D(P) 948
LOGD(P) 936
LREAL_TO_DINT(_E) 1240
LREAL_TO_INT(_E) 1239
LREAL_TO_REAL(_E) 1241
LT(_E) 1323

M

M+MX-SQ_ConnectionClose 1950
M+MX-SQ_ConnectionOpen 1946
M+MX-SQ_Recv_Socket 1952
M+MX-SQ_Send_Socket 1955
M+MX-SQ_SLMP_DeviceRead_IP 1958
M+MX-SQ_SLMP_DeviceWrite_IP 1965
M+MX-TSN_RemoteReset2_IP 1986
M+MX-TSN_SLMP_DeviceRead_IP 1972
M+MX-TSN_SLMP_DeviceWrite_IP 1979
MAX(_E) 1316
MAX(P)(_U) 711
MC 245
MC_AbortTrigger 1550
MC_ABS_SYSTEM 1477
MC_ACC_ZERO_MODE 1478
MC_AXES_GROUP_STATUS 1475
MC_AXIS_STATUS 1475
MC_AXIS_TYPE 1477
MC_BUFFER_MODE 1474
MC_CAM_CURVE_TYPE 1476

MC_CAM_CYCLE_METHOD. 1481
MC_CAM_ID 1442
MC_CamIn. 1754
MC_CAM_REF. 1442
MC_CAM_REFERENCE_METHOD 1481
MC_CAM_RESTORE_METHOD 1481
MC_CAMSWITCH_DATA_REF 1443
MC_CAMSWITCH_REF 1443
MC_CAMSWITCH_SOURCE_REF 1443
MC_CamTableSelect 1555
MC_CIRC_MODE. 1475
MC_CIRC_PATHCHOICE 1475
MC_CLUTCH_METHOD 1480
MC_CLUTCH_REFERENCE 1480
MC_CLUTCH_SMOOTHING_METHOD 1480
MC_CombineAxes 1797
MC_COMBINE_MODE 1474
MC_CONTROL_METHOD. 1479
MC_DECEL_STOP_MODE 1477
MC_DigitalCamSwitch. 1590
MC_DIRECTION 1474
MC_DRIVE_MODE. 1476
MC_DRIVE_STATE 1477
MC_ENCODER_AXIS_TYPE. 1477
MC_ERROR_CLASS 1478
MC_EXECUTION_MODE 1474
MC_EXECUTION_STATE. 1478
MC_GearIn 1787
MC_GEAR_RESTORE_METHOD 1481
MC_GroupDisable 1486
MC_GroupEnable 1482
MC_GroupReset. 1530
MC_GroupSetOverride 1573
MC_GroupStop. 1616
MC_Home 1601
MC_INPUT_DIRECTION. 1480
MC_INPUT_REF 1442
MC_INTERPOLATE_SPEED_MODE 1476
MC_LENGTH_PER_CYCLE_CHANGE 1480
MC_MCODE_OUTPUT_OVERRIDE 1479
MC_MoveAbsolute 1622
MC_MoveRelative. 1635
MC_MoveVelocity. 1653
MC_OPERATION_PATTERN 1479
MC_OUTPUT_REF. 1442
MC_POSITIONING_DATA_ID 1442
MC_POSITIONING_DATA_REF 1442
MC_POS_RESTORATION_STATUS 1478
MC_POS_SOURCE 1478
MC_Power. 1490
MCR 245
MC_ReadParameter 1514
MC_ReadStatus 1598
MC_RECORD_MODE. 1476
MC_Reset 1527
MC_SetOverride. 1510
MC_SetPosition 1494
MC_SIGNAL_LOGIC. 1478
MC_SOURCE 1474
MC_START_MODE 1475
MC_STEP_MODE 1479
MC_Stop 1611
MC_STOP_CURVE_MODE. 1477
MC_SYNC_CHANGE_COMMAND. 1480
MC_TorqueControl 1661
MC_TouchProbe. 1534
MC_TRACK_REF. 1443
MC_TRIGGER_REF. 1442
MCv_AdvancedSync. 1862
MCv_AdvCamSetPositionCalc. 1587
MCv_AdvPositionPerCycleCalc 1584
MCv_AllPower 1570
MCv_BacklashCompensationFilter. 1808
MCv_ChangeCycle. 1562
MCv_CyclicPosition 1891
MCv_CyclicTorque 1903
MCv_CyclicVelocity 1899
MCv_DirectionFilter 1831
MC_VELOCITY_LIMIT_MODE 1476
MCv_GroupMoveWait. 1855
MCv_Jog,MCv 1644
MCv_MotionErrorReset. 1577
MCv_MoveCircularInterpolateAbsolute 1719
MCv_MoveCircularInterpolateRelative 1736
MCv_MoveLinearInterpolateAbsolute 1690
MCv_MoveLinearInterpolateRelative 1705
MCv_MovePositioningData 1868
MCv_MoveWait 1848
MCv_PositionControl 1878
MCv_PressureControl. 1672
MCv_ReadProfileData. 1909
MCv_SetTorqueLimit 1502
MCv_SmoothingFilter 1821
MCv_SpeedControl 1682
MCv_SpeedLimitFilter. 1840
MCv_SyncOperationCycles 1581
MCv_WriteProfileData. 1922
MC_WriteParameter 1521
MEAN(P)(_U). 727
MEF 212
MEP 212
MID(_E). 1332
MIDR(P) 818
MIDW(P) 820
MIN(_E). 1316
MIN(P)(_U) 715
MINUS(P)(_U) 270
MOD(_E) 1297
MOV(P). 500
MOVB(P). 528
MOVE(_E). 1300
MUL(_E) 1290
MULTI(P)(_U) 280
MUL_TIME(_E) 1349
MUX(_E) 1321

N

NDIS(P). 492
NE(_E) 1325
NEG(P) 478
NEXT 550
NOPLF 253
NOT(_E) 1313
NUNI(P) 494

O

OR 202
OR(_E) 1310
OR<(_U) 254
OR<=(_U) 254

OR<>(_U) 254
OR=(_U) 254
OR>(_U) 254
OR>=(_U) 254
OR$<. 737
OR$<=. 737
OR$<>. 737
OR$=. 737
OR$>. 737
OR$>=. 737
ORB 210
ORD<(_U) 256
ORD<=(_U) 256
ORD<>(_U) 256
ORD=(_U) 256
ORD>(_U) 256
ORD>=(_U) 256
ORD_EQ(_U) 256
ORD_GE(_U) 256
ORD_GT(_U) 256
ORD_LE(_U) 256
ORD_LT(_U) 256
ORD_NE(_U) 256
ORDT< 1004
ORDT<= 1004
ORDT<> 1004
ORDT= 1004
ORDT> 1004
ORDT>= 1004
ORDT_EQ 1004
ORDT_GE 1004
ORDT_GT 1004
ORDT_LE 1004
ORDT_LT 1004
ORDT_NE 1004
ORE< 829
ORE<= 829
ORE<> 829
ORE= 829
ORE> 829
ORE>= 829
ORED< 831
ORED<= 831
ORED<> 831
ORED= 831
ORED> 831
ORED>= 831
ORED_EQ 831
ORED_GE 831
ORED_GT 831
ORED_LE 831
ORED_LT 831
ORED_NE 831
ORE_EQ 829
ORE_GE 829
ORE_GT 829
ORE_LE 829
ORE_LT 829
ORE_NE 829
OR_EQ(_U) 254
ORF 204
ORFI 207
OR_GE(_U) 254
OR_GT(_U) 254
ORI 202
OR_LE(_U) 254
OR_LT(_U) 254
OR_NE(_U) 254
ORP 204
ORPI. 207
ORSTRING_EQ 737
ORSTRING_GE 737
ORSTRING_GT 737
ORSTRING_LE 737
ORSTRING_LT 737
ORSTRING_NE 737
ORTM< 1008
ORTM<= 1008
ORTM<> 1008
ORTM= 1008
ORTM> 1008
ORTM>= 1008
ORTM_EQ. 1008
ORTM_GE. 1008
ORTM_GT. 1008
ORTM_LE 1008
ORTM_LT 1008
ORTM_NE. 1008
OUT 215
OUT C. 223
OUT F. 227
OUT LC. 225
OUT LST 220
OUT LT 220
OUT ST. 217
OUT T. 217
OUT_C 223
OUTH 217
OUTH ST. 217
OUTH T. 217
OUT_T 217

P

PID 1069
PIDCONT(P) 1098
PIDINIT(P). 1096
PIDPRMW(P). 1103
PIDRUN(P) 1102
PIDSTOP(P) 1101
PLF. 238
PLS. 236
PLUS(P)(_U) 266
POW(P). 942
POWD(P) 944
PROFILE_CAM_DATA 1449
PROFILE_CAM_ELEMENT. 1450
PROFILE_CAMSWITCH_DATA. 1453
PROFILE_CAMSWITCH_ELEMENT 1453
PROFILE_DATA 1447
PROFILE_ID 1447
PROFILE_POSITIONING_DATA 1454
PROFILE_POSITIONING_DATA_ELEMENT . .1454
PROFILE_ROTARY_CUTTER. 1451
PROFILE_SIGNAL_SELECT_ELEMENT 1459

R

RAD(P) 918
RADD(P) 922
RCL(P) 588
RCR(P) 585

REAL_TO_DINT(_E) 1233
REAL_TO_INT(_E) 1231
REAL_TO_LREAL(_E) 1235
REAL_TO_STRING(_E) 1236
REPLACE(_E) 1340
RET 561
RFS(P) 1023
RIGHT(_E) 1329
RIGHT(P) 814
RND(P) 958
ROL(_E) 1306
ROL(P) 588
ROR(_E) 1308
ROR(P) 585
RS(_E) 1356
RSET(P) 970
RST 230
RST F 234
R_TRIG(_E) 1358

S

S.SOCRCVS 1121
S(P).DATE- 1017
S(P).DATE+ 1015
S(P).DATERD 1013
S(P).DEVLD 606
S(P).PIDCONT 1087
S(P).PIDINIT 1084
S(P).PIDPRMW 1092
S(P).PIDRUN 1091
S(P).PIDSTOP 1090
S(P).SOCRDATA 1137
S(P).ZCOM 1027
S(P)_DATEMINUS 1017
S(P)_DATEPLUS 1015
S(P)_DATERD 1013
S(P)_DEVLD 606
S(P)_PIDCONT 1087
S(P)_PIDINIT 1084
S(P)_PIDPRMW 1092
S(P)_PIDRUN 1091
S(P)_PIDSTOP 1090
S(P)_SOCRDATA 1137
S(P)_ZCOM 1027
SCJ 532
SCL(P)(_U) 677
SCL2(P)(_U) 694
SCL2N(P)(_U) 691
SCLN(P)(_U) 673
SEC2DATE(P)(_U) 1002
SEC2TIME(P) 998
SEG(P) 485
SEL(_E) 1314
SERDATA(P) 701
SERMM(P) 705
SET 228
SET F 232
SET_BIT_OF_INT(_E) 1269
SFL(P) 372
SFR(P) 370
SFT(P) 243
SFTBL(P) 384
SFTBR(P) 382
SFTWL(P) 388
SFTWR(P) 386
SHL(_E) 1302
SHR(_E) 1304
SIGNAL_SELECT 1447
SIMASK 546
SIN(_E) 1281
SIN(P) 882
SIND(P) 894
SJIS2WS(P) 808
SJIS2WSB(P) 810
SLAVE_OBJECT_ENCODER 1461
SLAVE_OBJECT_REAL 1460
SLAVE_OBJECT_VIRTUAL_ENCODER 1461
SORTD(_U) 719
SP.DEVST 608
SP.ECPRTCL 1139
SP.FCOPY 641
SP.FDELETE 638
SP.FMOVE 646
SP.FREAD 610
SP.FRENAME 651
SP.FSTATUS 655
SP.FTPGET 1171
SP.FTPPUT 1165
SP.FWRITE 624
SP.SLMPSND 1147
SP.SOCCINF 1128
SP.SOCCLOSE 1115
SP.SOCCSET 1131
SP.SOCOPEN 1110
SP.SOCRCV 1117
SP.SOCRMODE 1133
SP.SOCSND 1124
SP_DEVST 608
SP_ECPRTCL 1139
SPF(P) 789
SP_FCOPY 641
SP_FDELETE 638
SP_FMOVE 646
SP_FREAD 610
SP_FRENAME 651
SP_FSTATUS 655
SP_FTPGET 1171
SP_FTPPUT 1165
SP_FWRITE 624
SP_SLMPSND 1147
SP_SOCCINF 1128
SP_SOCCLOSE 1115
SP_SOCCSET 1131
SP_SOCOPEN 1110
SP_SOCRCV 1117
SP_SOCRMODE 1133
SP_SOCSND 1124
SQRT(_E) 1276
SQRT(P) 731
SR(_E) 1354
SRND(P) 959
S_SOCRCVS 1121
STMR 985
STOP 252
STR(P)(_U) 766
STRDEL(P) 827
STRINGMOV(P) 744
STRINGMOV(P)_WS 746
STRINGPLUS(P) 742
STRING_TO_BCD(_E) 1256
STRING_TO_BOOL(_E) 1249

STRING_TO_DINT(_E) 1254
STRING_TO_DWORD(_E) 1251
STRING_TO_INT(_E) 1252
STRING_TO_REAL(_E) 1258
STRING_TO_TIME(_E) 1261
STRING_TO_WORD(_E) 1250
STRINS(P) 825
SUB(_E) 1292
SUB_TIME(_E) 1347
SUM(P) 709
SWAP(P) 526
SYS_CMD 1433
SYS_CONST 1430
SYS_MONI 1432,1441
SYS_PRM 1431

T

TAN(_E) 1283
TAN(P) 886
TAND(P) 898
TARGET_REF 1444
TCMP(P) 1011
TEST(P) 364
TIMCHK 1021
TIME2SEC(P) 996
TIMER_100_FB_M 1379
TIMER_10_FB_M 1379
TIMER_CONT_FB_M 1379
TIMER_CONTHFB_M 1379
TIMER_HIGH_FB_M 1379
TIMER_LOW_FB_M 1379
TIME_TO_BOOL(_E) 1242
TIME_TO_DINT(_E) 1246
TIME_TO_DWORD(_E) 1244
TIME_TO_INT(_E) 1245
TIME_TO_STRING(_E) 1247
TIME_TO_WORD(_E) 1243
TO(P) 1032
TOD(P) 1041
TOF(_E) 1377
TON(_E) 1374
TP(_E) 1371
TTMR 983
TYPERD(P) 1046

U

UDCNT1 979
UDCNT2 981
UDINT2DBL(P) 869
UDINT2DINT(P) 436
UDINT2FLT(P) 862
UDINT2INT(P) 432
UDINT2UINT(P) 434
UINT2DBL(P) 867
UINT2DINT(P) 422
UINT2FLT(P) 859
UINT2INT(P) 420
UINT2UDINT(P) 424
UNI(P) 490
UNIINFRD(P) 1050

V

VAL(P)(_U) 468

W

WAND(P) 323,325
WDT(P) 549
WOR(P) 333,334
WORD_TO_BOOL(_E) 1185
WORD_TO_DINT(_E) 1188
WORD_TO_DWORD(_E) 1186
WORD_TO_INT(_E) 1187
WORD_TO_STRING(_E) 1191
WORD_TO_TIME(_E) 1190
WS2SJIS(P) 805
WSUM(P)(_U) 723
WTOB(P) 496
WXNR(P) 351,352
WXOR(P) 342,343

X

XCALL 580
XCH(P) 522
XOR(_E) 1310

Z

ZONE(P)(_U) 669
ZPOP(P) 963,968
ZPOP(P)_2 968
ZPUSH(P) 960,964
ZPUSH(P)_2 964
ZRRDB(P) 972
ZRWRB(P) 974

REVISIONS

*The manual number is given on the bottom left of the back cover.

Revision date	*Manual number	Description
May 2025	SH(NA)-082644ENG-A	First edition
May 2025	SH(NA)-082644ENG-B	Error correction
October 2025	SH(NA)-082644ENG-C	Added instructions • SCLN(P)(_U) • DSCLN(P)(_U) • EDSCLN(P) • SCL2N(P)(_U) • DSCL2N(P)(_U) Added motion control function blocks • MC_ReadStatus • MCv_PressureControl

Japanese manual number: SH-082643-C

This manual confers no industrial property rights or any rights of any other kind, nor does it confer any patent licenses. Mitsubishi Electric Corporation cannot be held responsible for any problems involving industrial property rights which may occur as a result of using the contents noted in this manual.

© 2025 MITSUBISHI ELECTRIC CORPORATION

CONDITIONS OF USE FOR THE PRODUCT

(1) MELSEC programmable controller ("the PRODUCT") shall be used in conditions;
i) where any problem, fault or failure occurring in the PRODUCT, if any, shall not lead to any major or serious accident; and
ii) where the backup and fail-safe function are systematically or automatically provided outside of the PRODUCT for the case of any problem, fault or failure occurring in the PRODUCT.

(2) The PRODUCT has been designed and manufactured for the purpose of being used in general industries.
MITSUBISHI ELECTRIC SHALL HAVE NO RESPONSIBILITY OR LIABILITY (INCLUDING, BUT NOT LIMITED TO ANY AND ALL RESPONSIBILITY OR LIABILITY BASED ON CONTRACT, WARRANTY, TORT, PRODUCT LIABILITY) FOR ANY INJURY OR DEATH TO PERSONS OR LOSS OR DAMAGE TO PROPERTY CAUSED BY the PRODUCT THAT ARE OPERATED OR USED IN APPLICATION NOT INTENDED OR EXCLUDED BY INSTRUCTIONS, PRECAUTIONS, OR WARNING CONTAINED IN MITSUBISHI ELECTRIC USER'S, INSTRUCTION AND/OR SAFETY MANUALS, TECHNICAL BULLETINS AND GUIDELINES FOR the PRODUCT.
("Prohibited Application")
Prohibited Applications include, but not limited to, the use of the PRODUCT in;
- Nuclear Power Plants and any other power plants operated by Power companies, and/or any other cases in which the public could be affected if any problem or fault occurs in the PRODUCT.
- Railway companies or Public service purposes, and/or any other cases in which establishment of a special quality assurance system is required by the Purchaser or End User.
- Aircraft or Aerospace, Medical applications, Train equipment, transport equipment such as Elevator and Escalator, Incineration and Fuel devices, Vehicles, Manned transportation, Equipment for Recreation and Amusement, and Safety devices, handling of Nuclear or Hazardous Materials or Chemicals, Mining and Drilling, and/or other applications where there is a significant risk of injury to the public or property.

Notwithstanding the above restrictions, Mitsubishi Electric may in its sole discretion, authorize use of the PRODUCT in one or more of the Prohibited Applications, provided that the usage of the PRODUCT is limited only for the specific applications agreed to by Mitsubishi Electric and provided further that no special quality assurance or fail-safe, redundant or other safety features which exceed the general specifications of the PRODUCTs are required. For details, please contact the Mitsubishi Electric representative in your region.

(3) Mitsubishi Electric shall have no responsibility or liability for any problems involving programmable controller trouble and system trouble caused by DoS attacks, unauthorized access, computer viruses, and other cyberattacks.

WARRANTY

Please confirm the following product warranty details before using this product.

1. Gratis Warranty Term and Gratis Warranty Range

If any faults or defects (hereinafter "Failure") found to be the responsibility of Mitsubishi occurs during use of the product within the gratis warranty term, the product shall be repaired at no cost via the sales representative or Mitsubishi Service Company.

However, if repairs are required onsite at domestic or overseas location, expenses to send an engineer will be solely at the customer's discretion. Mitsubishi shall not be held responsible for any re-commissioning, maintenance, or testing on-site that involves replacement of the failed module.

[Gratis Warranty Term]

The gratis warranty term of the product shall be for one year after the date of purchase or delivery to a designated place. Note that after manufacture and shipment from Mitsubishi, the maximum distribution period shall be six (6) months, and the longest gratis warranty term after manufacturing shall be eighteen (18) months. The gratis warranty term of repair parts shall not exceed the gratis warranty term before repairs.

[Gratis Warranty Range]

(1) The range shall be limited to normal use within the usage state, usage methods and usage environment, etc., which follow the conditions and precautions, etc., given in the instruction manual, user's manual and caution labels on the product.

(2) Even within the gratis warranty term, repairs shall be charged for in the following cases.

1. Failure occurring from inappropriate storage or handling, carelessness or negligence by the user. Failure caused by the user's hardware or software design.
2. Failure caused by unapproved modifications, etc., to the product by the user.
3. When the Mitsubishi product is assembled into a user's device, Failure that could have been avoided if functions or structures, judged as necessary in the legal safety measures the user's device is subject to or as necessary by industry standards, had been provided.
4. Failure that could have been avoided if consumable parts (battery, backlight, fuse, etc.) designated in the instruction manual had been correctly serviced or replaced.
5. Failure caused by external irresistible forces such as fires or abnormal voltages, and Failure caused by force majeure such as earthquakes, lightning, wind and water damage.
6. Failure caused by reasons unpredictable by scientific technology standards at time of shipment from Mitsubishi.
7. Any other failure found not to be the responsibility of Mitsubishi or that admitted not to be so by the user.

2. Onerous repair term after discontinuation of production

(1) Mitsubishi shall accept onerous product repairs for seven (7) years after production of the product is discontinued. Discontinuation of production shall be notified with Mitsubishi Technical Bulletins, etc.

(2) Product supply (including repair parts) is not available after production is discontinued.

3. Overseas service

Overseas, repairs shall be accepted by Mitsubishi's local overseas FA Center. Note that the repair conditions at each FA Center may differ.

4. Exclusion of loss in opportunity and secondary loss from warranty liability

Regardless of the gratis warranty term, Mitsubishi shall not be liable for compensation to:

(1) Damages caused by any cause found not to be the responsibility of Mitsubishi.

(2) Loss in opportunity, lost profits incurred to the user by Failures of Mitsubishi products.

(3) Special damages and secondary damages whether foreseeable or not, compensation for accidents, and compensation for damages to products other than Mitsubishi products.

(4) Replacement by the user, maintenance of on-site equipment, start-up test run and other tasks.

5. Changes in product specifications

The specifications given in the catalogs, manuals or technical documents are subject to change without prior notice.

INFORMATION AND SERVICES

For further information and services, please contact your local Mitsubishi Electric sales office or representative.
Visit our website to find our locations worldwide.

MITSUBISHI ELECTRIC Factory Automation Global Website
Locations Worldwide
www.MitsubishiElectric.com/fa/about-us/overseas/

TRADEMARKS

CiA® and CANopen® are registered EU trademarks of CiA e. V.
Microsoft, Microsoft Access, Excel, SQL Server, Visual Basic, Visual C++, Visual Studio, Windows, Windows NT, Windows Server, Windows Vista, and Windows XP are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
Unicode is either a registered trademark or a trademark of Unicode, Inc. in the United States and other countries.
The company names, system names and product names mentioned in this manual are either registered trademarks or trademarks of their respective companies.
In some cases, trademark symbols such as '™' or '®' are not specified in this manual.

2022

SH(NA)-082644ENG-C

TRANSLATION OF THE ORIGINAL INSTRUCTIONS

SH(NA)-082644ENG-C(2510)
MODEL: MXR-P-E

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE: TOKYO BLDG., 2-7-3, MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS: 1-14, YADA-MINAMI 5-CHOME, HIGASHI-KU, NAGOYA 461-8670, JAPAN

When exported from Japan, this manual does not require application to the Ministry of Economy, Trade and Industry for service transaction permission.

Specifications subject to change without notice.