MITSUBISHI ELECTRIC

Programmable Controller

MELSEC iQ-R series

MELSEC iQ-R Programming Manual (CPU Module Instructions, Standard Functions/ Function Blocks)

SAFETY PRECAUTIONS

(Read these precautions before using this product.)
Before using MELSEC iQ-R series programmable controllers, please read the manuals for the product and the relevant manuals introduced in those manuals carefully, and pay full attention to safety to handle the product correctly.
Make sure that the end users read this manual and then keep the manual in a safe place for future reference.

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.

- When the SIL2 Process CPU is used

(1) Although Mitsubishi Electric has declared Product's compliance with the international safety standards IEC61508, IEC61511, this fact does not guarantee that Product will be free from any malfunction or failure. The user of this Product shall comply with any and all applicable safety standard, regulation or law and take appropriate safety measures for the system in which the Product is installed or used and shall take the second or third safety measures other than the Product. Mitsubishi Electric is not liable for damages that could have been prevented by compliance with any applicable safety standard, regulation or law.
(2) Mitsubishi Electric prohibits the use of Products with or in any application involving, and Mitsubishi Electric shall not be liable for a default, a liability for defect warranty, a quality assurance, negligence or other tort and a product liability in these applications.
(a) power plants,
(b) trains, railway systems, airplanes, airline operations, other transportation systems,
(c) hospitals, medical care, dialysis and life support facilities or equipment,
(d) amusement equipments,
(e) incineration and fuel devices,
(f) handling of nuclear or hazardous materials or chemicals,
(g) mining and drilling,
(h) and other applications where the level of risk to human life, health or property are elevated.
(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.

- When the Safety CPU is used

(1) Although Mitsubishi Electric has obtained the certification for Product's compliance to the international safety standards IEC61508, ISO13849-1 from TUV Rheinland, this fact does not guarantee that Product will be free from any malfunction or failure. The user of this Product shall comply with any and all applicable safety standard, regulation or law and take appropriate safety measures for the system in which the Product is installed or used and shall take the second or third safety measures other than the Product. Mitsubishi Electric is not liable for damages that could have been prevented by compliance with any applicable safety standard, regulation or law.
(2) Mitsubishi Electric prohibits the use of Products with or in any application involving, and Mitsubishi Electric shall not be liable for a default, a liability for defect warranty, a quality assurance, negligence or other tort and a product liability in these applications.
(a) power plants,
(b) trains, railway systems, airplanes, airline operations, other transportation systems,
(c) hospitals, medical care, dialysis and life support facilities or equipment,
(d) amusement equipments,
(e) incineration and fuel devices,
(f) handling of nuclear or hazardous materials or chemicals,
(g) mining and drilling,
(h) and other applications where the level of risk to human life, health or property are elevated.
(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.

INTRODUCTION

Thank you for purchasing the Mitsubishi Electric MELSEC iQ-R series programmable controllers.
This manual describes the instructions and standard 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 MELSEC iQ-R series programmable 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
CONDITIONS OF USE FOR THE PRODUCT . . . 1
INTRODUCTION . . . 3
RELEVANT MANUALS . . . 21
TERMS . . . 22
GENERIC TERMS AND ABBREVIATIONS . . . 24
MANUAL PAGE ORGANIZATION . . . 26

PART 1 OVERVIEW

CHAPTER 1 OVERVIEW 32

1.1 Instruction Configuration . . . 32
1.2 Data Specification Method . . . 34
Bit data . . . 38
16-bit data (word data) . . . 40
32-bit data (double word data) . . . 43
Real number data (floating-point data) . . . 46
String data . . . 49
1.3 Execution Condition . . . 51
1.4 High-speed Instruction Processing . . . 52
Subset processing . . . 52
1.5 Precautions on Programming . . . 53
Errors common to instructions . . . 53
Checking the ranges of instruction runtime devices and labels . . . 53
Operation when a long timer or long retentive timer device is used . . . 56
Operations arising when the OUT, SET/RST, and PLS/PLF instructions of the same device are used . . . 58
Restrictions on using file registers . . . 64

PART 2 LISTS OF INSTRUCTIONS AND FUN/FB

CHAPTER 2 CPU MODULE INSTRUCTIONS 66

2.1 Sequence Instructions . . . 66
2.2 Basic Instructions . . . 70
2.3 Application Instructions . . . 92
Program control . . . 92
Data processing . . . 94
Debugging and failure diagnostic . . . 102
String processing . . . 102
Real value processing . . . 105
Random number . . . 112
Device operation . . . 113
Timer, counter . . . 114
Shortcut control . . . 115
Ramp signal . . . 115
Matrix input . . . 115
CPU module database access function . . . 116

Clock . . . 117
Module access. . . . 120
Parameter setting operation . . . 122
CPU module data logging function . . . 122
Recording function. . . . 122
Built-in Ethernet function instructions . . . 123
PID operation instruction . . . 125
PID control instructions . . . 125
Process control instructions. . . . 127
Multiple CPU dedicated instructions . . . 128
SFC program instructions . . . 129
Redundant system instructions . . . 131
Safety system instructions. . . . 132

CHAPTER 3 MODULE DEDICATED INSTRUCTIONS 134

CHAPTER 4 STANDARD FUNCTIONS/FUNCTION BLOCKS 136

4.1 Standard Functions. . . . 136
4.2 Standard Function Blocks . . . 147

PART 3 SEQUENCE INSTRUCTIONS

CHAPTER 5 SEQUENCE INSTRUCTIONS 150

5.1 Contact Instructions . . . 150
Operation start, series connection, parallel connection. . . . 150
Pulse operation start, pulse series connection, pulse parallel connection . . . 153
Pulse NOT operation start, pulse NOT series connection, pulse NOT parallel connection. . . . 156
5.2 Association Instructions. . . . 159
Ladder block series/parallel connection . . . 159
Storing/reading/clearing the operation result. . . . 160
Inverting the operation result. . . . 162
Converting the operation result into a pulse . . . 163
Converting the edge relay operation result into a pulse . . . 164
5.3 Output Instructions . . . 166
Out (excluding the timer, counter, and annunciator) . . . 166
Timer . . . 168
Long timer . . . 171
Counter . . . 174
Long counter . . . 176
Annunciator . . . 178
Setting devices (excluding annunciator) . . . 179
Resetting devices (excluding annunciator) . . . 181
Setting annunciator . . . 183
Resetting annunciator . . . 185
Rising edge output. . . . 187
Falling edge output . . . 189
Inverting the bit device output . . . 191
Converting the direct access output into a pulse. . . . 193
5.4 Shift Instructions . . . 195
Shifting bit devices. . . . 195

5.5 Master Control Instructions . . . 197
Setting/resetting a master control . . . 197
5.6 Phase Processing Instructions . . . 201
Overview . . . 201
Starting the phase processing . . . 202
Changing the execution phase . . . 204
Terminating the execution phase . . . 206
5.7 Termination Instructions . . . 207
Ending the main routine program . . . 207
Ending the sequence program . . . 208
5.8 Stop Instruction . . . 210
Stopping the sequence program . . . 210
5.9 No Operation Instruction . . . 211
No operation (NOP) . . . 211
No operation (NOPLF) . . . 212

PART 4 BASIC INSTRUCTIONS

CHAPTER 6 BASIC INSTRUCTIONS 214
6.1 Comparison Operation Instructions . . . 214
Comparing 16-bit binary data . . . 214
Comparing 32-bit binary data . . . 216
Outputting a comparison result of 16-bit binary data . . . 218
Outputting a comparison result of 32-bit binary data . . . 220
Outputting a band comparison result of 16-bit binary data . . . 222
Outputting a band comparison result of 32-bit binary data . . . 224
Comparing 16-bit binary block data . . . 226
Comparing 32-bit binary block data . . . 228
6.2 Arithmetic Operation Instructions . . . 231
Adding 16-bit binary data . . . 231
Subtracting 16-bit binary data . . . 235
Adding 32-bit binary data . . . 239
Subtracting 32-bit binary data . . . 243
Multiplying 16-bit binary data . . . 247
Dividing 16-bit binary data . . . 249
Multiplying 32-bit binary data . . . 251
Dividing 32-bit binary data . . . 253
Adding BCD 4-digit data . . . 255
Subtracting BCD 4-digit data . . . 258
Adding BCD 8-digit data . . . 261
Subtracting BCD 8-digit data . . . 265
Multiplying BCD 4-digit data . . . 269
Dividing BCD 4-digit data . . . 271
Multiplying BCD 8-digit data . . . 273
Dividing BCD 8-digit data . . . 275
Adding 16-bit binary block data . . . 277
Subtracting 16-bit binary block data . . . 279
Adding 32-bit binary block data . . . 281
Subtracting 32-bit binary block data . . . 284
Incrementing 16-bit binary data . . . 287

Decrementing 16-bit binary data . . . 289
Incrementing 32-bit binary data . . . 291
Decrementing 32-bit binary data . . . 293
6.3 Logical Operation Instructions . . . 295
Performing an AND operation on 16-bit data. . . . 295
Performing an AND operation on 32-bit data. . . . 299
Performing an AND operation on 16-bit block data . . . 303
Performing an OR operation on 16-bit data. . . . 305
Performing an OR operation on 32-bit data. . . . 309
Performing an OR operation on 16-bit block data . . . 313
Performing an XOR operation on 16-bit data . . . 315
Performing an XOR operation on 32-bit data . . . 319
Performing an XOR operation on 16-bit block data. . . . 323
Performing an XNOR operation on 16-bit data . . . 325
Performing an XNOR operation on 32-bit data . . . 329
Performing an XNOR operation on 16-bit block data . . . 333
6.4 Bit Processing Instructions . . . 335
Setting a bit in the word device . . . 335
Resetting a bit in the word device . . . 337
Performing a 16-bit test . . . 339
Performing a 32-bit test . . . 341
Batch-resetting bit devices . . . 343
6.5 Shift Instructions . . . 345
Shifting 16-bit binary data to the right by n bit(s). . . . 345
Shifting 16-bit binary data to the left by n bit(s) . . . 347
Shifting n-bit data to the right by one bit . . . 349
Shifting n-bit data to the left by one bit . . . 351
Shifting n-word data to the right by one word . . . 353
Shifting n-word data to the left by one word . . . 355
Shifting n double word(s) of data to the right by one double word . . . 357
Shifting n double word(s) of data to the left by one double word. . . . 359
Shifting n point(s) of single-precision real number data to the right by one point . . . 361
Shifting n point(s) of single-precision real number data to the left by one point . . . 363
Shifting n point(s) of double-precision real number data to the right by one point . . . 365
Shifting n point(s) of double-precision real number data to the left by one point. . . . 367
Shifting n-bit data to the right by n bit(s) . . . 369
Shifting n-bit data to the left by n bit(s) . . . 373
Shifting n-word data to the right by n word(s) . . . 377
Shifting n-word data to the left by n word(s) . . . 381
Shifting n double word(s) of data to the right by n double word(s) . . . 385
Shifting n double word(s) of data to the left by n double word(s) . . . 389
Shifting n point(s) of single-precision real number data to the right by n point(s) . . . 393
Shifting n point(s) of single-precision real number data to the left by n point(s) . . . 397
Shifting n point(s) of double-precision real number data to the right by n point(s) . . . 401
Shifting n point(s) of double-precision real number data to the left by n point(s) . . . 405
6.6 Data Conversion Instructions . . . 409
Converting binary data to BCD 4-digit data. . . . 409
Converting binary data to BCD 8-digit data. . . . 411
Converting BCD 4-digit data to 16-bit binary data. . . . 413
Converting BCD 8-digit data to 32-bit binary data. . . . 415
Converting single-precision real number to 16-bit signed binary data. . . . 417

Converting single-precision real number to 16-bit unsigned binary data . . . 419
Converting single-precision real number to 32-bit signed binary data . . . 421
Converting single-precision real number to 32-bit unsigned binary data . . . 423
Converting double-precision real number to 16-bit signed binary data . . . 425
Converting double-precision real number to 16-bit unsigned binary data . . . 427
Converting double-precision real number to 32-bit signed binary data . . . 429
Converting double-precision real number to 32-bit unsigned binary data . . . 431
Converting 16-bit signed binary data to 16-bit unsigned binary data . . . 433
Converting 16-bit signed binary data to 32-bit signed binary data . . . 435
Converting 16-bit signed binary data to 32-bit unsigned binary data . . . 437
Converting 16-bit unsigned binary data to 16-bit signed binary data . . . 439
Converting 16-bit unsigned binary data to 32-bit signed binary data . . . 441
Converting 16-bit unsigned binary data to 32-bit unsigned binary data . . . 443
Converting 32-bit signed binary data to 16-bit signed binary data . . . 445
Converting 32-bit signed binary data to 16-bit unsigned binary data . . . 447
Converting 32-bit signed binary data to 32-bit unsigned binary data . . . 449
Converting 32-bit unsigned binary data to 16-bit signed binary data . . . 451
Converting 32-bit unsigned binary data to 16-bit unsigned binary data . . . 453
Converting 32-bit unsigned binary data to 32-bit signed binary data . . . 455
Converting 16-bit binary data to Gray code data . . . 457
Converting 32-bit binary data to Gray code data . . . 459
Converting 16-bit binary Gray code data to 16-bit binary data . . . 461
Converting 32-bit binary Gray code data to 32-bit binary data . . . 463
Converting 16-bit binary data block to BCD 4-digit data block . . . 465
Converting BCD 4-digit block data to 16-bit binary block data . . . 467
Converting decimal ASCII data to 16-bit binary data . . . 469
Converting decimal ASCII data to 32-bit binary data . . . 472
Converting hexadecimal ASCII data to 16-bit binary data . . . 476
Converting hexadecimal ASCII data to 32-bit binary data . . . 479
Converting decimal ASCII data to BCD 4-digit data . . . 482
Converting decimal ASCII data to BCD 8-digit data . . . 485
Converting decimal string data to 16-bit binary data . . . 488
Converting decimal string data to 32-bit binary data . . . 491
Converting hexadecimal ASCII to hexadecimal binary data . . . 494
Converting single-precision real number to BCD format data . . . 496
Two's complement of 16-bit binary data (sign inversion) . . . 498
Two's complement of 32-bit binary data (sign inversion) . . . 500
Decoding 8-bit data to 256-bit data . . . 502
Encoding 256-bit data to 8-bit data . . . 504
Decoding data to seven-segment display data . . . 506
Separating data in units of 4 bits . . . 509
Combining data in units of 4 bits . . . 511
Separating data in units of bits . . . 513
Combining data in units of bits . . . 515
Separating data in units of bytes . . . 517
Combining data in units of bytes . . . 519
6.7 Data Transfer Instructions . . . 521
Transferring 16-bit binary data . . . 521
Transferring 32-bit binary data . . . 523
Inverting and transferring 16-bit binary data . . . 525
Inverting and transferring 32-bit binary data . . . 527

Shifting data in units of 4 bits. 529
Inverting and transferring 1-bit data. 532
Transferring 16-bit binary data block (16 bits) 534
Transferring 16-bit binary data block (32 bits) 537
Transferring the same 16-bit binary data block (16 bits) 539
Transferring the same 16-bit binary data block (32 bits) 541
Transferring the same 32-bit binary data block (16 bits) 543
Transferring the same 32-bit binary data block (32 bits) 545
Exchanging 16-bit binary data 547
Exchanging 32-bit binary data 549
Exchanging 16-bit binary block data 551
Exchanging the upper and lower bytes of 16-bit binary data. 553
Exchanging the upper and lower bytes of 32-bit binary data. 554
Transferring 1-bit data 556
Transferring n-bit data 558

PART 5 APPLICATION INSTRUCTIONS

CHAPTER 7 PROGRAM CONTROL 563

7.1 Program Branch Instructions. 563
Pointer branch 563
Jumping to END 566
7.2 Program Execution Control Instructions 567
Disabling/enabling interrupt programs. 567
Disabling interrupt programs with specified priority or lower 570
Interrupt program mask 575
Disabling/enabling the specified interrupt pointer 577
Returning from the interrupt program 579
Resetting the watchdog timer 580
7.3 Structure Creation Instructions 581
Performing the FOR to NEXT instruction loop. 581
Forcibly terminating the FOR to NEXT instruction loop. 583
Calling a subroutine program 585
Returning from the subroutine program called 589
Calling a subroutine program and turning the output off 590
Calling a subroutine program in the specified program file 594
Calling a subroutine program in the specified program file and turning the output off 599
Calling a subroutine program 604
7.4 Program Control Instructions 609
Changing the program execution type to standby type 609
Changing the program execution type to standby type (output off). 611
Changing the program execution type to scan execution type 613

CHAPTER 8 DATA PROCESSING 615

8.1 Rotation Instructions 615
Rotating 16-bit binary data to the right 615
Rotating 16-bit binary data to the left. 618
Rotating 32-bit binary data to the right 621
Rotating 32-bit binary data to the left. 623

8.2 Data Table Operation Instructions . . . 625
Reading the oldest data from the data table . . . 625
Reading the newest data from the data table . . . 627
Writing data to the data table. . . . 629
Inserting data to the data table . . . 631
Deleting data from data table . . . 633
8.3 Reading/Writing Data Instructions. . . . 635
Reading data from the data memory. . . . 636
Writing data to the data memory . . . 638
8.4 File Operation Instructions. . . . 641
Reading data from the specified file . . . 641
Writing data to the specified file. . . . 660
Deleting the specified file. . . . 678
Copying the specified file. . . . 682
Moving the specified file . . . 687
Renaming the specified file . . . 692
Acquiring the status of the specified file . . . 696
Error codes generated for file operation instructions. . . . 700
8.5 Data Control Instructions . . . 701
Upper and lower limit control of 16-bit binary data . . . 701
Upper and lower limit control of 32-bit binary data . . . 703
Dead band control of 16-bit binary data . . . 705
Dead band control of 32-bit binary data . . . 707
Zone control of 16-bit binary data . . . 709
Zone control of 32-bit binary data . . . 711
Scaling 16-bit binary data (point coordinates). . . . 713
Scaling 32-bit binary data (point coordinates). . . . 716
Scaling 16-bit binary data (XY coordinates) . . . 719
Scaling 32-bit binary data (XY coordinates) . . . 721
8.6 Data Processing Instructions. . . . 723
Searching 16-bit binary data . . . 723
Searching 32-bit binary data . . . 725
Searching 16-bit binary data (minimum, match, maximum) . . . 727
Searching 32-bit binary data (minimum, match, maximum) . . . 729
Checking 16-bit binary data. . . . 731
Checking 32-bit binary data. . . . 733
Checking the bit status in 16-bit binary data . . . 735
Checking the bit status in 32-bit binary data . . . 737
Searching the maximum value of 16-bit binary data . . . 739
Searching the maximum value of 32-bit binary data . . . 741
Searching the minimum value of 16-bit binary data . . . 743
Searching the minimum value of 32-bit binary data . . . 745
Sorting 16-bit binary data . . . 747
Sorting 32-bit binary data . . . 749
Sorting 16-bit binary data table . . . 751
Sorting 16-bit binary data table 2. . . . 755
Sorting 32-bit binary data table 2. . . . 759
Adding 16-bit binary data. . . . 763
Adding 32-bit binary data. . . . 765
Calculating the mean value of 16-bit binary data . . . 767
Calculating the mean value of 32-bit binary data . . . 769

Calculating the square root of 16-bit binary data . . . 771
Calculating the square root of 32-bit binary data . . . 773
CRC operation . . . 774
8.7 Check Code Instructions . . . 776
Check code . . . 776

CHAPTER 9 DEBUGGING AND FAILURE DIAGNOSTIC 779

9.1 Debugging and Failure Diagnostic Instructions . . . 779
Resetting the error display and the annunciator display . . . 779
Generating a continuation error . . . 781
Generating a stop error . . . 783

CHAPTER 10 STRING PROCESSING 785

10.1 String Processing Instructions . . . 785
Comparing string data . . . 785
Concatenating string data . . . 788
Transferring string data . . . 792
Transferring Unicode string data . . . 794
Converting 16-bit binary data to decimal ASCII . . . 796
Converting 32-bit binary data to decimal ASCII . . . 800
Converting 16-bit binary data to hexadecimal ASCII . . . 805
Converting 32-bit binary data to hexadecimal ASCII . . . 809
Converting 16-bit binary data to string data . . . 813
Converting 32-bit binary data to string data . . . 816
Converting BCD 4-digit data to decimal ASCII code . . . 819
Converting BCD 8-digit data to decimal ASCII code . . . 823
Converting single-precision real number to string data . . . 828
Converting hexadecimal binary data to hexadecimal ASCII code . . . 832
Converting Unicode character string to Shift JIS character string . . . 834
Converting shift JIS character string to Unicode character string (without byte order mark) . . . 836
Converting shift JIS character string to Unicode (with byte order mark) . . . 838
Detecting a string length . . . 840
Extracting string data from the right . . . 842
Extracting string data from the left . . . 844
Extracting the specified string data . . . 846
Replacing the specified string data . . . 848
Searching string data . . . 851
Inserting string data . . . 853
Deleting string data . . . 855

CHAPTER 11 REAL VALUE PROCESSING 857

11.1 Floating-point instruction . . . 857
Comparing single-precision real numbers . . . 857
Comparing double-precision real numbers . . . 859
Outputting a comparison result of single-precision real numbers . . . 862
Outputting a comparison result of double-precision real numbers . . . 864
Outputting a band comparison result of single-precision real number . . . 866
Outputting a band comparison result of double-precision real number . . . 868
Adding single-precision real numbers . . . 870
Subtracting single-precision real numbers . . . 874

Adding double-precision real numbers 878
Subtracting double-precision real numbers 882
Multiplying single-precision real numbers 886
Dividing single-precision real numbers 888
Multiplying double-precision real numbers 890
Dividing double-precision real numbers 892
Converting 16-bit signed binary data to single-precision real number 894
Converting 16-bit unsigned binary data to single-precision real number 896
Converting 32-bit signed binary data to single-precision real number 898
Converting 32-bit unsigned binary data to single-precision real number 900
Converting double-precision real number to single-precision real number 902
Converting 16-bit signed binary data to double-precision real number 904
Converting 16-bit unsigned binary data to double-precision real number 906
Converting 32-bit signed binary data to double-precision real number 908
Converting 32-bit unsigned binary data to double-precision real number 910
Converting single-precision real number to double-precision real number 912
Converting string data to single-precision real number 914
Converting BCD format data to single-precision real number 918
Inverting the sign of single-precision real number 920
Inverting the sign of double-precision real number 921
Transferring single-precision real number 922
Transferring double-precision real number 923
Calculating the sine of single-precision real number 924
Calculating the cosine of single-precision real number 926
Calculating the tangent of single-precision real number 928
Calculating the arc sine of single-precision real number 930
Calculating the arc cosine of single-precision real number 932
Calculating the arc tangent of single-precision real number 934
Calculating the sine of double-precision real number 936
Calculating the cosine of double-precision real number 938
Calculating the tangent of double-precision real number 940
Calculating the arc sine of double-precision real number 942
Calculating the arc cosine of double-precision real number 944
Calculating the arc tangent of double-precision real number 946
Calculating the sine of BCD data 948
Calculating the cosine of BCD data 950
Calculating the tangent of BCD data 952
Calculating the arc sine of BCD data 954
Calculating the arc cosine of BCD data 956
Calculating the arc tangent of BCD data 958
Converting single-precision real number angle to radian 960
Converting single-precision real number radian to angle 962
Converting double-precision real number angle to radian 964
Converting double-precision real number radian to angle 966
Calculating the square root of single-precision real number 968
Calculating the square root of double-precision real number 970
Calculating the exponent of single-precision real number 972
Calculating the exponent of double-precision real number 974
Calculating the natural logarithm of single-precision real number 976
Calculating the natural logarithm of double-precision real number 978
Calculating the square root of BCD 4-digit data 980

Calculating the square root of BCD 8-digit data 982
Calculating the exponentiation of single-precision real number 984
Calculating the exponentiation of double-precision real number 986
Calculating the common logarithm of single-precision real number 988
Calculating the common logarithm of double-precision real number 990
Searching the maximum value of single-precision real number 992
Searching the maximum value of double-precision real number 994
Searching the minimum value of single-precision real number 996
Searching the minimum value of double-precision real number 998

CHAPTER 12 RANDOM NUMBER 1000

12.1 Random Number Instructions 1000
Generating random number 1000
Changing random sequence 1001

CHAPTER 13 DEVICE OPERATION 1002

13.1 Index Register Instructions 1002
Saving all data of the index register 1002
Returning all data of the index register 1004
Saving the selected data of the index register and long index register 1005
Returning the selected data of the index register and long index register 1008
13.2 File Register Operation Instructions 1010
Switching the file register block number 1010
Changing the file register file name 1012
13.3 File Register Read/Write Instructions 1014
Reading 1-byte data from the file register 1014
Writing 1-byte data to the file register 1016
13.4 Indirect Address Read Instructions 1018
Reading the indirect address 1018

CHAPTER 14 TIMER, COUNTER 1020

14.1 Special Counter Instructions 1020
Counting up or down the current value (1-phase input) 1020
Counting up or down the current value (2-phase input) 1023
14.2 Special Timer Instructions 1025
Teaching timer 1025
Special function timer 1027
14.3 Pulse Related Instructions 1030
Measuring the density of pulses 1030
Outputting pulses at regular intervals 1032
Performing the pulse width modulation 1034

CHAPTER 15 SHORTCUT CONTROL 1036

15.1 Shortcut Control Instruction 1036
Rotary table shortest direction control 1036

CHAPTER 16 RAMP SIGNAL 1039

16.1 Ramp Signal Instruction 1039
Ramp signal 1039

CHAPTER 17 MATRIX INPUT 1042

17.1 Matrix Input Instruction. 1042

Matrix input 1042

CHAPTER 18 CPU MODULE DATABASE ACCESS FUNCTION 1045

18.1 Database Access Instructions 1045

Importing data to the data base. 1045

Exporting data from the data base 1048

Opening the data base 1051

Closing the data base 1053

Adding a record to the data base 1055

Updating the record in the data base 1062

Searching the record in the data base. 1068

Deleting the record in the data base 1076

Starting a transaction. 1080

Committing a transaction. 1082

Performing a database rollback. 1084

Error codes related to database access instructions. 1086

CHAPTER 19 CLOCK 1090

19.1 Clock Instructions 1090

Reading clock data 1090

Writing clock data 1092

Adding clock data 1094

Subtracting clock data 1096

Converting time data from hour/minute/second to second 1098

Converting time data from second to hour/minute/second 1100

Converting date and time data from date and time to second. 1102

Converting date and time data from second to date and time. 1104

Comparing date data. 1106

Comparing time data 1110

Outputting a comparison result of time data 1113

Outputting a band comparison result of time data. 1115

Reading expansion clock data. 1117

Adding expansion clock data. 1119

Subtracting expansion clock data 1121

19.2 Timing Check Instructions 1123

Generating timing pulses. 1123

Measuring time of the specified data. 1125

Hour meter. 1127

CHAPTER 20 MODULE ACCESS 1131

20.1 Module Access Instructions. 1131

Performing I/O refresh. 1131

Selecting refresh to be performed. 1133

Performing module refresh 1135

Reading 1-word/2-word data from another module (16-bit specification) 1137

Writing 1-word/2-word data to a module (16-bit specification) 1141

Reading 1-word/2-word data from another module (32-bit specification) 1146

Writing 1-word/2-word data to a module (32-bit specification) 1150

Reading the module model name 1155
Reading module specific information. 1159

CHAPTER 21 PARAMETER SETTING OPERATION 1164

21.1 Routing Information Instructions. 1164
Reading routing information 1164
Registering routing information 1166

CHAPTER 22 CPU MODULE DATA LOGGING FUNCTION 1168

22.1 Logging Instructions. 1168
Setting trigger logging 1168
Resetting trigger logging 1170

CHAPTER 23 RECORDING FUNCTION 1172

23.1 Data Collection Instruction. 1172
Setting data collection trigger 1172

CHAPTER 24 BUILT-IN ETHERNET FUNCTION INSTRUCTIONS 1174

24.1 Open/Close Processing Instructions. 1174
Opening a connection 1174
Closing a connection 1177
24.2 Socket Communications Instructions 1179
Reading receive data during the END processing. 1179
Reading receive data when the instruction is executed 1183
Sending data 1185
Reading connection information 1188
Changing the communication target (UDP/IP) 1190
Changing the receive mode. 1192
Reading socket communications receive data 1196
24.3 Predefined Protocol Support Function Instruction 1198
Executing the registered protocols 1198
24.4 SLMP Frame Send Instruction 1206
Sending an SLMP frame 1206
24.5 File Transfer Function Instructions 1213
Sending FTP client files. 1213
Retrieving FTP client files 1223

CHAPTER 25 PID OPERATION INSTRUCTION 1234

25.1 Overview. 1234
Control data 1236
Auto tuning 1243
25.2 PID Operation Instruction 1245

CHAPTER 26 PID CONTROL INSTRUCTIONS 1249

26.1 Overview. 1249
Operation method 1249
PID control procedure 1250
Helpful functions 1256
26.2 PID Control Instructions (Inexact Differential) 1258
Registering the PID control data to the CPU module 1260

Performing PID operation 1263
Stopping the operation of specified loop number 1266
Starting the operation of specified loop number 1267
Changing the parameters of specified loop number 1268
26.3 PID Control Instructions (Exact Differential) 1270
Registering the PID control data to the CPU module 1272
Performing PID operation 1274
Stopping the operation of specified loop number 1277
Starting the operation of specified loop number 1278
Changing the parameters of specified loop number 1279

CHAPTER 27 MULTIPLE CPU DEDICATED INSTRUCTIONS 1281

27.1 Another CPU Module Access Instructions 1281
Reading device data from another CPU module 1285
Writing device data to another CPU module 1288

CHAPTER 28 SFC PROGRAM INSTRUCTIONS 1291

28.1 SFC Control Instructions 1291
Checking the status of a step 1291
Checking the status of a block. 1294
Batch-reading the status of steps 1296
Starting a block 1305
Ending a block 1307
Pausing a block 1309
Restarting a block 1311
Activating a step 1313
Deactivating a step 1315
Switching a target block. 1317
28.2 SFC Dedicated Instruction 1319
Creating a dummy transition condition 1319

CHAPTER 29 REDUNDANT SYSTEM INSTRUCTIONS 1320

29.1 System Switching 1320
29.2 Disabling/Enabling System Switching 1324
29.3 Writing Data from the Standby System to the Control System. 1326

CHAPTER 30 SAFETY SYSTEM INSTRUCTIONS 1331

30.1 Reading Safety Data Identify Check Information 1331
Error codes generated by safety system instructions 1333

PART 6 MODULE DEDICATED INSTRUCTIONS

CHAPTER 31 MODULE DEDICATED INSTRUCTIONS 1336

PART 7 STANDARD FUNCTIONS

CHAPTER 32 TYPE CONVERSION FUNCTIONS 1340

32.1 Converting BOOL to WORD 1340
32.2 Converting BOOL to DWORD. 1342

32.3 Converting BOOL to INT 1343
32.4 Converting BOOL to DINT 1344
32.5 Converting BOOL to TIME 1345
32.6 Converting BOOL to STRING 1346
32.7 Converting WORD to BOOL 1347
32.8 Converting WORD to DWORD 1348
32.9 Converting WORD to INT 1349
32.10 Converting WORD to DINT 1350
32.11 Converting WORD to TIME 1352
32.12 Converting WORD to STRING 1353
32.13 Converting DWORD to BOOL 1354
32.14 Converting DWORD to WORD 1355
32.15 Converting DWORD to INT 1357
32.16 Converting DWORD to DINT 1359
32.17 Converting DWORD to TIME 1360
32.18 Converting DWORD to STRING 1361
32.19 Converting INT to BOOL 1362
32.20 Converting INT to WORD 1363
32.21 Converting INT to DWORD 1364
32.22 Converting INT to DINT 1366
32.23 Converting INT to BCD 1367
32.24 Converting INT to REAL 1369
32.25 Converting INT to LREAL 1370
32.26 Converting INT to TIME 1371
32.27 Converting INT to STRING 1372
32.28 Converting DINT to BOOL 1374
32.29 Converting DINT to WORD 1375
32.30 Converting DINT to DWORD 1377
32.31 Converting DINT to INT 1378
32.32 Converting DINT to BCD 1379
32.33 Converting DINT to REAL 1381
32.34 Converting DINT to LREAL 1382
32.35 Converting DINT to TIME 1383
32.36 Converting DINT to STRING 1384
32.37 Converting BCD to INT 1386
32.38 Converting BCD to DINT 1388
32.39 Converting BCD to STRING 1391
32.40 Converting REAL to INT 1393
32.41 Converting REAL to DINT 1395
32.42 Converting REAL to LREAL 1397
32.43 Converting REAL to STRING 1399
32.44 Converting LREAL to INT 1402
32.45 Converting LREAL to DINT 1404
32.46 Converting LREAL to REAL 1406
32.47 Converting TIME to BOOL 1408
32.48 Converting TIME to WORD 1409
32.49 Converting TIME to DWORD 1410
32.50 Converting TIME to INT 1411
32.51 Converting TIME to DINT 1412
32.52 Converting TIME to STRING 1413
32.53 Converting STRING to BOOL 1415

32.54 Converting STRING to WORD 1416
32.55 Converting STRING to DWORD 1417
32.56 Converting STRING to INT 1418
32.57 Converting STRING to DINT 1420
32.58 Converting STRING to BCD 1422
32.59 Converting STRING to REAL 1424
32.60 Converting STRING to TIME 1427
32.61 Converting Bit Array to INT 1429
32.62 Converting Bit Array to DINT 1430
32.63 Converting INT to Bit Array 1431
32.64 Converting DINT to Bit Array 1432
32.65 Copying the Bit Array 1433
32.66 Reading the Specified Bit of the Word Label 1434
32.67 Writing the Specified Bit of the Word Label 1436
32.68 Copying the Specified Bit of the Word Label 1438
32.69 Getting the Start Data 1440

CHAPTER 33 SINGLE VARIABLE FUNCTIONS 1441

33.1 Calculating the Absolute Value 1441
33.2 Calculating the Square Root 1443
33.3 Calculating the Natural Logarithm 1444
33.4 Calculating the Common Logarithm 1445
33.5 Calculating the Exponent 1447
33.6 Calculating the Sine 1448
33.7 Calculating the Cosine 1449
33.8 Calculating the Tangent 1450
33.9 Calculating the Arc Sine 1451
33.10 Calculating the Arc Cosine 1452
33.11 Calculating the Arc Tangent 1453

CHAPTER 34 ARITHMETIC OPERATION FUNCTIONS 1454

34.1 Addition 1454
34.2 Multiplication 1457
34.3 Subtraction 1459
34.4 Division 1462
34.5 Remainder 1464
34.6 Exponentiation 1466
34.7 Assignment (Move Operation) 1467

CHAPTER 35 BIT SHIFT FUNCTIONS 1469

35.1 Shifting Data to the Left by n Bit(s) 1469
35.2 Shifting Data to the Right by n Bit(s) 1471
35.3 Rotating Data to the Left by n Bit(s) 1473
35.4 Rotating Data to the Right by n Bit(s) 1475

CHAPTER 36 BOOLEAN FUNCTIONS 1477

36.1 AND Operation, OR Operation, and XOR Operation 1477
36.2 NOT Operation 1480

CHAPTER 37 SELECTION FUNCTIONS 1481

37.1 Selecting a Value 1481
37.2 Selecting the Maximum/Minimum Value 1483
37.3 Controlling the Upper/Lower Limit 1485
37.4 Multiplexer 1488

CHAPTER 38 COMPARISON FUNCTIONS 1490

38.1 Comparing Data 1490
38.2 Comparing Data 1492

CHAPTER 39 STRING FUNCTIONS 1494

39.1 Detecting a String Length 1494
39.2 Extracting String Data From the Left/Right 1496
39.3 Extracting String Data 1498
39.4 Concatenating String Data 1500
39.5 Inserting String Data 1502
39.6 Deleting String Data 1504
39.7 Replacing String Data 1506
39.8 Searching String Data 1509

CHAPTER 40 TIME DATA TYPE FUNCTIONS 1511

40.1 Addition 1511
40.2 Subtraction 1513
40.3 Multiplication 1515
40.4 Division 1517

PART 8 STANDARD FUNCTION BLOCKS

CHAPTER 41 BISTABLE FUNCTION BLOCKS 1520

41.1 Bistable Function Block (Set-Dominant) 1520
41.2 Bistable Function Block (Reset-Dominant) 1522

CHAPTER 42 EDGE DETECTION FUNCTION BLOCKS 1524

42.1 Detecting a Rising Edge 1524
42.2 Detecting a Falling Edge 1526

CHAPTER 43 COUNTER/TIMER FUNCTION BLOCKS 1528

43.1 Up Counter 1528
43.2 Down Counter 1530
43.3 Up/Down Counter 1532
43.4 Counter Function Block 1535
43.5 Pulse Timer 1537
43.6 On Delay Timer 1540
43.7 Off Delay Timer 1543
43.8 Timer Function Block 1545

APPENDICES 1549

Appendix 1 Instruction Processing Time 1549
Time added to instruction processing time 1585

Appendix 2 Number of Basic Steps and Availability of Subset Processing 1586
Appendix 3 Determining Three PID Constants 1611
Appendix 4 PID Operation Program Examples 1613
Auto tuning (step response method) + PID control program example 1614
Auto tuning (step response method) program example 1616
Appendix 5 PID Control Program Examples 1618
Program examples for PID control in automatic mode 1618
Program examples for PID control when switching modes 1625
Appendix 6 Replacement of Other Format Projects 1632
Replacement of a GX Works2 format project 1632
Replacement of a PX Developer format project 1649

INDEX 1651

INSTRUCTION INDEX 1653

REVISIONS 1665
WARRANTY 1667
TRADEMARKS 1668

RELEVANT MANUALS

Manual name [manual number]	Description
MELSEC iQ-R Programming Manual (CPU Module Instructions, Standard Functions/Function Blocks) [SH-081266ENG] (this manual)	Instructions for the CPU module and standard functions/function blocks
MELSEC iQ-R Programming Manual (Module Dedicated Instructions) [SH-081976ENG]	Dedicated instructions for the intelligent function modules
MELSEC iQ-R Programming Manual (Process Control Function Blocks/ Instructions) [SH-081749ENG]	General process FBs, tag access FBs, tag FBs, and process control instructions designed for process control
MELSEC iQ-R Programming Manual (Program Design) [SH-081265ENG]	Program specifications (ladder, ST, FBD/LD, and SFC programs)
GX Works3 Operating Manual [SH-081215ENG]	System configuration, parameter settings, and online operations of GX Works3

TERMS

Unless otherwise specified, this manual uses the following terms.

Term	Description
Backup mode	An operation mode of the redundant system. This mode can continue the operation by switching the systems from the control system to the standby system when an error occurs in the control system.
Buffer memory	Memory in an intelligent function module for storing data such as setting values and monitored values. Buffer memory in a CPU module stores setting values and monitored values of the Ethernet function and data used for data communications among the CPU modules in a multiple CPU system.
Control CPU	A CPU module that controls connected I/O modules and intelligent function modules. In a multiple CPU system, there are multiple CPU modules and each connected module can be controlled by a different CPU module.
Control system	A system that takes control and performs network communications in a redundant system
Engineering tool	A tool used for setting up programmable controllers, programming, debugging, and maintenance
Intelligent function module	A module that has functions other than input and output, such as an analog module
Label	A label that represents a device in a given character string
Local station	A station that performs cyclic transmission and transient transmission with the master station and other local stations.
Master station	A station that controls the entire network. This station can perform cyclic transmission and transient transmission with all stations.
Predefined protocol support function	A function of GX Works3. This function sets protocols appropriate to each external device and reads/writes protocol setting data.
Process CPU (process mode)	A Process CPU operating in process mode. Process control function blocks and the online module change function can be used.
Process CPU (redundant mode)	A Process CPU operating in redundant mode. A redundant system is configured with this CPU module. Process control function blocks and the online module change function can be used even in this mode.
Redundant system	A system consisting of two systems that have same configuration (CPU module, power supply module, network module, and other modules). Even after an error occurs in one of the two system, the other system takes over the control of the entire system.
Request message	A processing request message sent from external devices to SLMP-compatible devices
Response message	A processing result message sent from SLMP-compatible devices in response to the request message
Safety CPU	A module that performs both standard control and safety control and is used with a safety function module. The Safety CPU models include the R08SFCPU, R16SFCPU, R32SFCPU, and R120SFCPU.
Safety function module	A module that performs safety control and must be used with a Safety CPU. This module can only be used with the Safety CPU. The safety function module model name is R6SFM.
Separate mode	A mode for system maintenance in a redundant system. This mode can maintain a redundant system without stopping control while the system is running.
SIL2 function module	A module that performs safety control and must be used with a SIL2 Process CPU. This module can only be used with the SIL2 Process CPU. The SIL2 function module model name is R6PSFM.
SIL2 Process CPU	A module that performs both standard control and safety control and is used with a SIL2 function module. This module is also used with a redundant function module and configures a redundant system. The SIL2 Process CPU models include the R08PSFCPU, R16PSFCPU, R32PSFCPU, and R120PSFCPU.
Standby system	A backup system in a redundant system
System A	A system that is set as system A to distinguish two systems, which are connected with two tracking cables. When the two systems start up at the same time, this system will be a control system. System switching does not affect the system A/B setting.
System B	A system that is set as system B to distinguish two systems, which are connected with two tracking cables. When the two systems start up at the same time, this system will be a standby system. System switching does not affect the system A/B setting.

The following terms are used to explain systems using the SIL2 Process CPU and the Safety CPU.

Term	Description
Safety communications	Communication service that performs send/receive processing in the safety layer of the safety communication protocol
Safety control	Machine control by safety programs and safety data communications. When an error occurs, the machine in operation is securely stopped.
Safety cycle processing	Processing of safety input/output and safety program
Safety device	A device that can be used in safety programs (📖 MELSEC iQ-R CPU Module User's Manual (Application))
Safety label	A generic term for the safety global label, safety local label, and standard/safety shared label (📖 MELSEC iQ-R CPU Module User's Manual (Application))
Safety program	A program for performing safety control

Term	Description
Standard communications	Communications other than safety communications, such as cyclic transmission and transient transmission of CC-Link IE Field Network
Standard device	A device (X, Y, M, D, or others) in a CPU module. (Safety devices are excluded.) This device can be used only in standard programs. (This term is used to distinguish from a safety device.)
Standard program	A program that performs sequence control. (Safety programs are excluded.) (This term is used to distinguish from a safety program.)

GENERIC TERMS AND ABBREVIATIONS

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

Generic term and abbreviation	Description
A/D converter module	A MELSEC iQ-R series analog-digital converter module, channel isolated analog-digital converter module, and high speed analog-digital converter module
Analog module	An A/D converter module, a D/A converter module, and a temperature input module
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 • RnENCPU
CC-Link IE Field Network-equipped master/local module	An RJ71GF11-T2 CC-Link IE Field Network master/local module and the following modules when the CC-Link IE Field Network function is used: • RJ71EN71 • RnENCPU
CC-Link IE TSN master/local module	RJ71GN11-T2, RJ71GN11-EIP (CC-Link IE TSN part)
D/A converter module	A MELSEC iQ-R series digital-analog converter module, channel isolated digital-analog converter module, and high speed digital-analog converter module
Ethernet interface module with built-in CC-Link IE	RJ71EN71
External device	A personal computer and other Ethernet-equipped modules connected over Ethernet for data communications
External device	A device that sends SLMP request messages to an SLMP-compatible device (personal computers, HMI (Human Machine Interface) and others)
FBD/LD	Function block diagram/ladder diagram
I/O module	An input module, an output module, an I/O combined module, and an interrupt module
MELSECNET/10	A MELSECNET/10 network system
MELSECNET/H	A MELSECNET/H network system
Network module	Includes the following: • Ethernet interface module • CC-Link IE TSN master/local module • CC-Link IE Controller Network module • CC-Link IE Field Network master/local module • MELSECNET/H module • MELSECNET/10 module • RnENCPU (network part)
Operand	A device, such as source data (s), destination data (d), number of devices (n), and others, used as parts to configure instructions and functions
Positioning module	A MELSEC iQ-R series positioning module
Process CPU	R08PCPU, R16PCPU, R32PCPU, R120PCPU
Programmable controller CPU	R00CPU, R01CPU, R02CPU, R04CPU, R04ENCPU, R08CPU, R08ENCPU, R16CPU, R16ENCPU, R32CPU, R32ENCPU, R120CPU, R120ENCPU
Remote head module	An RJ72GF15-T2 CC-Link IE Field Network remote head module
RJ71GN11-EIP (CC-Link IE TSN part)	An RJ71GN11-EIP when it performs communications on CC-Link IE TSN
RJ71GN11-EIP (EtherNet/IP part)	An RJ71GN11-EIP when it performs communications on EtherNet/IP
RnCPU	R00CPU, R01CPU, R02CPU, R04CPU, R08CPU, R16CPU, R32CPU, R120CPU
RnENCPU	R04ENCPU, R08ENCPU, R16ENCPU, R32ENCPU, R120ENCPU
RnENCPU (network part)	The right side (network part) of the RnENCPU (📖 MELSEC iQ-R Ethernet/CC-Link IE User's Manual (Startup))
RnPCPU	R08PCPU, R16PCPU, R32PCPU, R120PCPU
SFC	Sequential function chart
SLMP	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.
SLMP-compatible device	A device of the Mitsubishi Electric product that can transfer SLMP (Ethernet adapter module and Ethernet-equipped module)
ST language	Structured text language
Temperature input module	A MELSEC iQ-R series channel isolated thermocouple input module and channel isolated RTD input module

The following terms are used to explain systems using the SIL2 Process CPU and the Safety CPU.

Term	Description
Safety label	A safety global label, a safety local label, and a standard/safety shared label (📖 MELSEC iQ-R CPU Module User's Manual (Application))
Standard CPU	A MELSEC iQ-R series CPU module that performs standard control (This term is used to distinguish from the CPU modules that perform safety control.)

Instruction symbols

Unless otherwise specified, this manual uses the following generic symbols for some instructions.

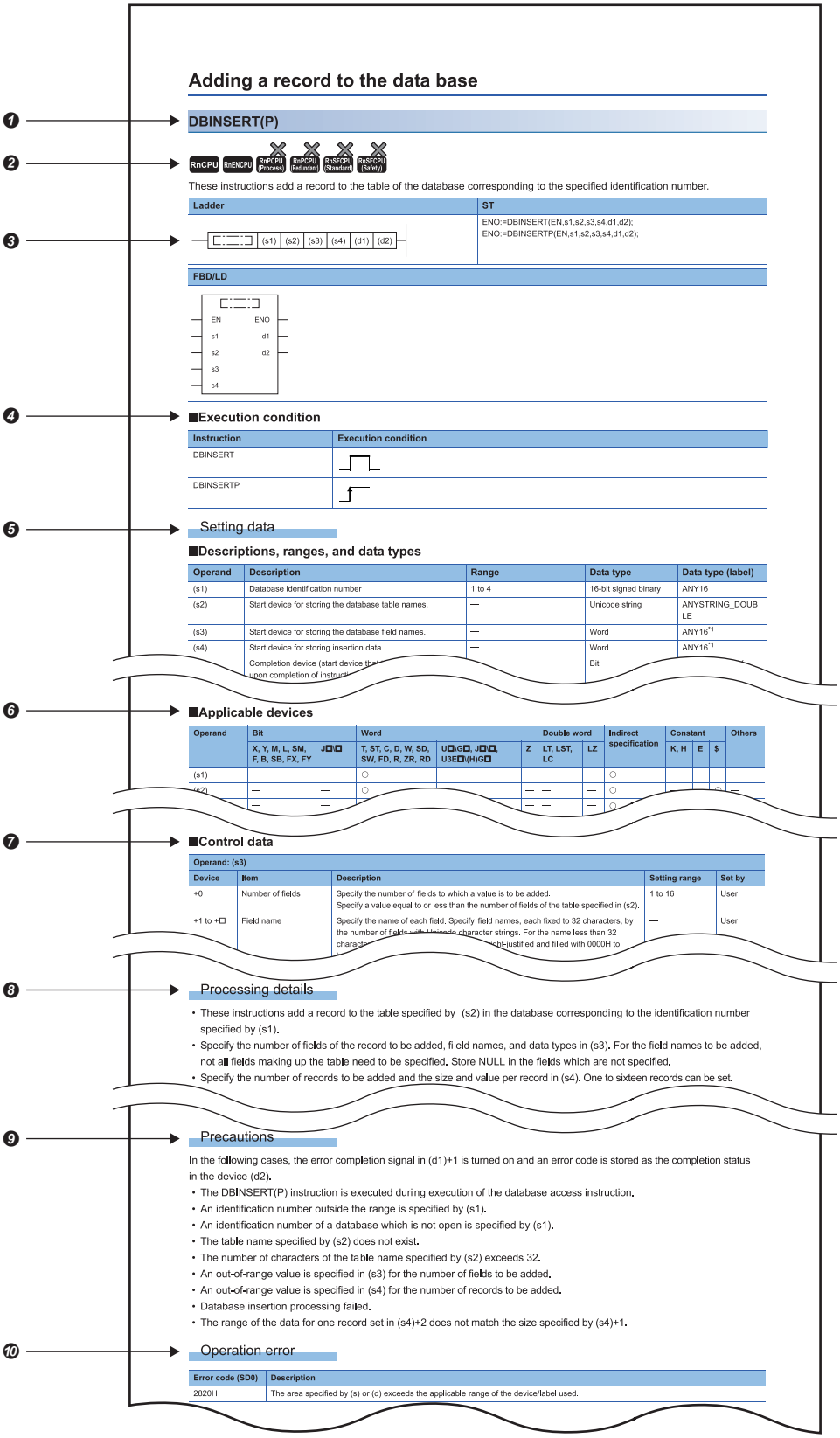
Classification	Instruction symbol	Generic symbol
Multiple CPU dedicated instruction	D(P).DDRD, M(P).DDRD	DDRD
	D(P).DDWR, M(P).DDWR	DDWR
PID control instruction	S(P).PIDINIT, PIDINIT(P)	PIDINIT
	S(P).PIDCONT, PIDCONT(P)	PIDCONT
	S(P).PIDPRMW, PIDPRMW(P)	PIDPRMW

MANUAL PAGE ORGANIZATION

In this manual, pages are organized and the symbols are used as shown below.

How to read Part 3 to Part 5

The following illustration is for explanation purpose only, and should not be referred to as an actual documentation.



❶ Instruction symbol

- An instruction symbol followed by parentheses indicates multiple instructions. For example, "GRY(P)(_U)" indicates four instructions: GRY, GRYP, GRY_U, and GRYP_U.

Instruction symbol	Meaning
Instruction symbol followed by "(P)"	This instruction is executed only on the rising edge (off to on).
Instruction symbol followed by "(_U)"	This instruction handles 16-bit or 32-bit unsigned binary data.

- An instruction symbol followed by "□" indicates multiple instructions. For example, "LDDT□" indicates six instructions: LDDT=, LDDT<>, LDDT>, LDDT<=, LDDT<, and LDDT>=.

❷ Availability by the CPU module type (The instruction cannot be used by the CPU module marked ×.)

❸ Description formats of ladder diagram, structured text language, and FBD/LD

An instruction symbol should be described in the enclosed area of each ladder or FBD/LD program.

Execution condition is input to EN of each structured text or FBD/LD program. And, execution result should be described for ENO.

❹ Execution condition (☞ Page 51 Execution Condition)

❺ Description of operands, setting ranges, data types, and label data types

- For the data type, refer to the following.

☞ Page 34 Data Specification Method

❻ Devices that can be used as operands

Operand	Bit		Word			Double word		Indirect specification	Constant			Others*5
	X, Y, M, L, SM, F, B, SB, S, FX, FY	J□\□*4	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□*4, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
Applicable device*1	X, Y, M, L, SM, F, B, SB, S, FX*2, FY*2	J□\X J□\Y J□\B J□\SB	T*3, ST*3, C*3, D, W, SD, SW, FD*2, R, ZR, RD	U□\G□ U3E□\G□ U3E□\HG□ J□\W J□\SW	Z	LT*3 LST*3 LC*3	LZ	@□ @□.□	K, H	E	$	P, I, J, U, DX, DY, N, V, BL, BL□\S□

The following table lists safety devices that can be used as operands in safety programs executed by the SIL2 Process CPU and the Safety CPU.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
Applicable device*1	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T*3, SA\ST*3, SA\C*3, SA\D, SA\W, SA\SD	K, H

*1 For details on each device, refer to the following.
📖 MELSEC iQ-R CPU Module User's Manual (Application)

*2 FX and FY can be used for bit data only, and FD for word data only.

*3 When T, ST, C, LT, LST, or LC is used for instructions other than those listed below, it can only be used as word data. It cannot be used as bit data.
[Instructions that can be used as bit data]
LD, LDI, AND, ANI, OR, ORI, LDP, LDF, ANDP, ANDF, ORP, ORF, LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI, OUT, RST, BKRST, MOVB(P), CMLB(P)
When SA\T, SA\ST, or SA\C is used for instructions other than those listed below, it can only be used as word data. It cannot be used as bit data.
[Instructions that can be used as bit data]
LD, LDI, AND, ANI, OR, ORI, LDP, LDF, ANDP, ANDF, ORP, ORF, LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI, OUT, RST, MOVB(P)

*4 This device can be used with a network module with a network number specified.

*5 In the "Others" column, a device(s) that can be set for each instruction is shown.

❼ Control data. Some instructions require control data that determine the operations of the instructions. When control data need to be set by a user, set values according the setting range.

❽ Processing details of the instruction. Unless otherwise specified, the following programs are regarded as interrupt programs.

- Interrupt program using the interrupt pointer (I)
- Fixed scan execution type program
- Event execution type program that is triggered by the interrupt pointer (I)

❾ Precautions

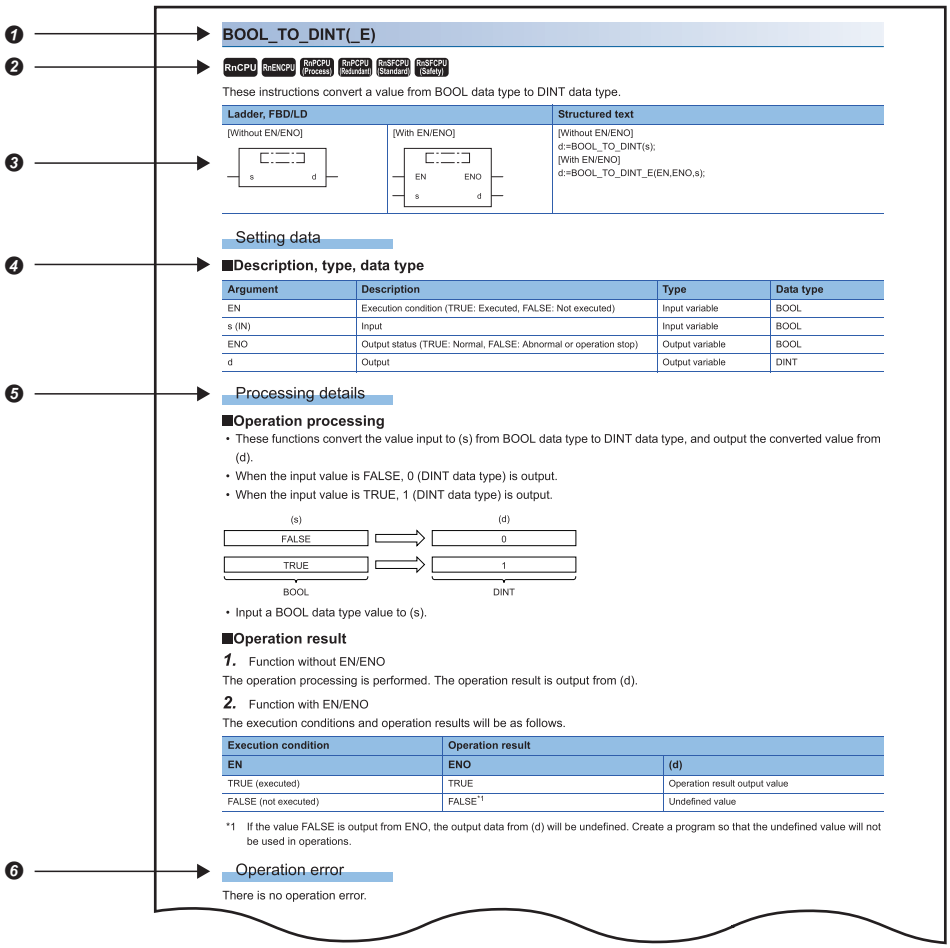
❿ Error code and error details if the instruction has any possible operation error

- A device in which an error code is stored is provided in the error code column. When an error code is stored in SD0, an error flag (SM0) turns on. (The error status can be checked with the module label of the CPU module.)
- For the errors not provided here, refer to the following.

📖 MELSEC iQ-R CPU Module User's Manual (Application)

How to read Part 7 and Part 8

The following illustration is for explanation purpose only, and should not be referred to as an actual documentation.



❶ Function symbol

A function symbol followed by parentheses indicates multiple functions or function blocks. For example, "BOOL_TO_DINT(_E)" includes two functions: "BOOL_TO_DINT" and "BOOL_TO_DINT_E".

Function symbol	Meaning
Function symbol followed by "(_E)"	This standard function or standard function block can write program with EN/ENO.

❷ Availability by the CPU module type (The function or function block cannot be used by the CPU module marked ×.)

❸ Description formats of ladder diagram, structured text language, and FBD/LD

In the enclosed area, either of the following symbol should be described.

- Standard function: Function symbol
- Standard function block: Instance name and function block symbol

Execution condition is input to EN of each standard function or function block. And, execution result is output from ENO of each standard function or function block.

The return value of functions are not displayed in FBD/LD programs.

For instances, refer to the following.

📖 MELSEC iQ-R Programming Manual (Program Design)

❹ Description of operands, types, data types, and label data types

- For the data type, refer to the following.

☞ Page 34 Data Specification Method

❺ Processing details of the standard function or standard function block

❻ Error code and error details if the standard function or standard function block has any possible operation error

A device in which an error code is stored is provided in the error code column. When an error code is stored in SD0, an error flag (SM0) turns on. (The error status can be checked with the module label of the CPU module.)

For the errors not provided here, refer to the following.

📖 MELSEC iQ-R CPU Module User's Manual (Application)

PART 1 OVERVIEW

This part consists of the following chapter.

1 OVERVIEW

1 OVERVIEW

1.1 Instruction Configuration

Many instructions available for programmable 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.

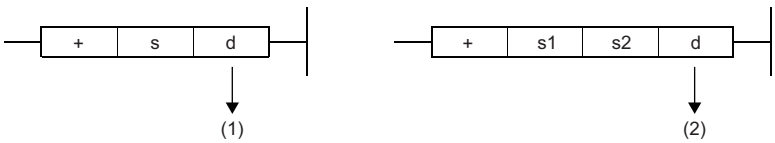
Type	Description
Constant	The constant specifies a numerical value used in the operation. It is set during program creation and cannot be changed during program execution. When using constants in variable data, perform index modification.*1
Device Label	The user specifies the device or label where the 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.
MELSEC iQ-R CPU Module User's Manual (Application)

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.

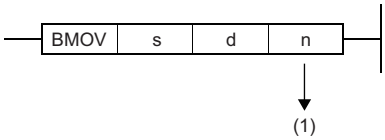
A label or device to store data must be set for the destination.

Numerical value (n)

For the numerical values of the numbers of devices, transfers, data, and character 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 character 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.
MELSEC iQ-R CPU Module User's Manual (Startup)

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 CPU modules.

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 38 Bit data	• Bit device • Bit specification of word device
Word	Word data can be handled. ☞ Page 40 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 unsigned binary	16-bit data can be handled. The value range varies depending on whether the value is signed or unsigned. ☞ Page 40 16-bit data (word data)	
Double word	Double-word data can be handled. ☞ Page 43 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 32-bit unsigned 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 43 32-bit data (double word data)	
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 46 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 47 Configuration of double-precision real number data	• Word device • Double-word device • Real constant
Character string	ASCII code and Shift JIS code character string data can be handled. ☞ Page 49 String data	• Word device • Character string constant
Unicode character string	Unicode character string data can be handled. ☞ Page 49 String data	• Word device • Character 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
Character string (STRING)	• Character string type label
Character string [Unicode] (WSTRING)	• Character string [Unicode] type label
Pointer (POINTER)	• Pointer type label

Point

For details on individual labels, refer to the following.

📖 MELSEC iQ-R CPU Module User's Manual (Application)

■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	Character 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	Character string array
		ANY_STRING_DOUBLE_ARRAY	Character 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, U3E□\HG) • File register (R, ZR) • Module refresh register (RD)

A bit number of a safety device used in safety programs executed by the SIL2 Process CPU and Safety CPU can be specified in hexadecimal within the range from 0 to F.
For example, bit 5 (b5) of SA\D0 is specified as SA\D0.5, and bit 10 (b10) of SA\D0 is specified as SA\D0.A.
The following word devices support bit specification.

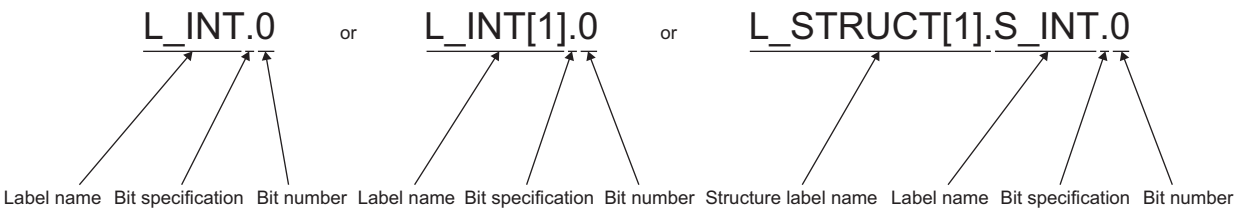
Item	Device
Word devices which support bit specification	• Safety data register (SA\D) • Safety link register (SA\W) • Safety special register (SA\SD)

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.)	K4X10 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\K2Y10

A bit device used in safety programs executed by the SIL2 Process CPU and Safety CPU can be handled as 16-bit data by performing digit specification.

Item	Notation	Example
Bit device	SA\K□Bit device start number □: Number of digits (Specify the number within the range of 1 to 4.)	SA\K4X10 SA\K2M113

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

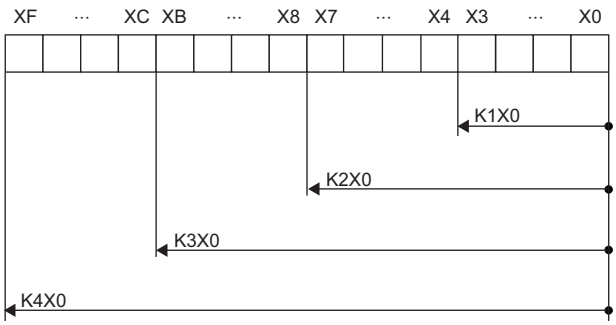
The following table lists the range of 16-bit data for each digit specification.

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.

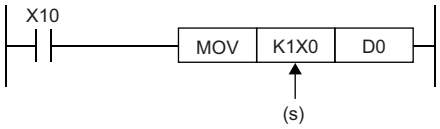
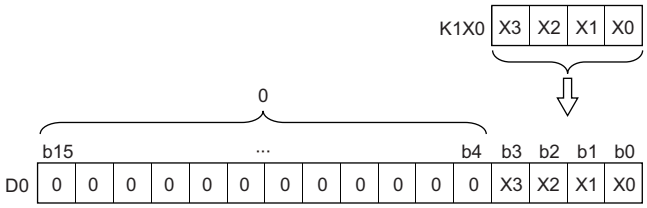
When digit specification is made for X0, the applicable number of points is as follows.

- K1X0→4 points from X0 to X3
- K2X0→8 points from X0 to X7
- K3X0→12 points from X0 to XB
- K4X0→16 points from X0 to XF



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

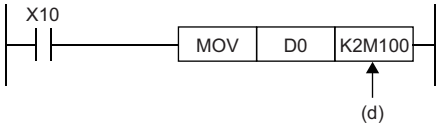
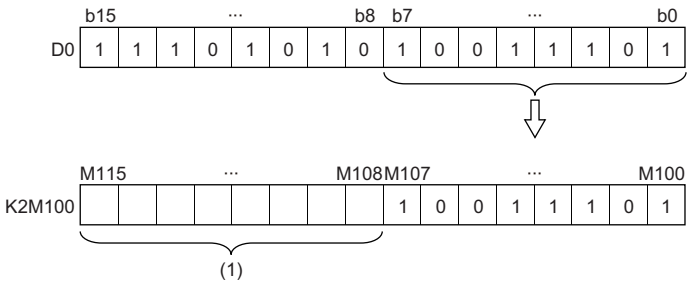
When a bit device is specified with digit specification in the source of an instruction, 0 is stored in the word device of the destination, in the upper bits than those specified in the source of the instruction.

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

A bit device used in safety programs executed by the SIL2 Process CPU and Safety CPU can be handled as 32-bit data by performing digit specification.

Item	Notation	Example
Bit device	SA\K□ Bit device start number Number of digits: Specify the number within the range of 1 to 8.	SA\K8X80 SA\K6B018

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

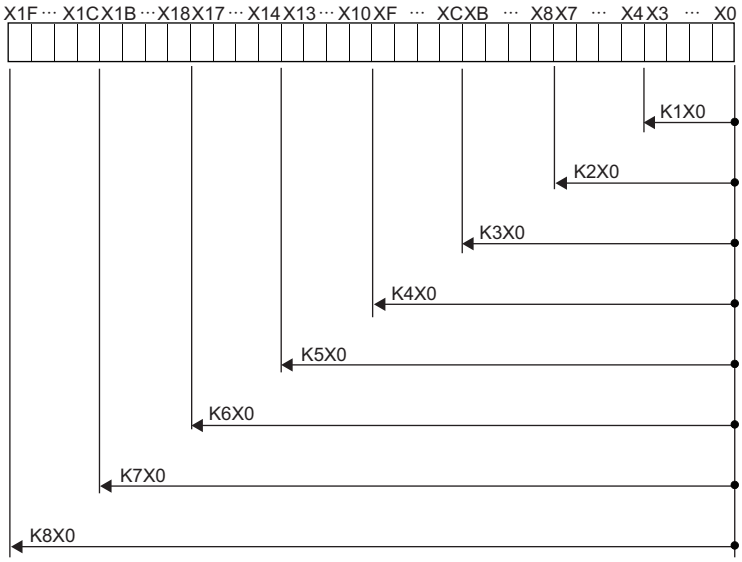
The following table lists the range of 32-bit data for each digit specification.

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.

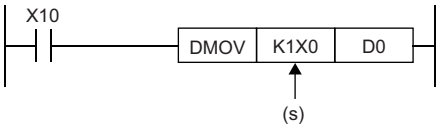
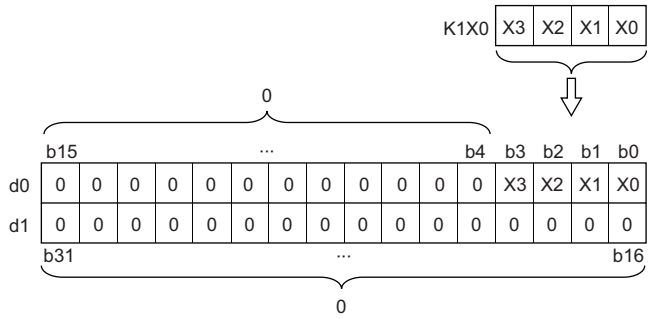
When digit specification is made for X0, the applicable number of points is as follows.

- K1X0→4 points from X0 to X3
- K2X0→8 points from X0 to X7
- K3X0→12 points from X0 to XB
- K4X0→16 points from X0 to XF
- K5X0→20 points from X0 to X13
- K6X0→24 points from X0 to X17
- K7X0→28 points from X0 to X1B
- K8X0→32 points from X0 to X1F



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

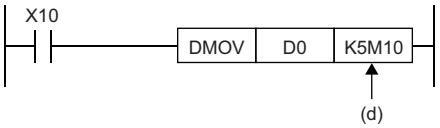
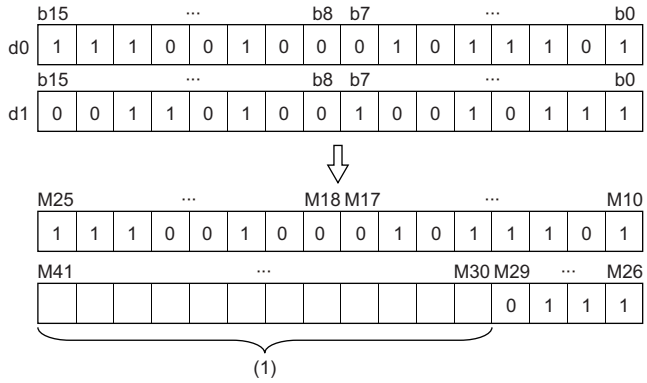
When a bit device is specified with digit specification in the source of an instruction, 0 is stored in the word device of the destination, in the upper bits than those specified in the source of the instruction.

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.

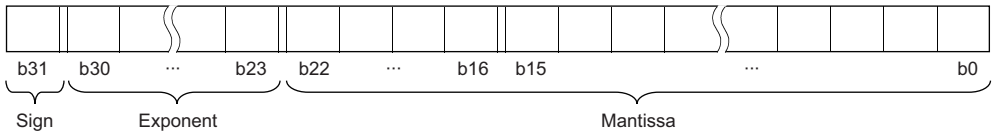
Real number data can be stored only in devices other than bit devices or in single-precision or double-precision real data type labels.

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···$\times 2^N$ representing a single-precision real number in binary.

■Exponent (8 bits)

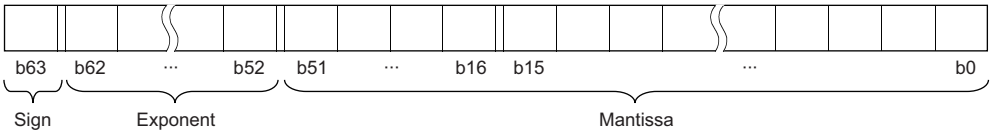
An exponent means N of 1.XXXXX···$\times 2^N$ representing a single-precision real number in binary. The following table shows the relationships between the exponent value and N of a single-precision real number.

Exponent (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···×2^N representing a single-precision real number in binary.

■Exponent (11 bits)

An exponent means N of 1.XXXXX···×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 ≤ 999999999999999000000.0

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 CPU modules.

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.
MELSEC iQ-R CPU Module User's Manual (Application)

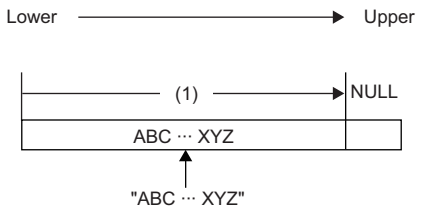
String data

Format of character string data

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

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

Character 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 character string

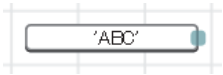
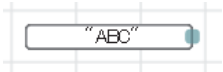
The following shows the notation of character strings in ladder programs.

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

The following shows the notation of character strings in ST programs.

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

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

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

Data range

The following table summarizes the ranges of character string data.

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

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

Number of words required for storing data

Character string data can be stored in word devices.

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

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

Character string data storage location

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

■Character strings

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

Character string to be stored	Image of storing character string data from D0	Image of storing character string data from word type label array arrayA[0]
Null character 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 character strings

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

Character string to be stored	Image of storing character string data from D0	Image of storing character string data from word type label array arrayA[0]
Null character 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.

1.4 High-speed Instruction Processing

Subset processing

Subset processing can reduce the number of steps or speed up the instruction processing when the device and label specified by each operand of an instruction satisfy the specified conditions.
Instruction symbols and the number of operands do not change whether subset processing is applicable or not.

Instructions that support subset processing

For the availability of subset processing for each instruction, refer to the following.
☞ Page 1586 Number of Basic Steps and Availability of Subset Processing

Operand condition

The conditions that the operands need to satisfy to enable subset processing are shown.

■When a device is specified in an operand

The following table lists the conditions that an operand which specifies a device needs to satisfy.

Data type of operand	Condition*1
Bit data	One of the following is satisfied. • User device • Host CPU specification of CPU buffer memory access device (excluding index modification to "U3En")*2 • Other CPU modules specification of fixed scan communication area of CPU buffer memory access device*3 • File register • Local device • Refresh data register
Signed 16-bit data Unsigned 16-bit data Signed 32-bit data Unsigned 32-bit data	One of the following is satisfied. • User device • Host CPU specification of CPU buffer memory access device (excluding index modification to "U3En")*2 • Other CPU modules specification of fixed scan communication area of CPU buffer memory access device*3 • Index register • File register • Local device • Refresh data register • Constant (decimal, hexadecimal)
Single-precision real number	One of the following is satisfied. • User device • Host CPU specification of CPU buffer memory access device (excluding index modification to "U3En")*2 • Other CPU modules specification of fixed scan communication area of CPU buffer memory access device*3 • Index register • File register • Local device • Refresh data register • Constant (single-precision real number)

*1 Including the cases where bit numbers, digits, indirect addresses, or index-modified devices are specified
*2 True when U3En\G□, U3En\G□Zn, U3En\HG□, or U3En\HG□Zn is used in the CPU buffer memory access device of the host CPU module.
*3 True when U3En\HG□ or U3En\HG□Zn is used in the CPU buffer memory access device of another CPU module.

■When the label assigned a device is specified in an operand

The same conditions as those applicable when a device is specified in an operand apply.

■When the label assigned to each label area is specified in an operand

When the label assigned to a label area or latch label area is specified in an operand, any instruction which supports subset processing performs subset processing regardless of the data type of the operand. (Including the cases where bit numbers or digits are specified.)

1.5 Precautions on Programming

Errors common to instructions

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

Error content*1	Error code
An I/O number which is out of range (other than 0H to FFH 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

Checking the ranges of devices and labels

When a device or label is specified in an instruction, no range check is performed, so a program needs to be created so that the operation result falls within the range of the relevant device or label.

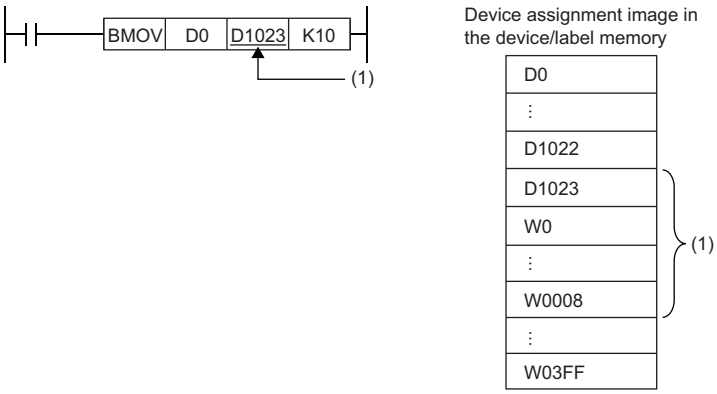
If a range exceeding that of the relevant device or label is specified, no error is detected but data is written to other device or label areas.

However, an error (error code: 2820H) occurs if data is written to outside the areas.

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

Ex.

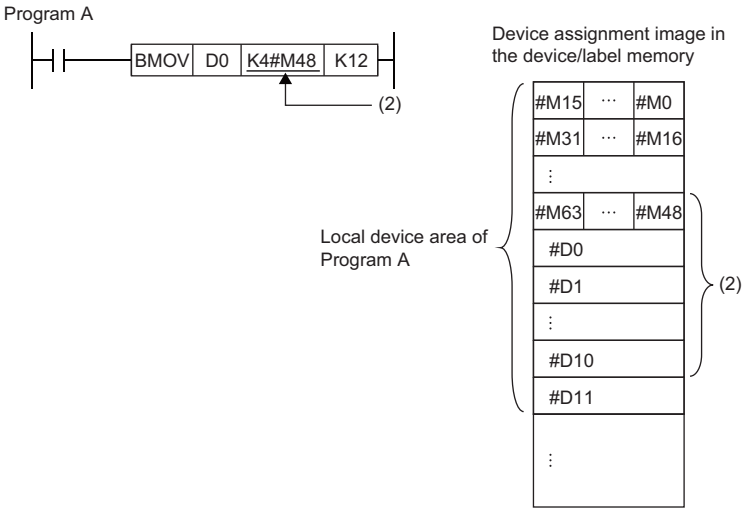
When W0 is assigned after global device D1023 in the device/label memory



(1) The transfer destination is in the range corresponding to D1023 to D1032. Even though the range D1024 to D1032 does not exist, the data are written and the data in W0 to W0008 are overwritten.

Ex.

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

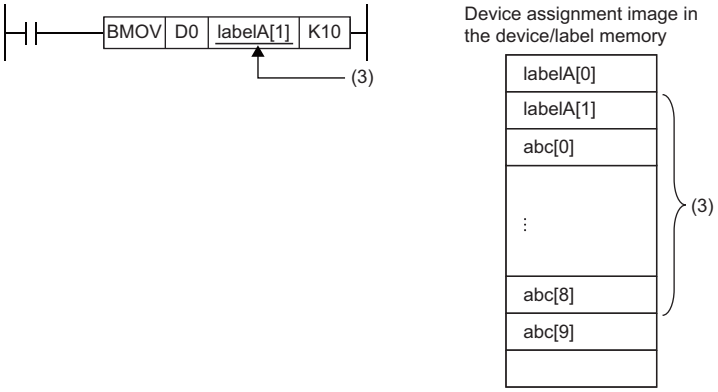


(2) The transfer destination is in the range corresponding to local devices #M48 to #M239 of program A. Even though the range #M64 to #M239 does not exist, the data are written and the data in #D0 to #D10 are overwritten.

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



(3) The transfer destination is in the range corresponding to 10 points from labelA[1]. Even though the range labelA[2] to labelA[9] does not exist, the data are written and the data in abc[0] to abc[8] are overwritten.

Checking the range of file register

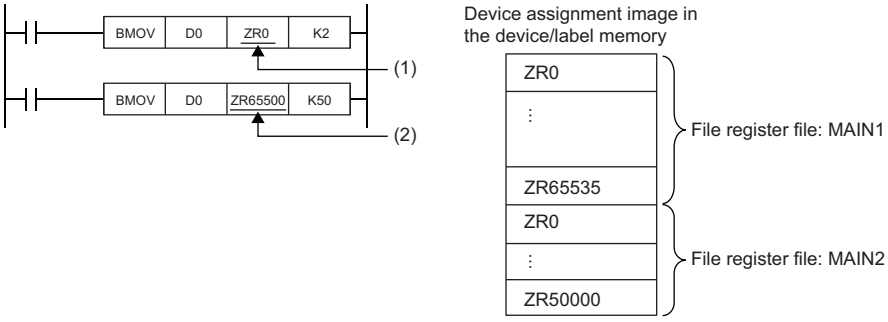
When a file register is specified in an instruction, a range check is performed, so a program needs to be created so that the operation result falls within the range of the relevant file register.

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

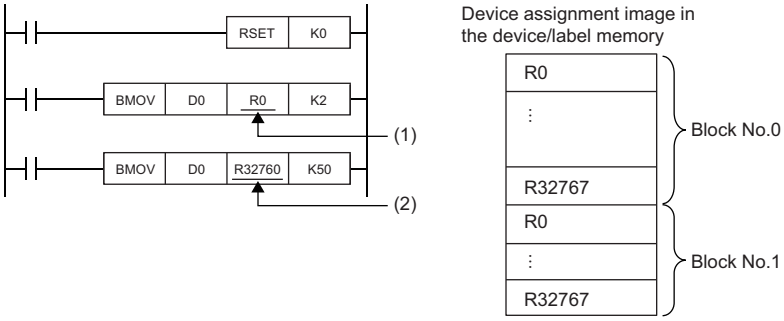


(1) The transfer destination is in the range of file register MAIN1. Data is written to ZR0 and ZR1.

(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

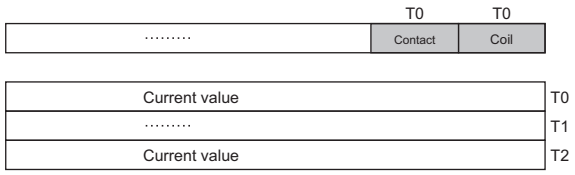
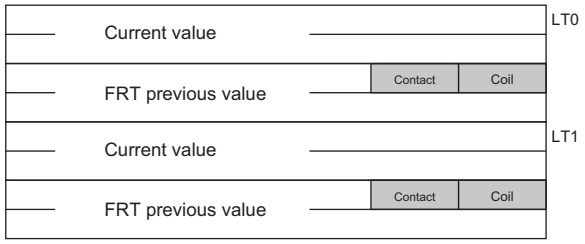
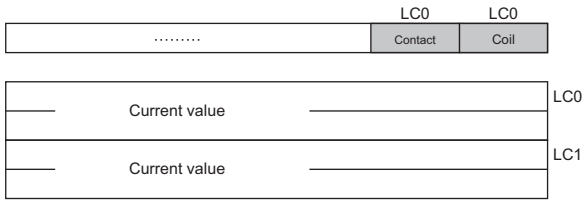


(1) The transfer destination is in the range of the R device of block number 0. Data is written to R0 and R1.

(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.

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.

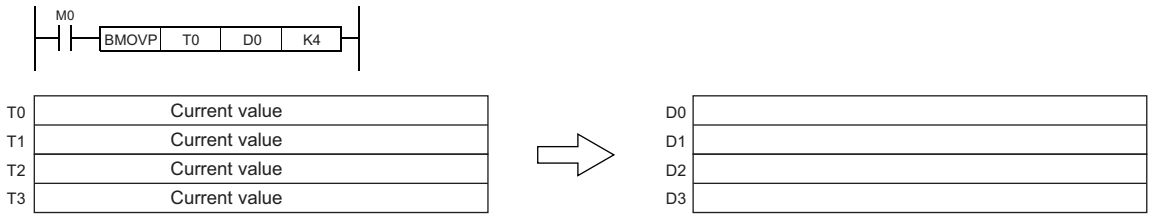
Device	Configuration
Timer (T) Retentive timer (ST) Counter (C)	
Long timer (LT) Long retentive timer (LST)	
Long counter (LC)	

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

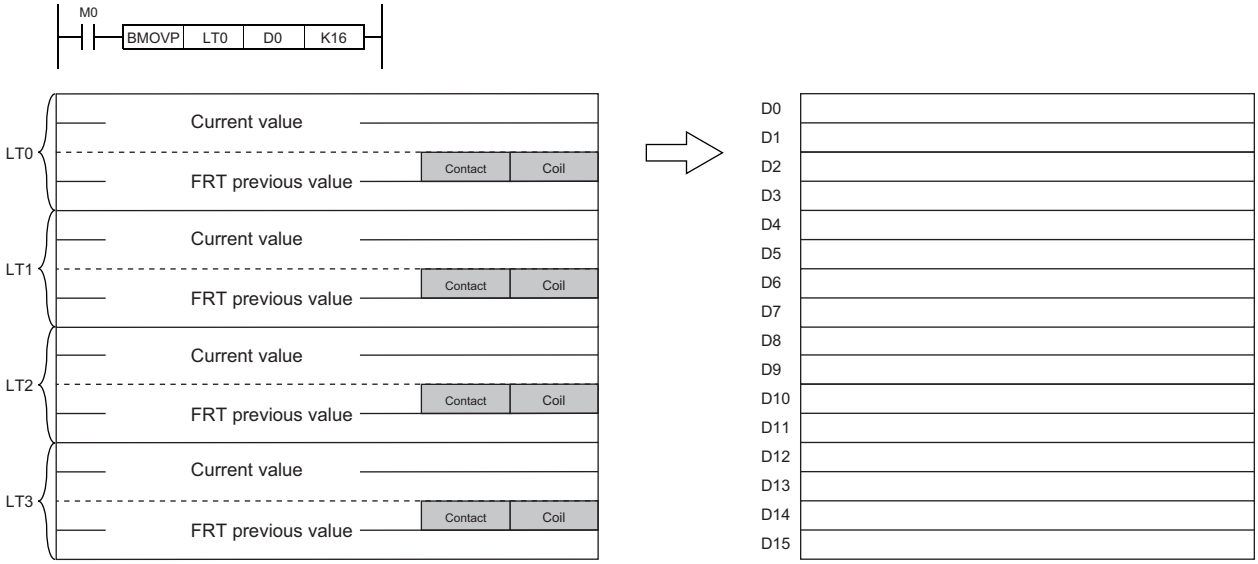
When the BMOV instruction is used, only current values are batch-transferred.



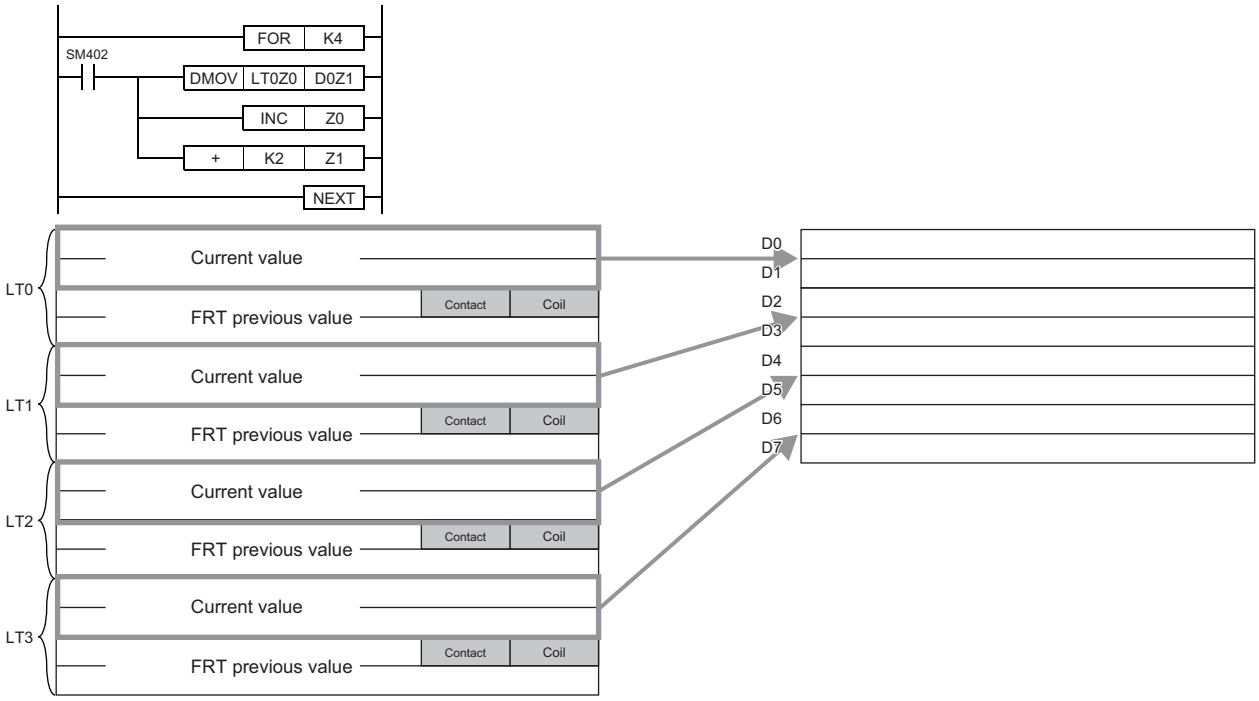
Ex.

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.



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.

Point

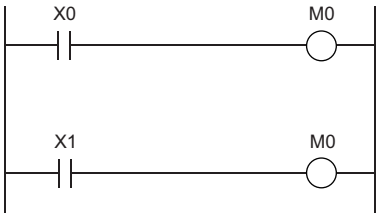
For the operation when two or more OUT, SET/RST, and PLS/PLF instructions that use the same device are executed within one safety cycle processing in safety programs executed by the SIL2 Process CPU and Safety CPU, replace some words as follows:

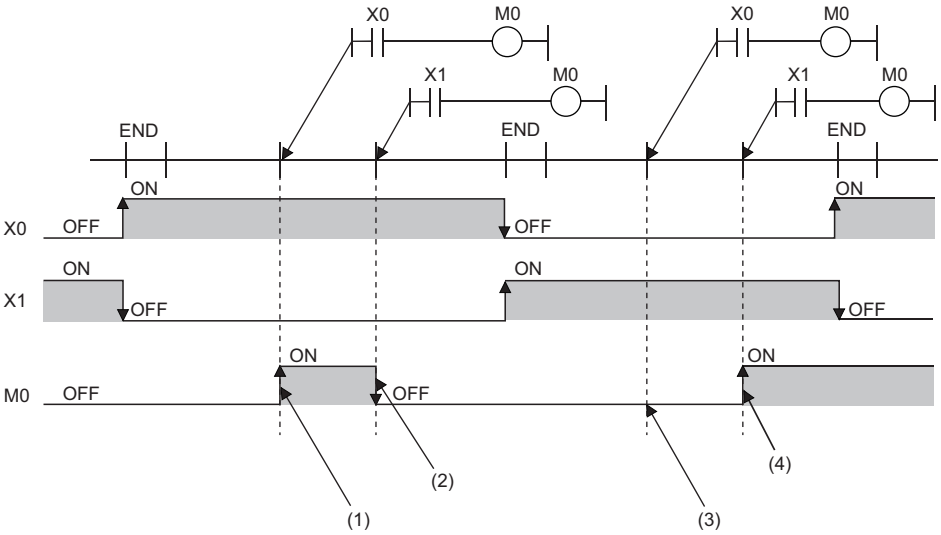
- "Scan" → "Safety cycle processing"
- "X0" → "SA\X0", "X1" → "SA\X1", "M0" → "SA\M0"
- "END" on the rising edge of X0 (in Figures) → "Safety cycle processing start"*1
- "END" on the falling edge of X0 (in Figures) → "Safety cycle processing end"*1

*1 For the PLF instruction, replace "END" with "safety cycle processing start" regardless of the X0 status (rising edge or falling edge).

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 X0 and X1.





(1) Since X0 is on, M0 turns on.
(2) Since X1 is off, M0 turns off.
(3) Since X0 is off, M0 remains off.
(4) Since X1 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

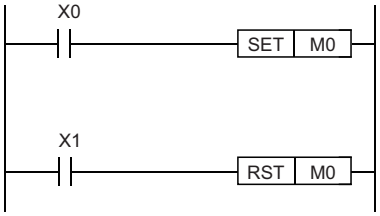
The SET instruction turns on the specified device if the execution command is on, and causes no operation if it is off.
Thus, if two or more SET instructions of the same device are executed during one scan, the specified device turns on even if one execution command is on.

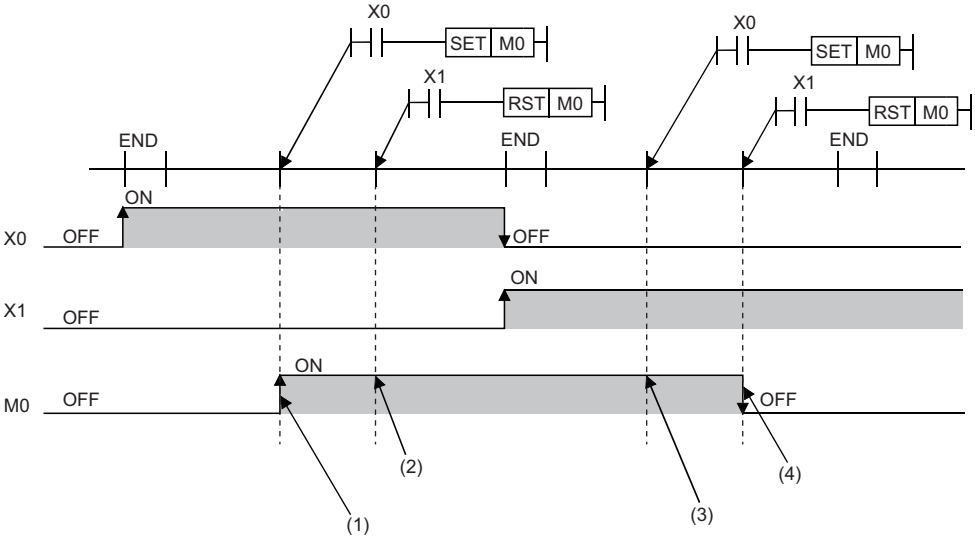
■For RST instructions

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.



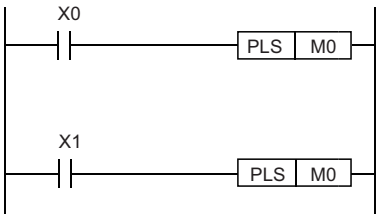


(1) Since X0 is on, M0 turns on.
(2) Since X1 is off, M0 remains on. (The RST instruction results in non-processing.)
(3) Since X0 is off, M0 remains on. (The SET instruction results in non-processing.)
(4) Since X1 is on, M0 turns off.

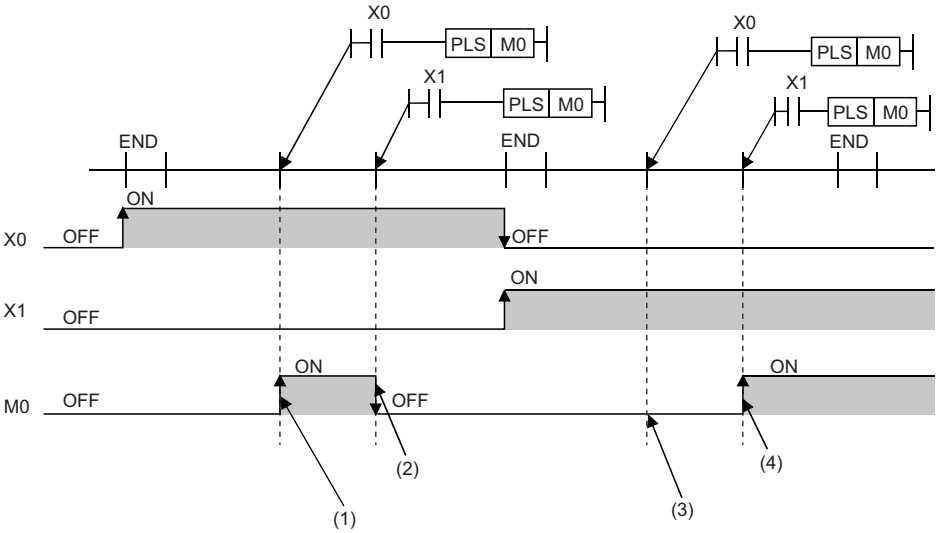
If output (Y) is specified using a SET/RST instruction, the on/off state of the last SET/RST instruction executed during the one scan will be output.

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.

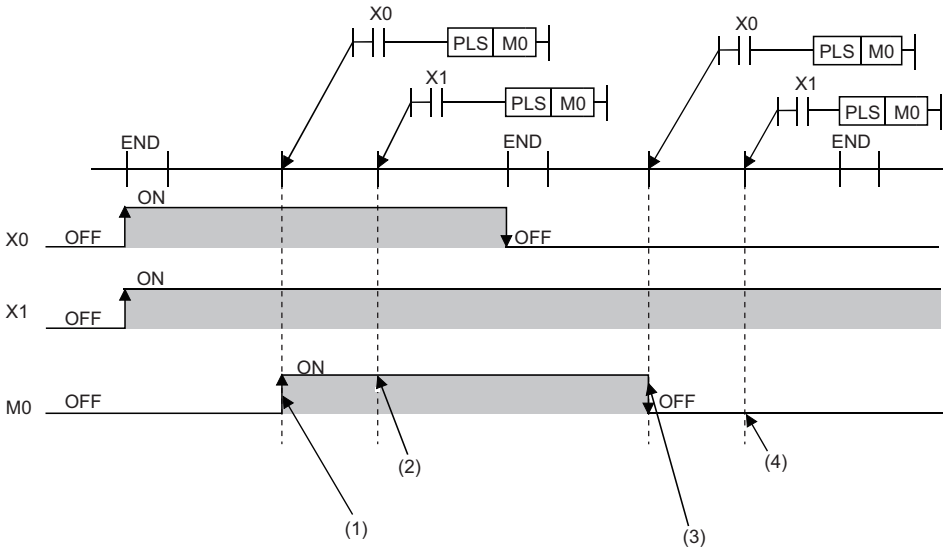


- If X0 and X1 differs in the on/off timing (i.e. the specified device does not turn on for one scan)



(1) Since X0 turns on, M0 turns on.
(2) Since X1 is other than turning on, M0 turns off.
(3) Since X0 is other than turning on, M0 remains off.
(4) Since X1 turns on, M0 turns on.

- If the off-to-on changes of X0 and X1 are at the same timing

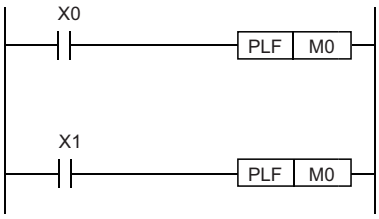


(1) Since X0 turns on, M0 turns on.
(2) Since X1 turns on, M0 remains off.
(3) Since X0 is other than turning on, M0 turns off.
(4) Since X1 is other than turning on, M0 remains off.

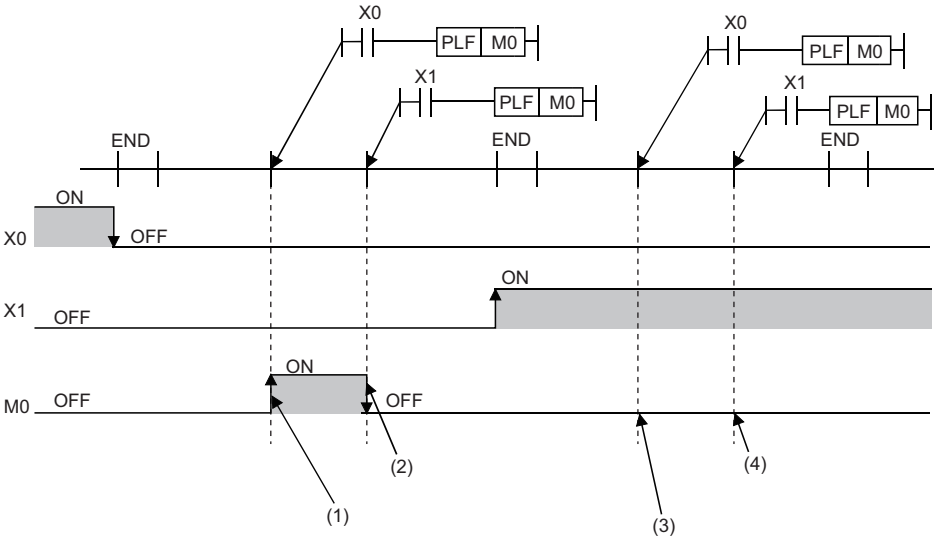
If output (Y) is specified using a PLS instruction, the on/off state of the last PLS instruction executed during the one scan will be output.

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.

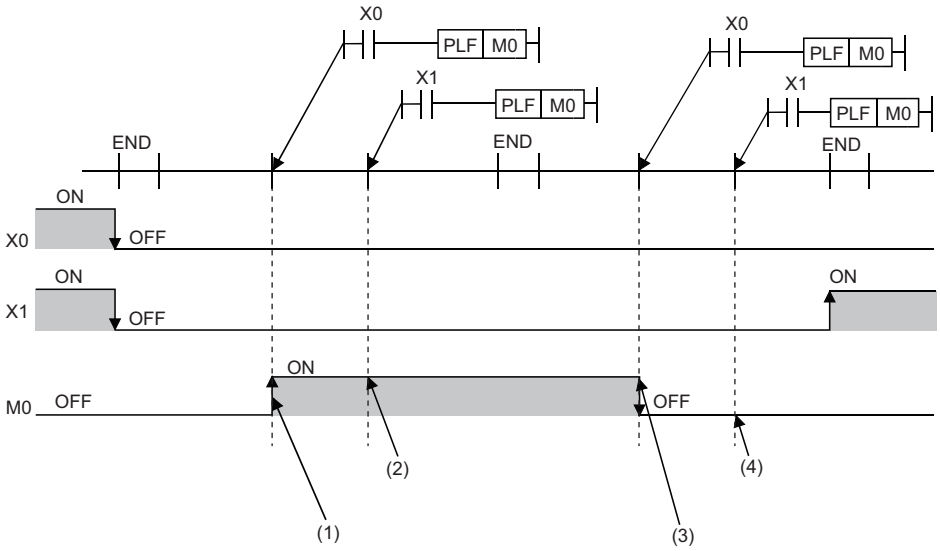


- If X0 and X1 differs in the on/off timing (i.e. the specified device does not turn on for one scan)



(1) Since X0 turns off, M0 turns on.
(2) Since X1 is other than turning off, M0 turns off.
(3) Since X0 is other than turning off, M0 remains off.
(4) Since X1 is other than turning off, M0 remains off.

- If the on-to-off changes of X0 and X1 are at the same timing



(1) Since X0 turns off, M0 turns on.
(2) Since X1 turns off, M0 remains on.
(3) Since X0 is other than turning off, M0 turns off.
(4) Since X1 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 a file register having the same name as a program is specified

If the use of a file register having the same name as a program is specified in the parameter, refresh cannot be performed correctly. When a file register having the same name of a program is used, data is refreshed by the file register having the same name of the program that has been set at the final number in the program settings.
To read or write refresh data, use the QDRSET instruction to switch to the corresponding file register and specify it.

If the file name or drive number is changed by the QDRSET instruction

If the file register file name or drive number is changed by the QDRSET instruction, the setting file is linked immediately before refresh.
To read or write refresh data, specify it in the setting file immediately before refresh.

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.

PART 2 LISTS OF INSTRUCTIONS AND FUN/FB

This part consists of the following chapters.

2 CPU MODULE INSTRUCTIONS

3 MODULE DEDICATED INSTRUCTIONS

4 STANDARD FUNCTIONS/FUNCTION BLOCKS

2 CPU MODULE INSTRUCTIONS

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

2.1 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 150 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 153 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 156 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 159 ANB, ORB
ORB	Performs OR operations between logical blocks (series connection between logical blocks)	

■Storing/reading/clearing the operation result

Instruction symbol	Processing details	Reference
MPS	Stores the operation result (on/off) immediately before the MPS instruction.	Page 160 MPS, MRD, MPP
MRD	Reads the operation result stored by using the MPS instruction.	
MPP	Clears the operation result stored by using the MPS instruction.	

■Inverting the operation result

Instruction symbol	Processing details	Reference
INV	Inverts the operation result up to just before the INV instruction.	Page 162 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 163 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 164 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 166 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 168 OUT T, OUTH T, OUT ST, OUTH ST
OUTH T		
OUT ST		
OUTH ST		
OUT LT		Page 171 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 174 OUT C
OUT LC		Page 176 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 178 OUT F

■Setting devices (excluding annunciator)

Instruction symbol	Processing details	Reference
SET	Turns on the specified bit.	Page 179 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 181 RST

■Setting/resetting annunciator

Instruction symbol	Processing details	Reference
SET F	Turns on the specified annunciator.	Page 183 SET F
RST F	Turns off the specified annunciator.	Page 185 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 187 PLS
PLF	Turns on the specified device for one scan on the falling edge (on to off) of the execution command.	Page 189 PLF

■Inverting the bit device output

Instruction symbol	Processing details	Reference
FF	Inverts the status of the specified device.	Page 191 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 193 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 195 SFT(P)
SFTP		

Master control instructions

■Setting/resetting a master control

Instruction symbol	Processing details	Reference
MC	Starts a master control.	Page 197 MC, MCR
MCR	Ends a master control.	

Phase processing instructions

■Phase processing

Instruction symbol	Processing details	Reference
PHASE	Starts phase processing.	Page 202 PHASE
PHASECHG	Ends the phase currently being executed and shifts to the specified phase.	Page 204 PHASECHG
PHASEEND	Ends the phase currently being executed.	Page 206 PHASEEND

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 207 FEND

■Ending the sequence program

Instruction symbol	Processing details	Reference
END	Indicates the end of a program.	Page 208 END

Stop instruction

■Stopping the sequence program

Instruction symbol	Processing details	Reference
STOP	Stops the operation of the CPU module. (The operation of this instruction is the same as setting the switch of the CPU module to the STOP position.)	Page 210 STOP

No operation instruction

■No operation

Instruction symbol	Processing details	Reference
NOP	Inserts a space for debugging.	Page 211 NOP
NOPLF	This instruction is a no-operation instruction and has no impact on the previous operations.	Page 212 NOPLF

2.2 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 214 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 216 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		

■Outputting a comparison result of 16-bit binary data

Instruction symbol	Processing details	Reference
CMP	Compares the 16-bit binary data specified by (s1) with the 16-bit binary data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.	Page 218 CMP(P)(_U)
CMPP		
CMP_U		
CMPP_U		

■Outputting a comparison result of 32-bit binary data

Instruction symbol	Processing details	Reference
DCMP	Compares the 32-bit binary data specified by (s1) with the 32-bit binary data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.	Page 220 DCMP(P)(_U)
DCMPP		
DCMP_U		
DCMPP_U		

■Outputting a band comparison result of 16-bit binary data

Instruction symbol	Processing details	Reference
ZCP	Compares the band between the 16-bit binary data specified by lower limit value (s1) and the 16-bit binary data specified by upper limit value (s2) with the 16-bit binary data in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.	Page 222 ZCP(P)(_U)
ZCPP		
ZCP_U		
ZCPP_U		

■Outputting a band comparison result of 32-bit binary data

Instruction symbol	Processing details	Reference
DZCP	Compares the band between the 32-bit binary data specified by lower limit value (s1) and the 32-bit binary data specified by upper limit value (s2) with the 32-bit binary data in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.	Page 224 DZCP(P)(_U)
DZCPP		
DZCP_U		
DZCPP_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 226 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>=_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 228 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 231 +(P)(_U) [when two operands are set]
+P		
+_U		
+P_U		
+	Adds the two sets of 16-bit binary data specified. (Three operands)	Page 233 +(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 235 -(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 237 -(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 239 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 241 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 243 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 245 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 247 *(P)(_U)
*P		
*_U		
*P_U		
/	Performs division between the two sets of 16-bit binary data specified.	Page 249 /(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 251 D*(P)(_U)
D*P		
D*_U		
D*P_U		
D/	Performs division between the two sets of 32-bit binary data specified.	Page 253 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 255 B+(P) [when two operands are set]
B+P		
B+	Adds the two sets of BCD 4-digit data specified. (Three operands)	Page 256 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 258 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 259 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 261 DB+(P) [when two operands are set]
DB+P		
DB+	Adds the two sets of BCD 8-digit data specified. (Three operands)	Page 263 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 265 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 267 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 269 B*(P)
B*P		
B/	Performs division between the two sets of BCD 4-digit data specified.	Page 271 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 273 DB*(P)
DB*P		
DB/	Performs division between the two sets of BCD 8-digit data specified.	Page 275 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 277 BK+(P)(_U)
BK+P		
BK+_U		
BK+P_U		
BK-	Performs subtraction between the two 16-bit binary data blocks specified.	Page 279 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 281 DBK+(P)(_U)
DBK+P		
DBK+_U		
DBK+P_U		
DBK-	Performs subtraction between the two 32-bit binary data blocks specified.	Page 284 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 287 INC(P)(_U)
INCP		
INC_U		
INCP_U		
DEC	Decrements the specified 16-bit binary data by one.	Page 289 DEC(P)(_U)
DECP		
DEC_U		
DECP_U		

■Incrementing/decrementing 32-bit binary data

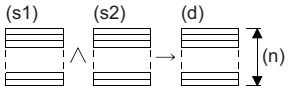
Instruction symbol	Processing details	Reference
DINC	Increments the specified 32-bit binary data by one.	Page 291 DINC(P)(_U)
DINCP		
DINC_U		
DINCP_U		
DDEC	Decrements the specified 32-bit binary data by one.	Page 293 DDEC(P)(_U)
DDECP		
DDEC_U		
DDECP_U		

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 295 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 297 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 299 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 301 DAND(P) [when three operands are set]

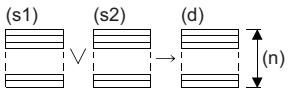
■Performing an AND operation on 16-bit block data

Instruction symbol	Processing details	Reference
BKAND BKANDP	Performs an AND operation on the two 16-bit binary data blocks specified. 	Page 303 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 305 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 307 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 309 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 311 DOR(P) [when three operands are set]

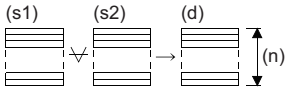
■Performing an OR operation on 16-bit block data

Instruction symbol	Processing details	Reference
BKOR BKORP	Performs an OR operation on the two 16-bit binary data blocks specified. 	Page 313 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 315 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 317 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 319 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 321 DXOR(P) [when three operands are set]

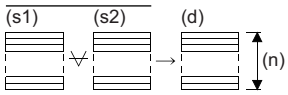
■Performing an XOR operation on 16-bit block data

Instruction symbol	Processing details	Reference
BKXOR	Performs an XOR operation on the two 16-bit binary data blocks specified.	Page 323 BKXOR(P)
BKXORP		

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

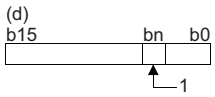
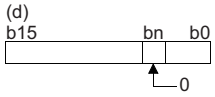
Instruction symbol	Processing details	Reference
WXNR	Performs an XNOR operation on the two sets of 16-bit binary data specified. (Two operands)	Page 325 WXNR(P) [when two operands are set]
WXNRP		
WXNR	Performs an XNOR operation on the two sets of 16-bit binary data specified. (Three operands)	Page 327 WXNR(P) [when three operands are set]
WXNRP		
DXNR	Performs an XNOR operation on the two sets of 32-bit binary data specified. (Two operands)	Page 329 DXNR(P) [when two operands are set]
DXNRP		
DXNR	Performs an XNOR operation on the two sets of 32-bit binary data specified. (Three operands)	Page 331 DXNR(P) [when three operands are set]
DXNRP		

■Performing an XNOR operation on 16-bit block data

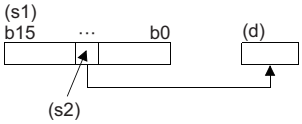
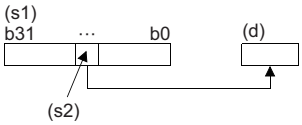
Instruction symbol	Processing details	Reference
BKXNR	Performs an XNOR operation on the two 16-bit binary data blocks specified.	Page 333 BKXNR(P)
BKXNRP		

Bit processing instructions

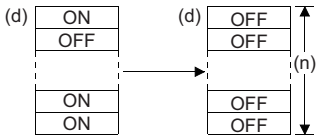
■Setting/resetting a bit in the word device

Instruction symbol	Processing details	Reference
BSET BSETP	Sets the 'n'th bit in the specified word device to 1. 	Page 335 BSET(P)
BRST BRSTP	Resets the 'n'th bit in the specified word device to 0. 	Page 337 BRST(P)

■Performing a bit test

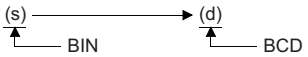
Instruction symbol	Processing details	Reference
TEST TESTP	Extracts the 'n'th bit in the specified word device. 	Page 339 TEST(P)
DTEST DTESTP	Extracts the 'n'th bit in the specified double-word device. 	Page 341 DTEST(P)

■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 343 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 409 BCD(P)
DBCD DBCDP	Converts the specified 32-bit binary data (0 to 99999999) to BCD 8-digit data. 	Page 411 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. (s) → (d) (s): BCD, (d): BIN	Page 413 BIN(P)
DBIN DBINP	Converts the specified BCD 8-digit data (0 to 99999999) to 32-bit binary data. (s)+1, (s) → (d)+1, (d) (s): BCD, (d): BIN	Page 415 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. (s)+1, (s) → (d) (s): (1), (d): BIN (1) Real number	Page 417 FLT2INT(P)
FLT2DINT FLT2DINTP	Converts the specified single-precision real number (-2147483648 to 2147483647) to 32-bit signed binary data. (s)+1, (s) → (d)+1, (d) (s): (1), (d): BIN (1) Real number	Page 421 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. (s)+1, (s) → (d) (s): (1), (d): BIN (1) Real number	Page 419 FLT2UINT(P)
FLT2UDINT FLT2UDINTP	Converts the specified single-precision real number (0 to 4294967295) to 32-bit unsigned binary data. (s)+1, (s) → (d)+1, (d) (s): (1), (d): BIN (1) Real number	Page 423 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. (s)+3, (s)+2, (s)+1, (s) → (d) (s): (1), (d): BIN (1) Real number	Page 425 DBL2INT(P)
DBL2DINT DBL2DINTP	Converts the specified double-precision real number (-2147483648 to 2147483647) to 32-bit signed binary data. (s)+3, (s)+2, (s)+1, (s) → (d)+1, (d) (s): (1), (d): BIN (1) Real number	Page 429 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. (s)+3, (s)+2, (s)+1, (s) → (d) (1) BIN (1) Real number	Page 427 DBL2UINT(P)
DBL2UDINT DBL2UDINTP	Converts the specified double-precision real number (0 to 4294967295) to 32-bit unsigned binary data. (s)+3, (s)+2, (s)+1, (s) → (d)+1, (d) (1) BIN (1) Real number	Page 431 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 433 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 437 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 435 INT2DINT(P)

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

Instruction symbol	Processing details	Reference
UINT2INT UINT2INTP	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 439 UINT2INT(P)
UINT2DINT UINT2DINTP	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 441 UINT2DINT(P)

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

Instruction symbol	Processing details	Reference
UINT2UDINT UINT2UDINTP	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 443 UINT2UDINT(P)

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

Instruction symbol	Processing details	Reference
DINT2INT DINT2INTP	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 445 DINT2INT(P)

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

Instruction symbol	Processing details	Reference
DINT2UINT DINT2UINTP	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 447 DINT2UINT(P)
DINT2UDINT DINT2UDINTP	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 449 DINT2UDINT(P)

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

Instruction symbol	Processing details	Reference
UDINT2INT UDINT2INTP	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 451 UDINT2INT(P)
UDINT2DINT UDINT2DINTP	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 455 UDINT2DINT(P)

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

Instruction symbol	Processing details	Reference
UDINT2UINT UDINT2UINTP	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 453 UDINT2UINT(P)

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

Instruction symbol	Processing details	Reference
GRY GRYP	Converts the specified 16-bit binary data (-32768 to 32767) to 16-bit binary Gray code data. (s) → (d) (s): BIN, (d): Gc Gc: Gray code	Page 457 GRY(P)(_U)
GRY_U GRYP_U	Converts the specified 16-bit binary data (0 to 65535) to 16-bit binary Gray code data. (s) → (d) (s): BIN, (d): Gc Gc: Gray code	
DGRY DGRYP	Converts the specified 32-bit binary data (-2147483648 to 2147483647) to 32-bit binary Gray code data. (s)+1, (s) → (d)+1, (d) (s): BIN, (d): Gc Gc: Gray code	Page 459 DGRY(P)(_U)
DGRY_U DGRYP_U	Converts the specified 32-bit binary data (0 to 4294967295) to 32-bit binary Gray code data. (s)+1, (s) → (d)+1, (d) (s): BIN, (d): Gc 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. (s) → (d) (s): Gc, (d): BIN Gc: Gray code	Page 461 GBIN(P)(_U)
GBIN_U GBINP_U	Converts the specified 16-bit binary Gray code data (0 to 65535) to 16-bit binary data. (s) → (d) (s): Gc, (d): BIN Gc: Gray code	
DGBIN DGBINP	Converts the specified 32-bit binary Gray code data (-2147483648 to 2147483647) to 32-bit binary data. (s)+1, (s) → (d)+1, (d) (s): Gc, (d): BIN Gc: Gray code	Page 463 DGBIN(P)(_U)
DGBIN_U DGBINP_U	Converts the specified 32-bit binary Gray code data (0 to 4294967295) to 32-bit binary data. (s)+1, (s) → (d)+1, (d) (s): Gc, (d): BIN 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 465 BKBCD(P)

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

Instruction symbol	Processing details	Reference
BKBIN	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 467 BKBIN(P)
BKBINP		

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

Instruction symbol	Processing details	Reference
DABIN	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 469 DABIN(P)(_U)
DABINP		
DABIN_U		
DABINP_U		
DDABIN	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 472 DDABIN(P)(_U)
DDABINP		
DDABIN_U		
DDABINP_U		

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

Instruction symbol	Processing details	Reference
HABIN	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 476 HABIN(P)
HABINP		
DHABIN	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 479 DHABIN(P)
DHABINP		

■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 482 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 485 DDABCD(P)
DDABCDP		

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

Instruction symbol	Processing details	Reference
VAL	Converts a character 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 488 VAL(P)(_U)
VALP		
VAL_U		
VALP_U		
DVAL	Converts a character 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 491 DVAL(P)(_U)
DVALP		
DVAL_U		
DVALP_U		

■Converting hexadecimal ASCII data 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 494 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 496 EMOD(P)
EMODP		

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

Instruction symbol	Processing details	Reference
NEG NEGP	Inverts the sign of 16-bit binary device. $\overline{(d)}$ → (d) BIN	Page 498 NEG(P)
DNEG DNEGP	Inverts the sign of 32-bit binary device. $\overline{(d)+1, (d)}$ → (d)+1, (d) BIN	Page 500 DNEG(P)

■Decoding 8-bit data to 256-bit data

Instruction symbol	Processing details	Reference
DECO DECOP	Decodes the lower (n) bits of the specified device. (s) → (d) (n) $2^{(n)}$ bits	Page 502 DECO(P)

■Encoding 256-bit data to 8-bit data

Instruction symbol	Processing details	Reference
ENCO ENCOP	Encodes the bit data of 'n'th power of 2. (s) $2^{(n)}$ bits → (d) (n)	Page 504 ENCO(P)

■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. (s) b3 ··· b0 → (d) 7SEG	Page 506 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 509 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 511 UNI(P)

■Separating/combining data in units of bits

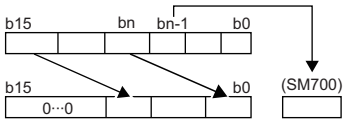
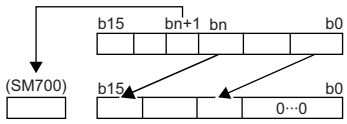
Instruction symbol	Processing details	Reference
NDIS NDISP	Separates the data in the device, specified by (s1) and later, to the bits in the device specified by (s2) and later, and stores the separated data in order in the device starting from the one specified by (d).	Page 513 NDIS(P)
NUNI NUNIP	Connects 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 515 NUNI(P)

■Separating/combining data in units of bytes

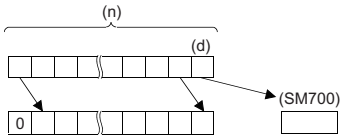
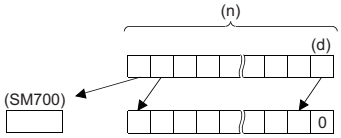
Instruction symbol	Processing details	Reference
WTOB WTOBP	Converts the (n) points of 16 bit data in the device specified by (s) in units of 8 bits, and stores the converted data in order in the device starting from the one specified by (d).	Page 517 WTOB(P)
BTOW BTOWP	Connects 8 low-order bits of the (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 519 BTOW(P)

Shift instructions

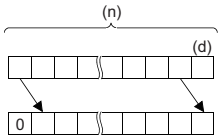
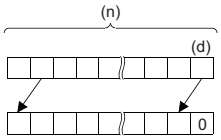
■Shifting 16-bit binary data to the right/left by n bit(s)

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 345 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 347 SFL(P)

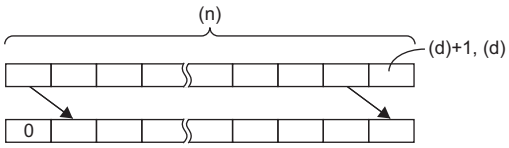
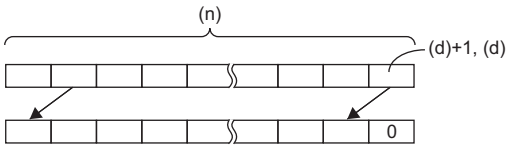
■Shifting n-bit data to the right/left by one bit

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 349 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 351 BSFL(P)

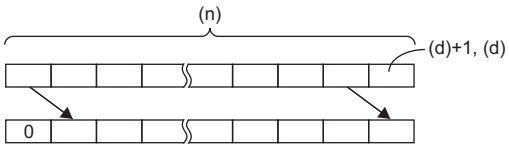
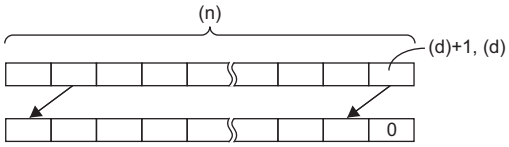
■Shifting n-word data to the right/left by one word

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 353 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 355 DSFL(P)

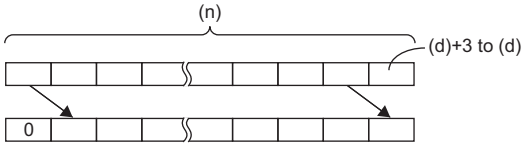
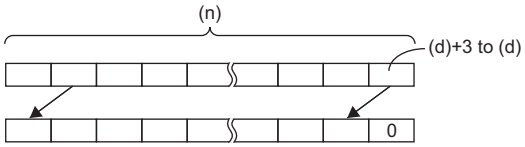
■Shifting n double word(s) of data to the right/left by one double word

Instruction symbol	Processing details	Reference
DDSFR DDSFRP	Shifts the n double word(s) of data starting from the specified device to the right by one double word. In the empty area after the shift, 0 is stored. 	Page 357 DDSFR(P)
DDSFL DDSFLP	Shifts the n double word(s) of data starting from the specified device to the left by one double word. In the empty area after the shift, 0 is stored. 	Page 359 DDSFL(P)

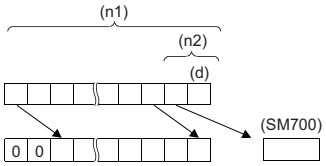
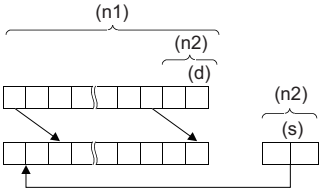
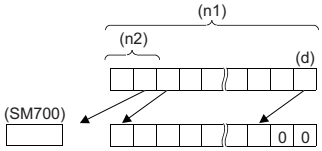
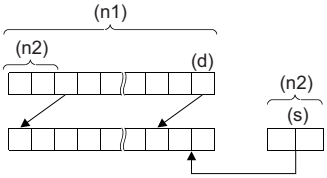
■Shifting n point(s) of single-precision real number data to the right/left by one point

Instruction symbol	Processing details	Reference
ESFR ESFRP	Shifts the n point(s) of single-precision real number data starting from the specified device to the right by one point. In the empty area after the shift, 0 is stored. 	Page 361 ESFR(P)
ESFL ESFLP	Shifts the n point(s) of single-precision real number data starting from the specified device to the left by one point. In the empty area after the shift, 0 is stored. 	Page 363 ESFL(P)

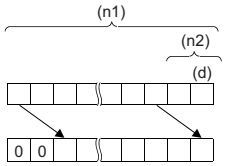
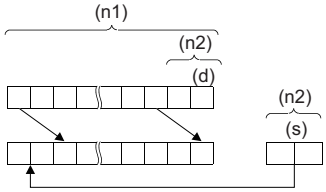
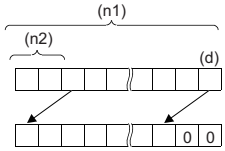
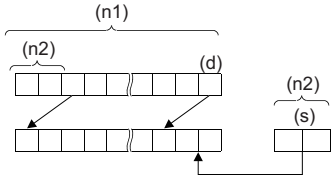
■Shifting n point(s) of double-precision real number data to the right/left by one point

Instruction symbol	Processing details	Reference
EDSFR EDSFRP	Shifts the n point(s) of double-precision real number data starting from the specified device to the right by one point. In the empty area after the shift, 0 is stored. 	Page 365 EDSFR(P)
EDSFL EDSFLP	Shifts the n point(s) of double-precision real number data starting from the specified device to the left by one point. In the empty area after the shift, 0 is stored. 	Page 367 EDSFL(P)

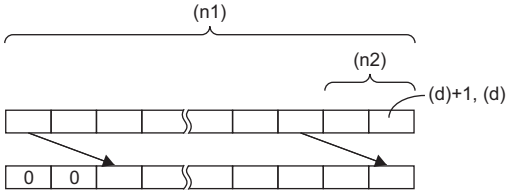
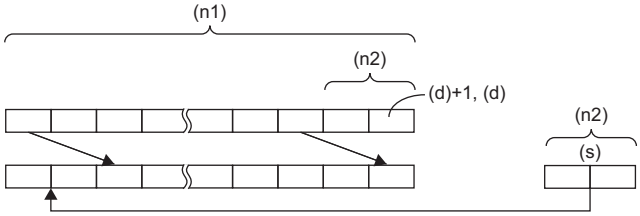
■Shifting n-bit data to the right/left by n bit(s)

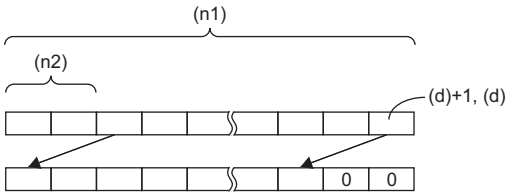
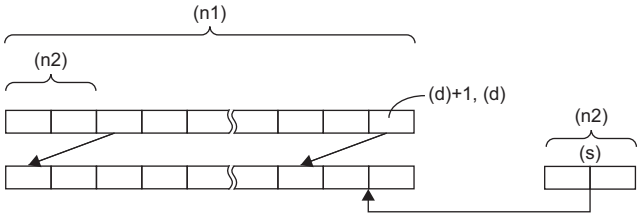
Instruction symbol	Processing details	Reference
SFTBR SFTBRP	Shifts the n-bit data starting from the specified device to the right by n bit(s). In the empty area after the shift, 0 is stored. 	Page 369 SFTBR(P)
SFTR SFTRP	Shifts bit data to the right by the (n2) bit(s) within the (n1) bits of data area starting from the specified device. In the empty area after the shift, specified data is stored. 	Page 371 SFTR(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 373 SFTBL(P)
SFTL SFTLP	Shifts bit data to the left by the (n2) bit(s) within the (n1) bits of data area starting from the specified device. In the empty area after the shift, specified data is stored. 	Page 375 SFTL(P)

■Shifting n-word data to the right/left by n word(s)

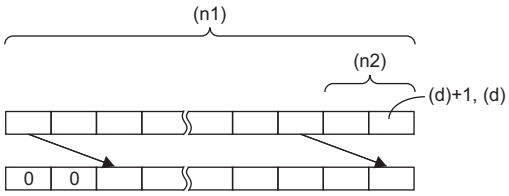
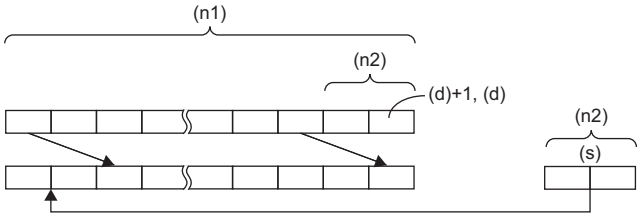
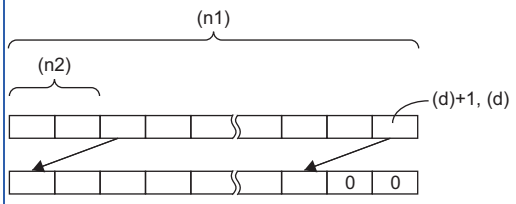
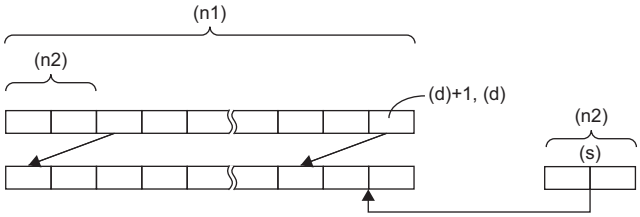
Instruction symbol	Processing details	Reference
SFTWR SFTWRP	Shifts the n-word data starting from the specified device to the right by n word(s). In the empty area after the shift, 0 is stored. 	Page 377 SFTWR(P)
WSFR WSFRP	Shifts word data to the right by the (n2) word(s) within the (n1) words of data area starting from the specified device. In the empty area after the shift, specified data is stored. 	Page 379 WSFR(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 381 SFTWL(P)
WSFL WSFLP	Shifts word data to the left by the (n2) word(s) within the (n1) words of data area starting from the specified device. In the empty area after the shift, specified data is stored. 	Page 383 WSFL(P)

■Shifting n double word(s) of data to the right/left by n double word(s)

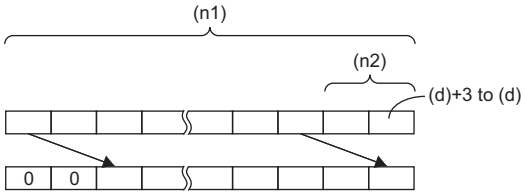
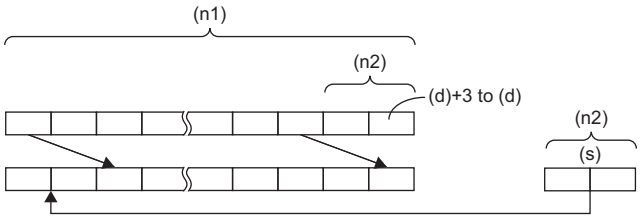
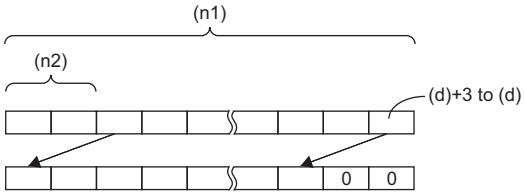
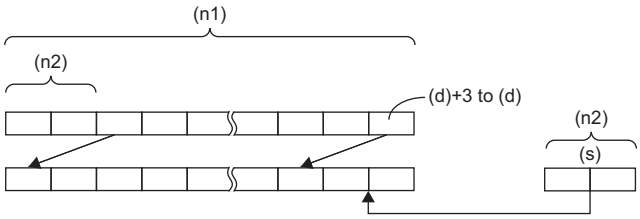
Instruction symbol	Processing details	Reference
SFTDWR SFTDWRP	Shifts the (n2) double word(s) of area to the right within the (n1) double word(s) of data area starting from the specified device. In the empty area after the shift, 0 is stored. 	Page 385 SFTDWR(P)
DWSFTR DWSFTRP	Shifts the (n2) double word(s) of area to the right within the (n1) double word(s) of data area starting from the specified device. In the empty area after the shift, specified data is stored. 	Page 387 DWSFTR(P)

Instruction symbol	Processing details	Reference
SFTDWL SFTDWLP	Shifts the (n2) double word(s) of area to the left within the (n1) double word(s) of data area starting from the specified device. In the empty area after the shift, 0 is stored. 	Page 389 SFTDWL(P)
DWSFTL DWSFTLP	Shifts the (n2) double word(s) of area to the left within the (n1) double word(s) of data area starting from the specified device. In the empty area after the shift, specified data is stored. 	Page 391 DWSFTL(P)

■Shifting n point(s) of single-precision real number data to the right/left by n point(s)

Instruction symbol	Processing details	Reference
SFTER SFTERP	Shifts the (n2) point(s) of area of the single-precision real number data to the right within the (n1) point(s) of area starting from the specified device. In the empty area after the shift, 0 is stored. 	Page 393 SFTER(P)
ESFTR ESFTRP	Shifts the (n2) point(s) of area of the single-precision real number data to the right within the (n1) point(s) of area starting from the specified device. In the empty area after the shift, specified data is stored. 	Page 395 ESFTR(P)
SFTEL SFTELP	Shifts the (n2) point(s) of area of the single-precision real number data to the left within the (n1) point(s) of area starting from the specified device. In the empty area after the shift, 0 is stored. 	Page 397 SFTEL(P)
ESFTL ESFTLP	Shifts the (n2) point(s) of area of the single-precision real number data to the left within the (n1) point(s) of area starting from the specified device. In the empty area after the shift, specified data is stored. 	Page 399 ESFTL(P)

■Shifting n point(s) of double-precision real number data to the right/left by n point(s)

Instruction symbol	Processing details	Reference
SFTEDR SFTEDRP	Shifts the (n2) point(s) of area of the double-precision real number data to the right within the (n1) point(s) of area starting from the specified device. In the empty area after the shift, 0 is stored. 	Page 401 SFTEDR(P)
EDSFTR EDSFTRP	Shifts the (n2) point(s) of area of the double-precision real number data to the right within the (n1) point(s) of area starting from the specified device. In the empty area after the shift, specified data is stored. 	Page 403 EDSFTR(P)
SFTEDL SFTEDLP	Shifts the (n2) point(s) of area of the double-precision real number data to the left within the (n1) point(s) of area starting from the specified device. In the empty area after the shift, 0 is stored. 	Page 405 SFTEDL(P)
EDSFTL EDSFTLP	Shifts the (n2) point(s) of area of the double-precision real number data to the left within the (n1) point(s) of area starting from the specified device. In the empty area after the shift, specified data is stored. 	Page 407 EDSFTL(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 521 MOV(P)
DMOV DMOVP	Transfers the 32-bit binary data in the device specified. (s)+1, (s) → (d)+1, (d)	Page 523 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 525 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 527 DCML(P)

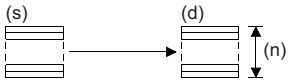
■Shifting data in units of 4 bits

Instruction symbol	Processing details	Reference
SMOV SMOVP	Distributes and combines data in units of 4 bits.	Page 529 SMOV(P)

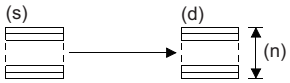
■Inverting and transferring 1-bit data

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 532 CMLB(P)

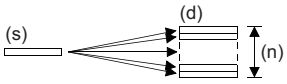
■Transferring 16-bit binary data block (16 bits)

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 534 BMOV(P)

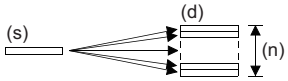
■Transferring 16-bit binary data block (32 bits)

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 537 BMOVL(P)

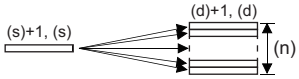
■Transferring the same 16-bit binary data block (16 bits)

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 539 FMOV(P)

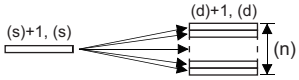
■Transferring the same 16-bit binary data block (32 bits)

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 541 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 543 DFMOV(P)

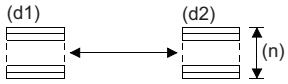
■Transferring the same 32-bit binary data block (32 bits)

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 545 DFMOVL(P)

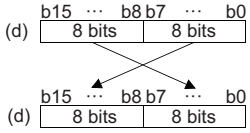
■Exchanging 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
XCH XCHP	Exchanges the 16-bit binary data specified. (d1) ◀——▶ (d2)	Page 547 XCH(P)
DXCH DXCHP	Exchanges the 32-bit binary data specified. (d1)+1, (d1) ◀——▶ (d2)+1, (d2)	Page 549 DXCH(P)

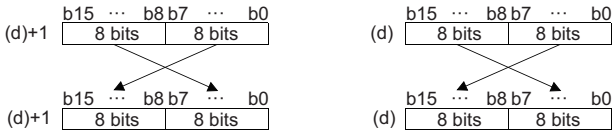
■Exchanging 16-bit binary block data

Instruction symbol	Processing details	Reference
BXCH BXCHP	Exchanges the (n) points of 16-bit binary data starting from the devices specified.  	Page 551 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 553 SWAP(P)

■Exchanging the upper and lower bytes of 32-bit binary data

Instruction symbol	Processing details	Reference
DSWAP	Exchanges upper and lower 8-bit data in the specified device. 	Page 554 DSWAP(P)
DSWAPP		

■Transferring 1-bit data

Instruction symbol	Processing details	Reference
MOVB	Stores the bit data in the device specified by (s) in the device specified by (d).	Page 556 MOVB(P)
MOVBP		

■Transferring n-bit data

Instruction symbol	Processing details	Reference
BLKMOVB	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 558 BLKMOVB(P)
BLKMOVBP		

2.3 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 563 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 566 GOEND

Program execution control instructions

■Disabling/enabling interrupt programs

Instruction symbol	Processing details	Reference
DI	Disables the execution of interrupt programs.	Page 567 DI, EI
EI	Clears the interrupt disabled state.	

■Disabling the interrupt program with specified priority or lower

Instruction symbol	Processing details	Reference
DI	Disables the execution of the interrupt program with a priority specified by (s) or lower until the EI instruction is executed.	Page 570 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 575 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 577 SIMASK

■Returning from the interrupt program

Instruction symbol	Processing details	Reference
IRET	Indicates the end of the processing of an interrupt program.	Page 579 IRET

■Resetting the watchdog timer

Instruction symbol	Processing details	Reference
WDT	Resets the watchdog timer.	Page 580 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 581 FOR, NEXT
NEXT		

■Forcibly terminating the FOR to NEXT instruction loop

Instruction symbol	Processing details	Reference
BREAK	Forcibly terminates the loop processing between the FOR and NEXT instructions, and passes the control to the specified pointer.	Page 583 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 585 CALL(P)
CALLP		

■Returning from the subroutine program called

Instruction symbol	Processing details	Reference
RET	Indicates the end of a subroutine program.	Page 589 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 590 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 594 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 599 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 604 XCALL

Program control instructions

■Changing the program execution type to standby type

Instruction symbol	Processing details	Reference
PSTOP	Changes the type of the specified program to standby type.	Page 609 PSTOP(P)
PSTOPP		

■Changing the program execution type to standby type (output off)

Instruction symbol	Processing details	Reference
POFF	Turns off the coil of the OUT instruction used in the specified program and changes the type of the specified program to standby type.	Page 611 POFF(P)
POFFP		

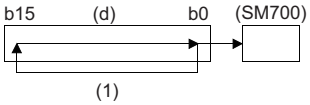
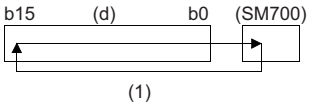
■Changing the program execution type to scan execution type

Instruction symbol	Processing details	Reference
PSCAN	Changes the type of the specified program to scan execution type.	Page 613 PSCAN(P)
PSCANP		

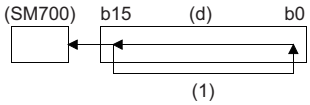
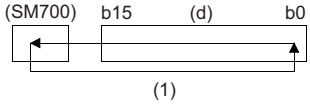
Data processing

Rotation instructions

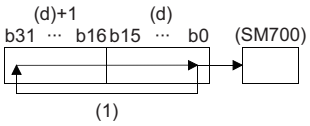
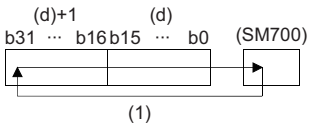
■Rotating 16-bit binary data to the right

Instruction symbol	Processing details	Reference
ROR	Rotates the 16-bit binary data to the right by (n) bit(s) (not including the carry flag).	Page 615 ROR(P), RCR(P)
RORP	 (1) Right rotation by (n) bits	
RCR	Rotates the 16-bit binary data to the right by (n) bit(s) (including the carry flag).	
RCRP	 (1) Right rotation by (n) bits	

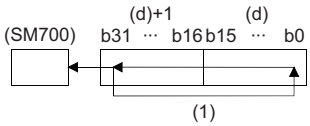
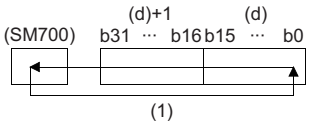
■Rotating 16-bit binary data to the left

Instruction symbol	Processing details	Reference
ROL	Rotates the 16-bit binary data to the left by (n) bit(s) (not including the carry flag).	Page 618 ROL(P), RCL(P)
ROLP	 (1) Left rotation by (n) bits	
RCL	Rotates the 16-bit binary data to the left by (n) bit(s) (including the carry flag).	
RCLP	 (1) Left rotation by (n) bits	

■Rotating 32-bit binary data to the right

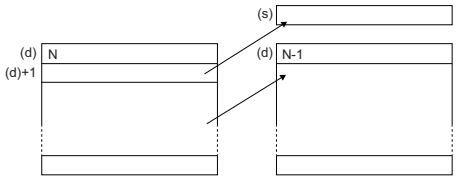
Instruction symbol	Processing details	Reference
DROR	Rotates the 32-bit binary data to the right by (n) bit(s) (not including the carry flag).	Page 621 DROR(P), DRCR(P)
DRORP	 (1) Right rotation by (n) bits	
DRCR	Rotates the 32-bit binary data to the right by (n) bit(s) (including the carry flag).	
DRCRP	 (1) Right rotation by (n) bits	

■Rotating 32-bit binary data to the left

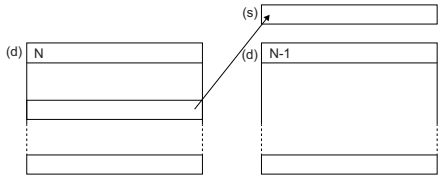
Instruction symbol	Processing details	Reference
DROL DROLP	Rotates the 32-bit binary data to the left by (n) bit(s) (not including the carry flag).  (1) Left rotation by (n) bits	Page 623 DROL(P), DRCL(P)
DRCL DRCLP	Rotates the 32-bit binary data to the left by (n) bit(s) (including the carry flag).  (1) Left rotation by (n) bits	

Data table operation instructions

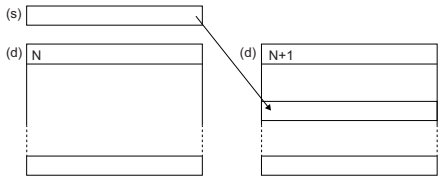
■Reading the oldest data from the data table

Instruction symbol	Processing details	Reference
FIFR FIFRP	Stores the data first stored in the table in the specified device.  N: Number of data	Page 625 FIFR(P)

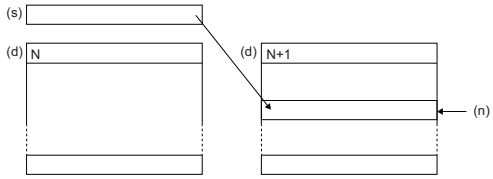
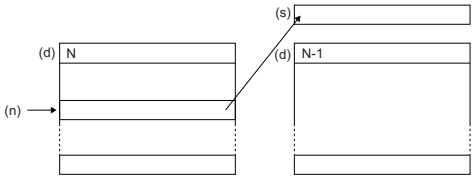
■Reading the newest data from the data table

Instruction symbol	Processing details	Reference
FPOP FPOPP	Stores the data last stored in the table in the specified device.  N: Number of data	Page 627 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 629 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 631 FINS(P)
FDEL FDELP	Deletes the data at the (n)th position in the data table.  N: Number of data	Page 633 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 636 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 638 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 641 SP.FREAD

■Writing data to the specified file

Instruction symbol	Processing details	Reference
SP.FWRITE	Writes data to the specified file.	Page 660 SP.FWRITE

■Deleting the specified file

Instruction symbol	Processing details	Reference
SP.FDELETE	Deletes the specified file or folder.	Page 678 SP.FDELETE

■Copying the specified file

Instruction symbol	Processing details	Reference
SP.FCOPY	Copies the specified file or folder.	Page 682 SP.FCOPY

■Moving the specified file

Instruction symbol	Processing details	Reference
SP.FMOVE	Moves the specified file or folder.	Page 687 SP.FMOVE

■Renaming the specified file

Instruction symbol	Processing details	Reference
SP.FRENAME	Renames the specified file or folder.	Page 692 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 696 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 701 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 703 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 705 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 707 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 709 ZONE(P)(_U)
DZONE DZONEP DZONE_U DZONEP_U	Adds a bias value to the specified input value (32-bit binary).	Page 711 DZONE(P)(_U)

■Scaling 16-bit/32-bit binary data (point coordinates)

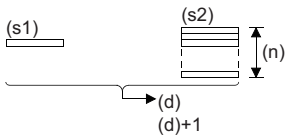
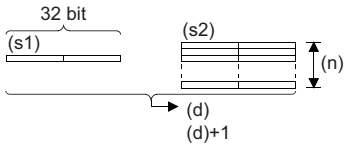
Instruction symbol	Processing details	Reference
SCL SCLP SCL_U SCLP_U	Scales the scaling conversion data (16-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.	Page 713 SCL(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 716 DSCL(P)(_U)

■Scaling 16-bit/32-bit binary data (XY coordinates)

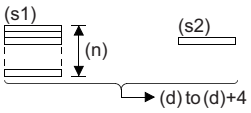
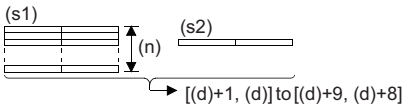
Instruction symbol	Processing details	Reference
SCL2 SCL2P SCL2_U SCL2P_U	Scales the scaling conversion data (16-bit data) in the device specified by (s2) on the basis of the input value in the device specified by (s1), and stores the operation result in the device specified by (d). Scaling conversion is performed based on the scaling conversion data stored in the device specified by (s2) and later.	Page 719 SCL2(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 721 DSCL2(P)(_U)

Data processing instructions

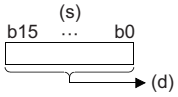
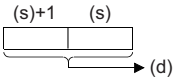
■Searching 16-bit/32-bit binary data

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 723 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 725 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 727 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 729 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 731 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 733 DSUM(P)

■Checking the bit status in 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
BON BONP	Checks whether (n) bit(s) of the specified device are on or off, and stores the result in the device specified by (d).	Page 735 BON(P)
DBON DBONP	Checks whether (n) bit(s) of the specified device are on or off, and stores the result in the device specified by (d).	Page 737 DBON(P)

■Searching for the maximum value of 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
MAX MAXP MAX_U MAXP_U	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 739 MAX(P)(_U)
DMAX DMAXP DMAX_U DMAXP_U	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 741 DMAX(P)(_U)

■Searching for the minimum value of 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
MIN MINP MIN_U MINP_U	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 743 MIN(P)(_U)
DMIN DMINP DMIN_U DMINP_U	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 745 DMIN(P)(_U)

■Sorting 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
SORTD SORTD_U	Sorts the (n) points of data in the device specified by (s1) in units of 16 bits. ("(n)×((n)-1)÷2" scanning required)	Page 747 SORTD(_U)
DSORTD DSORTD_U	Sorts the (n) points of data in the device specified by (s1) in units of 32 bits. ("(n)×((n)-1)÷2" scanning required)	Page 749 DSORTD(_U)

■Sorting 16-bit binary data table

Instruction symbol	Processing details	Reference
SORTTBL SORTTBL_U	Sorts the data rows in the 16-bit binary data table (sorting source) of ((n1) × (n2)) points specified by (s), based on the data in the column (n3) in ascending or descending order. (For the sorting, the data table where the value to the right of (s) in each column increases consecutively is used.) The result is stored in the 16-bit binary data table (sorting result) of ((n1) × (n2)) points specified by (d).	Page 751 SORTTBL(_U)

■Sorting 16-bit/32-bit binary data table 2

Instruction symbol	Processing details	Reference
SORTTBL2 SORTTBL2_U	Sorts the data rows in the 16-bit binary data table (sorting source) of ((n1) × (n2)) points specified by (s), based on the data in the column (n3) in ascending or descending order. (For the sorting, the data table where the value to the right of (s) in each row increases consecutively is used.) The result is stored in the 16-bit binary data table (sorting result) of ((n1) × (n2)) points specified by (d).	Page 755 SORTTBL2(_U)
DSORTTBL2 DSORTTBL2_U	Sorts the data rows in the 32-bit binary data table (sorting source) of ((n1) × (n2)) points specified by (s), based on the data in the column (n3) in ascending or descending order. (For the sorting, the data table where the values (odd number and even number) to the right of (s) in each row increase consecutively is used.) The result is stored in the 32-bit binary data table (sorting result) of ((n1) × (n2)) points specified by (d).	Page 759 DSORTTBL2(_U)

■Adding 16-bit binary data

Instruction symbol	Processing details	Reference
WSUM WSUM_U WSUMP WSUMP_U	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 763 WSUM(P)(_U)

■Adding 32-bit binary data

Instruction symbol	Processing details	Reference
DWSUM DWSUM_U DWSUMP DWSUMP_U	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 765 DWSUM(P)(_U)

■Calculating the mean value of 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
MEAN MEANP MEAN_U MEANP_U	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 767 MEAN(P)(_U)
DMEAN DMEANP DMEAN_U DMEANP_U	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 769 DMEAN(P)(_U)

■Calculating the square root of 16-bit/32-bit binary data

Instruction symbol	Processing details	Reference
SQRT SQRTP	Performs a square root operation of the specified 16-bit binary data. $\sqrt{(s)} \rightarrow (d)$	Page 771 SQRT(P)
DSQRT DSQRTP	Performs a square root operation of the specified 32-bit binary data. $\sqrt{(s)+1, (s)} \rightarrow (d)$	Page 773 DSQRT(P)

■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 774 CRC(P)
CRCP		

Check code instructions

■Check code

Instruction symbol	Processing details	Reference
CCD	Performs addition of the data stored in the devices specified by (s) to (s)+(n)-1 and calculate the horizontal parity, and stores the added data in the device specified by (d) and the horizontal parity in the device specified by (d)+1.	Page 776 CCD(P)
CCDP		

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 CPU module.	Page 779 LEDR

■Generating a continuation error

Instruction symbol	Processing details	Reference
PALERT	Generates a continuation error in the CPU module.	Page 781 PALERT(P)
PALERTP		

■Generating a stop error

Instruction symbol	Processing details	Reference
PABORT	Stops program execution and generates a stop error in the CPU module.	Page 783 PABORT

String processing

String processing instructions

■Comparing string data

Instruction symbol	Processing details	Reference
LD$=, AND$=, OR$=	Compares the character string specified by (s1) with the character string specified by (s2) (character by character).	Page 785 LD$□, 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 character 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 788 $+(P) [when two operands are set]
$+P		
$+	Connects the character 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 790 $+(P) [when three operands are set]
$+P		

■Transferring string data

Instruction symbol	Processing details	Reference
$MOV	Transfers the character strings in the device specified by (s) to the device specified by (d) and later.	Page 792 $MOV(P)
$MOVP		
$MOV_WS	Transfers the Unicode character strings in the device specified by (s) to the device specified by (d) and later.	Page 794 $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 796 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 800 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 805 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 809 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 character string consisting of the total number digits and the number of digits in the decimal part in the device specified by (s1), and stores the converted data in the word device specified by (d).	Page 813 STR(P)(_U)
STRP		
STR_U		
STRP_U		
DSTR	Converts the 2-word binary data in the device specified by (s2) to a decimal character string consisting of the total number digits and the number of digits in the decimal part in the device specified by (s1), and stores the converted data in the word device specified by (d).	Page 816 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 819 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 823 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 character string, and stores the converted data in the word device specified by (d).	Page 828 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 832 INT2ASC(P)
INT2ASCP		

■Converting Unicode character string to Shift JIS character string

Instruction symbol	Processing details	Reference
WS2SJIS	Converts the Unicode character string in the device specified by (s) to the shift JIS character string, and stores the converted data in the device specified by (d).	Page 834 WS2SJIS(P)
WS2SJISP		

■Converting shift JIS character string to Unicode character string (without byte order mark)

Instruction symbol	Processing details	Reference
SJIS2WS SJIS2WSP	Converts the shift JIS character string in the device specified by (s) to a Unicode character string, and stores the converted data in the device specified by (d).	Page 836 SJIS2WS(P)

■Converting shift JIS to Unicode (with byte order mark)

Instruction symbol	Processing details	Reference
SJIS2WSB SJIS2WSBP	Converts the shift JIS character string in the device specified by (s) to the Unicode character string, add a byte order mark to the head of the converted data, and stores it in the device specified by (d).	Page 838 SJIS2WSB(P)

■Detecting a string length

Instruction symbol	Processing details	Reference
LEN LENP	Stores the length (the number of characters) of the character string data, which is stored in the device specified by (s), in the device specified by (d).	Page 840 LEN(P)

■Extracting string data from the right/left

Instruction symbol	Processing details	Reference
RIGHT RIGHTP	Stores the (n) characters from the last character of the character string, which is stored in the device specified by (s), in the device specified by (d).	Page 842 RIGHT(P)
LEFT LEFTP	Stores the (n) characters from the first character of the character string, which is stored in the device specified by (s), in the device specified by (d).	Page 844 LEFT(P)

■Extracting/replacing the specified string data

Instruction symbol	Processing details	Reference
MIDR MIDRP	Retrieves the character 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 846 MIDR(P)
MIDW MIDWP	Retrieves the specified number of characters from the character string in the device specified by (s1), and stores the retrieved data at the location specified by (s2) in the character string stored in the device specified by (d).	Page 848 MIDW(P)

■Searching string data

Instruction symbol	Processing details	Reference
INSTR INSTRP	Searches the character string in the device specified by (s2), starting from the (s3)th character, for the character string in the device specified by (s1), and stores the matching location in the device specified by (d).	Page 851 INSTR(P)

■Inserting string data

Instruction symbol	Processing details	Reference
STRINS STRINSP	Inserts the character string data in the device specified by (s1) to the (s2)th character (insertion position) from the head of the character string data in the device specified by (d).	Page 853 STRINS(P)

■Deleting string data

Instruction symbol	Processing details	Reference
STRDEL STRDELP	Deletes the (n) characters starting from the position (deletion start position) specified by the (s)th character from the head of the character string data in the device specified by (d).	Page 855 STRDEL(P)

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 857 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 859 LDED□, ANDED□, ORED□
LDED<>, ANDED<>, ORED<>		
LDED>, ANDED>, ORED>		
LDED<=, ANDED<=, ORED<=		
LDED<, ANDED<, ORED<		
LDED>=, ANDED>=, ORED>=		

■Outputting a comparison result of single-precision real numbers

Instruction symbol	Processing details	Reference
ECMP	Compares the single-precision real number data specified by (s1) with the single-precision real number data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.	Page 862 ECMP(P)
ECMPP		

■Outputting a comparison result of double-precision real numbers

Instruction symbol	Processing details	Reference
EDCMP	Compares the double-precision real number data specified by (s1) with the double-precision real number data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.	Page 864 EDCMP(P)
EDCMPP		

■Outputting a band comparison result of single-precision real number

Instruction symbol	Processing details	Reference
EZCP	Compares the band between the single-precision real number specified by lower limit value (s1) and the single-precision real number specified by upper limit value (s2) with the single-precision real number in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.	Page 866 EZCP(P)
EZCPP		

■Outputting a band comparison result of double-precision real number

Instruction symbol	Processing details	Reference
EDZCP	Compares the band between the double-precision real number specified by lower limit value (s1) and the double-precision real number specified by upper limit value (s2) with the double-precision real number in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.	Page 868 EDZCP(P)
EDZCPP		

■Adding/subtracting single-precision real numbers

Instruction symbol	Processing details	Reference
E+	Adds single-precision real numbers. (Using two operands)	Page 870 E+(P) [when two operands are set]
E+P		
E+	Adds single-precision real numbers. (Using three operands)	Page 872 E+(P) [when three operands are set]
E+P		
E-	Performs subtraction between single-precision real numbers. (Using two operands)	Page 874 E-(P) [when two operands are set]
E-P		
E-	Performs subtraction between single-precision real numbers. (Using three operands)	Page 876 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 878 ED+(P) [when two operands are set]
ED+P		
ED+	Adds double-precision real numbers. (Using three operands)	Page 880 ED+(P) [when three operands are set]
ED+P		
ED-	Performs subtraction between double-precision real numbers. (Using two operands)	Page 882 ED-(P) [when two operands are set]
ED-P		
ED-	Performs subtraction between double-precision real numbers. (Using three operands)	Page 884 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 886 E*(P)
E*P		
E/	Performs division between single-precision real numbers.	Page 888 E/(P)
E/P		

■Multiplying/dividing double-precision real numbers

Instruction symbol	Processing details	Reference
ED*	Multiplies double-precision real numbers.	Page 890 ED*(P)
ED*P		
ED/	Performs division between double-precision real numbers.	Page 892 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 894 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 898 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 896 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 900 UDINT2FLT(P)
UDINT2FLTP		

■Converting double-precision real number to single-precision real number

Instruction symbol	Processing details	Reference
DBL2FLT DBL2FLTP	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 902 DBL2FLT(P)

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

Instruction symbol	Processing details	Reference
INT2DBL INT2DBLP	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 904 INT2DBL(P)
DINT2DBL DINT2DBLP	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 908 DINT2DBL(P)

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

Instruction symbol	Processing details	Reference
UINT2DBL UINT2DBLP	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 906 UINT2DBL(P)
UDINT2DBL UDINT2DBLP	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 910 UDINT2DBL(P)

■Converting single-precision real number to double-precision real number

Instruction symbol	Processing details	Reference
FLT2DBL FLT2DBLP	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 912 FLT2DBL(P)

■Converting string data to single-precision real number

Instruction symbol	Processing details	Reference
EVAL EVALP	Converts the character 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 914 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 918 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. (d)+1, (d) → (d)+1, (d) (1) (1) Real number	Page 920 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. (d)+3, (d)+2, (d)+1, (d) → (d)+3, (d)+2, (d)+1, (d) (1) (1) Real number	Page 921 EDNEG(P)

■Transferring single-precision real number

Instruction symbol	Processing details	Reference
EMOV EMOVP	Transfers single-precision real number data to the specified device. (s)+1, (s) → (d)+1, (d) (1) (1) Real number	Page 922 EMOV(P)

■Transferring double-precision real number

Instruction symbol	Processing details	Reference
EDMOV EDMOVP	Transfers double-precision real number data to the specified device. (s)+3, (s)+2, (s)+1, (s) → (d)+3, (d)+2, (d)+1, (d) (1) (1) Real number	Page 923 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 924 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 926 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 928 TAN(P)

■Calculating the arc sine of single-precision real number

Instruction symbol	Processing details	Reference
ASIN ASINP	Calculates the angle from the sine specified by a single-precision real number.	Page 930 ASIN(P)

■Calculating the arc cosine of single-precision real number

Instruction symbol	Processing details	Reference
ACOS ACOSP	Calculates the angle from the cosine specified by a single-precision real number.	Page 932 ACOS(P)

■Calculating the arc tangent of single-precision real number

Instruction symbol	Processing details	Reference
ATAN ATANP	Calculates the angle from the tangent specified by a single-precision real number.	Page 934 ATAN(P)

■Calculating the sine of double-precision real number

Instruction symbol	Processing details	Reference
SIND SINDP	Calculates the sine of the angle specified by a double-precision real number.	Page 936 SIND(P)

■Calculating the cosine of double-precision real number

Instruction symbol	Processing details	Reference
COSD COSDP	Calculates the cosine of the angle specified by a double-precision real number.	Page 938 COSD(P)

■Calculating the tangent of double-precision real number

Instruction symbol	Processing details	Reference
TAND TANDP	Calculates the tangent of the angle specified by a double-precision real number.	Page 940 TAND(P)

■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 942 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 944 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 946 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 948 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	Calculates the cosine of the angle specified by a BCD value.	Page 950 BCOS(P)
BCOSP	COS (s) → (d) / (d)+1 / (d)+2 (d): Sign (d)+1: Integral part (d)+2: Decimal part	

■Calculating the tangent of BCD data

Instruction symbol	Processing details	Reference
BTAN	Calculates the tangent of the angle specified by a BCD value.	Page 952 BTAN(P)
BTANP	TAN(s) → (d) / (d)+1 / (d)+2 (d): Sign (d)+1: Integral part (d)+2: Decimal part	

■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 954 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 956 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 958 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 960 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 962 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 964 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 966 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. $\sqrt{(s)+1, (s)} \longrightarrow (d)+1, (d)$	Page 968 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. $\sqrt{(s)+3, (s)+2, (s)+1, (s)} \longrightarrow (d)+3, (d)+2, (d)+1, (d)$	Page 970 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 972 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 974 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 976 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 978 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. $\sqrt{(s)} \longrightarrow$ (d) / (d)+1 (d): Integral part (d)+1: Decimal part	Page 980 BSQRT(P)
BDSQRT BDSQRTP	Calculates the square root of the value specified by a BCD 8-digit data. $\sqrt{(s)+1, (s)} \longrightarrow$ (d) / (d)+1 (d): Integral part (d)+1: Decimal part	Page 982 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 984 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 986 POWD(P)

■Calculating the common logarithm of single-precision real number

Instruction symbol	Processing details	Reference
LOG10	Calculates the logarithm using the common logarithm (using 10 as the base) of the value specified by a single-precision real number.	Page 988 LOG10(P)
LOG10P		

■Calculating the common logarithm of double-precision real number

Instruction symbol	Processing details	Reference
LOG10D	Calculates the logarithm using the common logarithm (using 10 as the base) of the value specified by a double-precision real number.	Page 990 LOG10D(P)
LOG10DP		

■Searching the maximum value of single-precision real number

Instruction symbol	Processing details	Reference
EMAX	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 992 EMAX(P)
EMAXP		

■Searching the maximum value of double-precision real number

Instruction symbol	Processing details	Reference
EDMAX	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 994 EDMAX(P)
EDMAXP		

■Searching the minimum value of single-precision real number

Instruction symbol	Processing details	Reference
EMIN	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 996 EMIN(P)
EMINP		

■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 998 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 1000 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 1001 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 1002 ZPUSH(P)
ZPUSHP		
ZPOP	Reads the data, which has been saved to the area specified by (d)and later, into the index register.	Page 1004 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 1005 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 1008 ZPOP(P)
ZPOPP		

File register operation instructions

■Switching the file register block number

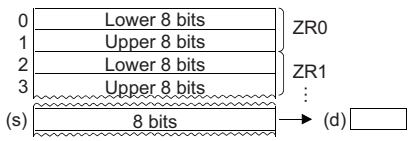
Instruction symbol	Processing details	Reference
RSET	Switches the block number of the file register used in the program to that stored in the device specified by (s).	Page 1010 RSET(P)
RSETP		

■Changing the file register file name

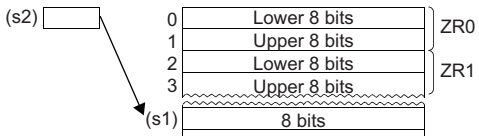
Instruction symbol	Processing details	Reference
QDRSET	Changes the file name of the file register used in the program.	Page 1012 QDRSET(P)
QDRSETP		

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 1014 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 1016 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. (s) → (d) (1) (2) (1) Indirect address of specified device (2) Device name	Page 1018 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. (s1) (s1)+1 Up Down Up (1) 0 1 2 3 4 5 6 7 6 5 4 3 2 1 0 -1 -2 -3 -2 -1 0 (2) (1) Current value of Cn (2) Contact of Cn	Page 1020 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. (s1) (s1)+1 (1) 0 1 2 3 4 5 4 3 2 1 0 -1 (2) (1) Current value of Cn (2) Contact of Cn	Page 1023 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. T_{ON}×(s) → (d) (s)=0:1, (s)=1:10, (s)=2:100 T_{ON}: On time of TTMR	Page 1025 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 1027 STMR

Pulse related instructions

■Measuring the density of pulses

Instruction symbol	Processing details	Reference
SPD	Counts the input pulses in the device specified by (s1) for the period specified by (s2), and stores the result data in the device specified by (d).	Page 1030 SPD

■Outputting pulses at regular intervals

Instruction symbol	Processing details	Reference
PLSY	Outputs the pulses at the frequency specified by (s), by the number of times specified by (n), to the output number (Y) in the device specified by (d).	Page 1032 PLSY

■Performing the pulse width modulation

Instruction symbol	Processing details	Reference
PWM	Outputs the on time specified by (s1) and the pulses in the period specified by (s2) to the output number (Y) in the device specified by (d).	Page 1034 PWM

Shortcut control

Shortcut control instruction

■Rotary table shortest direction control

Instruction symbol	Processing details	Reference
ROTC	Controls the rotary table divided by (n1) so that it makes s shortcut rotation from the stop position to the position specified by (s)+1.	Page 1036 ROTC

Ramp signal

Ramp signal instruction

■Ramp signal

Instruction symbol	Processing details	Reference
RAMPQ	Changes the value specified by (s1) to the value specified by (s2) by the number of times specified by (n). The current value is stored in the device specified by (d1)+0.	Page 1039 RAMPQ

Matrix input

Matrix input instruction

■Matrix input

Instruction symbol	Processing details	Reference
MTR	Reads 16 points by (n) columns of data from the device specified by (s), and stores it in the device specified by (d2) and later.	Page 1042 MTR

CPU module database access function

Database access instructions

■Opening the database

Instruction symbol	Processing details	Reference
DBOPEN	Connects to the database specified by (s), and makes it available.	Page 1051 DBOPEN(P)
DBOPENP		

■Closing the database

Instruction symbol	Processing details	Reference
DBCLOSE	Releases the identification number specified by (s) and the allocation of the database.	Page 1053 DBCLOSE(P)
DBCLOSEP		

■Adding a record to the database

Instruction symbol	Processing details	Reference
DBINSERT	Adds a record to the table specified by (s2) in the database corresponding to the identification number specified by (s1).	Page 1055 DBINSERT(P)
DBINSERTP		

■Updating the record in the database

Instruction symbol	Processing details	Reference
DBUPDATE	Updates all record that meets the condition specified by (s5) in the table specified by (s2) in the database specified by the identification number specified by (s1).	Page 1062 DBUPDATE(P)
DBUPDATEP		

■Searching the record in the database

Instruction symbol	Processing details	Reference
DBSELECT	Searches the records in the table specified by (s2) in the database corresponding to the identification number specified by (s1).	Page 1068 DBSELECT(P)
DBSELECTP		

■Deleting the record in the database

Instruction symbol	Processing details	Reference
DBDELETE	Deletes the record that meets the condition specified by (s3) in the table specified by (s2) in the database corresponding to the identification number specified by (s1).	Page 1076 DBDELETE(P)
DBDELETEP		

■Importing data to the database

Instruction symbol	Processing details	Reference
DBIMPORT	Imports the data set in the Unicode text file stored in the path specified by (s) to construct a database.	Page 1045 DBIMPORT(P)
DBIMPORTP		

■Exporting data from the database

Instruction symbol	Processing details	Reference
DBEXPORT	Exports the data stored in the database to the Unicode text file stored in the path specified by (s).	Page 1048 DBEXPORT(P)
DBEXPORTP		

■Starting a transaction

Instruction symbol	Processing details	Reference
DBTRANS	Declares the start of a transaction in relation to the database corresponding to the identification number specified by (s).	Page 1080 DBTRANS(P)
DBTRANSP		

■Committing a transaction

Instruction symbol	Processing details	Reference
DBCOMMIT	Commits the transaction in relation to the database corresponding to the identification number specified by (s).	Page 1082 DBCOMMIT(P)
DBCOMMITP		

■Performing a rollback

Instruction symbol	Processing details	Reference
DBROLBAK DBROLBAKP	Executes the rollback in relation to the database corresponding to the identification number specified by (s).	Page 1084 DBROLBAK(P)

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 CPU module. (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 1090 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 CPU module. (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 1092 DATEWR(P)

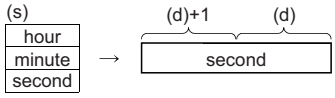
■Adding clock data

Instruction symbol	Processing details	Reference
DATE+ DATE+P	Adds time data. (s1) hour, minute, second + (s2) hour, minute, second → (d) hour, minute, second	Page 1094 DATE+(P)

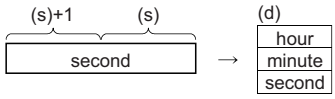
■Subtracting clock data

Instruction symbol	Processing details	Reference
DATE- DATE-P	Subtracts time data. (s1) hour, minute, second - (s2) hour, minute, second → (d) hour, minute, second	Page 1096 DATE-(P)

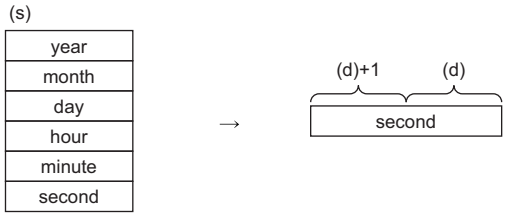
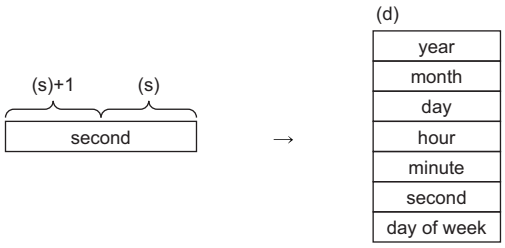
■Converting time data from hour/minute/second to second

Instruction symbol	Processing details	Reference
TIME2SEC TIME2SECP	Converts time data from hour/minute/second to second. 	Page 1098 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 1100 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 1102 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 1104 SEC2DATE(P)(_U)

■Comparing date data

Instruction symbol	Processing details	Reference
LDDT=, ANDDT=, ORDT=	Compares the specified date data, or compares the date data with the current date.	Page 1106 LDDT□, ANDDT□, ORDT□
LDDT<>, ANDDT<>, ORDT<>		
LDDT>, ANDDT>, ORDT>		
LDDT<=, ANDDT<=, ORDT<=		
LDDT<, ANDDT<, ORDT<		
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 1110 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 TCMPP	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 1113 TCMP(P)

■Outputting a band comparison result of time data

Instruction symbol	Processing details	Reference
TZCP TZCPP	Compares the band between the time data of lower limit value (s1) and the time data of upper limit value (s2) with the time data (s3) to be compared, and according to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.	Page 1115 TZCP(P)

■Reading expansion clock data

Instruction symbol	Processing details	Reference
S.DATERD SP.DATERD	Reads clock data including millisecond from the clock elements in the CPU module. (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	Page 1117 S(P).DATERD

■Adding expansion clock data

Instruction symbol	Processing details	Reference
S.DATE+ SP.DATE+	Adds time data. (s1): hour, minute, second, —, 1/1000 second + (s2): hour, minute, second, —, 1/1000 second → (d): hour, minute, second, —, 1/1000 second	Page 1119 S(P).DATE+

■Subtracting expansion clock data

Instruction symbol	Processing details	Reference
S.DATE- SP.DATE-	Subtracts time data. (s1) hour / minute / second / — / 1/1000 second - (s2) hour / minute / second / — / 1/1000 second → (d) hour / minute / second / — / 1/1000 second	Page 1121 S(P).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. (d) (n1) (n2) (n1): (n1) scans (n2): (n2) scans (d): SM420 to SM424	Page 1123 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 1125 TIMCHK

■Hour meter

Instruction symbol	Processing details	Reference
HOURM	Measures the period of time for which the start contact is ON in units of hour, and turns on the device specified by (d2) when the accumulated ON time reaches the time (16-bit binary data) specified in (s).	Page 1127 HOURM
DHOURM	Measures the period of time for which the start contact is ON in units of hour, and turns on the device specified by (d2) when the accumulated ON time reaches the time (32-bit binary data) specified in (s).	Page 1129 DHOURM

Module access

Module access instructions

■Performing I/O refresh

Instruction symbol	Processing details	Reference
RFS RFSP	Performs partial refresh of the relevant input/output during one scan.	Page 1131 RFS(P)

■Selecting refresh to be performed

Instruction symbol	Processing details	Reference
COM COMP	Performs refresh and service processing for various modules when the input condition is met.	Page 1133 COM(P)

■Performing module refresh

Instruction symbol	Processing details	Reference
S.ZCOM SP.ZCOM	Performs refresh processing for the specified modules.	Page 1135 S(P).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 module and other CPU modules.	Page 1137 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 module and other CPU 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 own CPU module.	Page 1141 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 own CPU module.	
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 module and other CPU modules.	Page 1146 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 module and other CPU 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 own CPU module.	Page 1150 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 own CPU module.	
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 1155 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 1159 UNIINFRD(P)
UNIINFRDP		

Parameter setting operation

Routing information instructions

■Reading routing information

Instruction symbol	Processing details	Reference
S.RTREAD	Reads the data set by routing parameters.	Page 1164 S(P).RTREAD
SP.RTREAD		

■Registering routing information

Instruction symbol	Processing details	Reference
S.RTWRITE	Registers the routing information to the area specified by a routing parameter.	Page 1166 S(P).RTWRITE
SP.RTWRITE		

CPU module data logging function

Logging instructions

■Setting/resetting trigger logging

Instruction symbol	Processing details	Reference
LOGTRG	Generates a trigger for trigger logging. Data sampled for the number of records (specified in the trigger logging setting parameter using the engineering tool) are stored in the data logging file.	Page 1168 LOGTRG
LOGTRGR	Resets the trigger condition.	Page 1170 LOGTRGR

Recording function

Data collection instruction

■Setting data collection trigger

Instruction symbol	Processing details	Reference
DATATRG	Collects data for the specified setting number of the recording function.	Page 1172 DATATRG

Built-in Ethernet function instructions

Open/close processing instructions

■Opening a connection

Instruction symbol	Processing details	Reference
SP.SOCOPEN	Opens the connection specified by (s1).	Page 1174 SP.SOCOPEN

■Closing a connection

Instruction symbol	Processing details	Reference
SP.SOCCLOSE	Closes the connection specified by (s1). (Closing a connection)	Page 1177 SP.SOCCLOSE

Socket communications instructions

■Reading receive data during the END processing

Instruction symbol	Processing details	Reference
SP.SOCRCV	Reads the receive data of the connection specified by (s1) during END processing from the socket communication receive data area.	Page 1179 SP.SOCRCV

■Reading receive data when the instruction is executed

Instruction symbol	Processing details	Reference
S.SOCRCVS	Reads the receive data of the connection specified by (s) during instruction execution from the socket communication receive data area.	Page 1183 S.SOCRCVS

■Sending data

Instruction symbol	Processing details	Reference
SP.SOCSND	Sends the data in the device specified by (s3) to the external device of the connection specified by (s1).	Page 1185 SP.SOCSND

■Reading connection information

Instruction symbol	Processing details	Reference
SP.SOCCINF	Reads the connection information of the connection specified by (s1).	Page 1188 SP.SOCCINF

■Changing the communication target (UDP/IP)

Instruction symbol	Processing details	Reference
SP.SOCCSET	Changes the communication target IP address and port number of the connection specified by (s1). (UDP/IP communications only)	Page 1190 SP.SOCCSET

■Changing the receive mode

Instruction symbol	Processing details	Reference
SP.SOCRMODE	Changes the TCP receive mode and receive data size for the connection specified by (s1).	Page 1192 SP.SOCRMODE

■Reading socket communications receive data

Instruction symbol	Processing details	Reference
S.SOCRDATA SP.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 1196 S(P).SOCRDATA

Predefined protocol support function instruction

■Executing the registered protocols

Instruction symbol	Processing details	Reference
SP.ECPRTCL	Executes the protocol specified by the communication protocol support tool of the engineering tool.	Page 1198 SP.ECPRTCL

SLMP frame send instruction

■Sending an SLMP frame

Instruction symbol	Processing details	Reference
SP.SLMPSND	Sends SLMP messages to the SLMP-compatible device.	Page 1206 SP.SLMPSND

File transfer function instructions

■Sending FTP client files

Instruction symbol	Processing details	Reference
SP.FTPPUT	Sends files in the CPU module, which are specified by (s2), to the folder path of the FTP server, which is specified by (s3).	Page 1213 SP.FTPPUT

■Retrieving FTP client files

Instruction symbol	Processing details	Reference
SP.FTPGET	Retrieves files on the FTP server, which are specified by (s2), to the folder path of the CPU module, which is specified by (s3).	Page 1223 SP.FTPGET

PID operation instruction

Performing 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 1245 PID

PID control instructions

PID control instructions (inexact differential)

■Registering the PID control data to the CPU module

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 CPU module to enable PID control.	Page 1260 S(P).PIDINIT
SP.PIDINIT		

■Performing PID operation

Instruction symbol	Processing details	Reference
S.PIDCONT	Measures the sampling cycle and performs PID operation.	Page 1263 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 1266 S(P).PIDSTOP
SP.PIDSTOP		
S.PIDRUN	Starts the PID operation of the loop number in the device specified by (s).	Page 1267 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 1268 S(P).PIDPRMW
SP.PIDPRMW		

PID control instructions (exact differential)

■Registering the PID control data to the CPU module

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 CPU module to enable PID control.	Page 1272 PIDINIT(P)
PIDINITP		

■Performing PID operation

Instruction symbol	Processing details	Reference
PIDCONT	Measures the sampling cycle and performs PID operation.	Page 1274 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 1277 PIDSTOP(P)
PIDSTOPP		
PIDRUN	Starts the PID operation of the loop number in the device specified by (s).	Page 1278 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 1279 PIDPRMW(P)
PIDPRMWP		

Process control instructions

For details on the process control instructions, refer to the following.

📖 MELSEC iQ-R Programming Manual (Process Control Function Blocks/Instructions)

Point

When a process control program is created, using process control function blocks is recommended.

Process control function blocks have features as follows.

- A process control program can be easily created by placing and connecting FB elements.
- Since the initial value of the function block can be set in the "FB Property" window of the engineering tool, the program for the initial value setting is not required.
- An operation constant can be input to a label indicating a tag name without being conscious of address of a device.
- The operating status of a tag FB can be checked and controlled by accessing the tag data from the faceplate of an engineering tool.

For details on the process control function blocks, refer to the following.

📖 MELSEC iQ-R Programming Manual (Process Control Function Blocks/Instructions)

Multiple CPU dedicated instructions

Reading device data from another CPU module

Instruction symbol	Processing details	Reference
D.DDRD	Reads the data in the device of another CPU module specified by (n), and stores the data to the read-source CPU module in a multiple CPU system.	Page 1285 D(P).DDRD, M(P).DDRD
DP.DDRD		
M.DDRD		
MP.DDRD		

Writing device data to another CPU module

Instruction symbol	Processing details	Reference
D.DDWR	Writes the data specified in the own CPU module to another CPU module specified by (n) in a multiple CPU system.	Page 1288 D(P).DDWR, M(P).DDWR
DP.DDWR		
M.DDWR		
MP.DDWR		

Motion CPU dedicated instructions

For available Motion CPU dedicated instructions, refer to the following.

📖 MELSEC iQ-R Motion Controller Programming Manual (Program Design)

SFC program instructions

SFC control instructions

■Checking the status of a step

Instruction symbol	Processing details	Reference
LD [S□/BL□\S□]	Outputs the status (active or inactive) of the specified step as the operation result. (Normally open contact instruction)	Page 1291 LD, LDI, AND, ANI, OR, ORI [S□/BL□\S□]
LDI [S□/BL□\S□]	Outputs the status (active or inactive) of the specified step as the operation result. (Normally closed contact instruction)	
AND [S□/BL□\S□]	Performs an AND operation between the status (active or inactive) of the specified step and the previous operation result(s), and output the operation result. (Normally open contact series connection instruction)	
ANI [S□/BL□\S□]	Performs an AND operation between the status (active or inactive) of the specified step and the previous operation result(s), and output the operation result. (Normally closed contact series connection instruction)	
OR [S□/BL□\S□]	Performs an OR operation between the status (active or inactive) of the specified step and the previous operation result(s), and output the operation result. (Single normally open contact parallel connection instruction)	
ORI [S□/BL□\S□]	Performs an OR operation between the status (active or inactive) of the specified step and the previous operation result(s), and output the operation result. (Single normally closed contact parallel connection instruction)	

■Checking the status of a block

Instruction symbol	Processing details	Reference
LD [BL□]	Outputs the status (active or inactive) of the specified block as the operation result. (Normally open contact instruction)	Page 1294 LD, LDI, AND, ANI, OR, ORI [BL□]
LDI [BL□]	Outputs the status (active or inactive) of the specified block as the operation result. (Normally closed contact instruction)	
AND [BL□]	Performs an AND operation between the status (active or inactive) of the specified block and the previous operation result(s), and output the operation result. (Normally open contact series connection instruction)	
ANI [BL□]	Performs an AND operation between the status (active or inactive) of the specified block and the previous operation result(s), and output the operation result. (Normally closed contact series connection instruction)	
OR [BL□]	Performs an OR operation between the status (active or inactive) of the specified block and the previous operation result(s), and output the operation result. (Single normally open contact parallel connection instruction)	
ORI [BL□]	Performs an OR operation between the status (active or inactive) of the specified block and the previous operation result(s), and output the operation result. (Single normally closed contact parallel connection instruction)	

■Batch-reading the status of steps

Instruction symbol	Processing details	Reference
MOV [K4S□/BL□\K4S□]	Batch-reads (in units of 16-bit binary data) the status (active or inactive) of steps in the specified block, and stores the read data in the specified device.	Page 1296 MOV(P) [K4S□/BL□\K4S□]
MOVP [K4S□/BL□\K4S□]		
DMOV [K8S□/BL□\K8S□]	Batch-reads (in units of 32-bit binary data) the status (active or inactive) of steps in the specified block, and stores the read data in the specified device.	Page 1299 DMOV(P) [K8S□/BL□\K8S□]
DMOVP [K8S□/BL□\K8S□]		
BMOV [K4S□/BL□\K4S□]	Batch-reads (in units of the specified number of words starting from the specified step) the status (active or inactive) of steps in the specified block.	Page 1302 BMOV(P) [K4S□/BL□\K4S□]
BMOVP [K4S□/BL□\K4S□]		

■Starting a block

Instruction symbol	Processing details	Reference
SET [BL□]	Activates the specified block, and executes a step sequence starting from an initial step.	Page 1305 SET [BL□]

■Ending a block

Instruction symbol	Processing details	Reference
RST [BL□]	Deactivates the specified block.	Page 1307 RST [BL□]

■Pausing a block

Instruction symbol	Processing details	Reference
PAUSE [BL□]	Temporarily stops the step sequence in the specified block.	Page 1309 PAUSE [BL□]

■Restarting a block

Instruction symbol	Processing details	Reference
RSTART [BL□]	Releases the temporary stop, and restarts the sequence from the step where the sequence was stopped in the specified block.	Page 1311 RSTART [BL□]

■Activating a step

Instruction symbol	Processing details	Reference
SET [S□/BL□\S□]	Activates the specified step.	Page 1313 SET [S□/BL□\S□]

■Deactivating a step

Instruction symbol	Processing details	Reference
RST [S□/BL□\S□]	Deactivates the specified step.	Page 1315 RST [S□/BL□\S□]

■Switching a target block

Instruction symbol	Processing details	Reference
BRSET	Specifies an SFC control instruction target block No.	Page 1317 BRSET

SFC dedicated instruction

■Creating a dummy transition condition

Instruction symbol	Processing details	Reference
TRAN	A dummy output which satisfies a transition condition	Page 1319 TRAN

Redundant system instructions

System switching

Instruction symbol	Processing details	Reference
SP.CONTSW	Switches the systems (control system and standby system) during END processing of the scan where the instruction is executed.	Page 1320 SP.CONTSW

Disabling/enabling system switching

Instruction symbol	Processing details	Reference
DCONTSW	Disables manual system switching.	Page 1324 DCONTSW, ECONTSW
ECONTSW	Enables manual system switching.	

Writing data from the standby system to the control system

Instruction symbol	Processing details	Reference
CONTWR	Writes data from the standby system to the control system in a program executed in both systems.	Page 1326 CONTWR(P)
CONTWRP		

Safety system instructions

Reading safety data identify check information

Instruction symbol	Processing details	Reference
SP.SIDRD	Reads the identifier for the safety data identify check file from the specified file.	Page 1331 SP.SIDRD

MEMO

3 MODULE DEDICATED INSTRUCTIONS

For details on the module dedicated instructions, refer to the following.

📖 MELSEC iQ-R Programming Manual (Module Dedicated Instructions)

MEMO

4 STANDARD FUNCTIONS/FUNCTION BLOCKS

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.

4.1 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 1340 BOOL_TO_WORD(_E)
BOOL_TO_WORD_E		
BOOL_TO_DWORD	Converts a value from BOOL data type to DWORD data type.	Page 1342 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 1343 BOOL_TO_INT(_E)
BOOL_TO_INT_E		
BOOL_TO_DINT	Converts a value from BOOL data type to DINT data type.	Page 1344 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 1345 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 1346 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 1347 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 1348 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 1349 WORD_TO_INT(_E)
WORD_TO_INT_E		
WORD_TO_DINT	Converts a value from WORD data type to DINT data type.	Page 1350 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 1352 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 1353 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 1354 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 1355 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 1357 DWORD_TO_INT(_E)
DWORD_TO_INT_E		
DWORD_TO_DINT	Converts a value from DWORD data type to DINT data type.	Page 1359 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 1360 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 1361 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 1362 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 1363 INT_TO_WORD(_E)
INT_TO_WORD_E		
INT_TO_DWORD	Converts a value from INT data type to DWORD data type.	Page 1364 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 1366 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 1367 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 1369 INT_TO_REAL(_E)
INT_TO_REAL_E		
INT_TO_LREAL	Converts a value from INT data type to LREAL data type.	Page 1370 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 1371 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 1372 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 1374 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 1375 DINT_TO_WORD(_E)
DINT_TO_WORD_E		
DINT_TO_DWORD	Converts a value from DINT data type to DWORD data type.	Page 1377 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 1378 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 1379 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 1381 DINT_TO_REAL(_E)
DINT_TO_REAL_E		
DINT_TO_LREAL	Converts a value from DINT data type to LREAL data type.	Page 1382 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 1383 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 1384 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 1386 BCD_TO_INT(_E)
BCD_TO_INT_E		
BCD_TO_DINT	Converts a value from BCD data type to DINT data type.	Page 1388 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 1391 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 1393 REAL_TO_INT(_E)
REAL_TO_INT_E		
REAL_TO_DINT	Converts a value from REAL data type to DINT data type.	Page 1395 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 1397 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 1399 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 1402 LREAL_TO_INT(_E)
LREAL_TO_INT_E		
LREAL_TO_DINT	Converts a value from LREAL data type to DINT data type.	Page 1404 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 1406 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 1408 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 1409 TIME_TO_WORD(_E)
TIME_TO_WORD_E		
TIME_TO_DWORD	Converts a value from TIME data type to DWORD data type.	Page 1410 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 1411 TIME_TO_INT(_E)
TIME_TO_INT_E		
TIME_TO_DINT	Converts a value from TIME data type to DINT data type.	Page 1412 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 1413 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 1415 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 1416 STRING_TO_WORD(_E)
STRING_TO_WORD_E		
STRING_TO_DWORD	Converts a value from STRING data type to DWORD data type.	Page 1417 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 1418 STRING_TO_INT(_E)
STRING_TO_INT_E		
STRING_TO_DINT	Converts a value from STRING data type to DINT data type.	Page 1420 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 1422 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 1424 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 1427 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 1429 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 1430 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 1431 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 1432 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 1433 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 1434 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 1436 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 1438 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 1440 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 1441 ABS(_E)
ABS_E		

■Calculating the square root

Function symbol	Processing details	Reference
SQRT	Calculates the square root of an input value.	Page 1443 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 1444 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 1445 LOG(_E)
LOG_E		

■Calculating the exponent

Function symbol	Processing details	Reference
EXP	Outputs the exponent of an input value.	Page 1447 EXP(_E)
EXP_E		

■Calculating the sine/cosine/tangent

Function symbol	Processing details	Reference
SIN	Outputs the sine of an input value.	Page 1448 SIN(_E)
SIN_E		
COS	Outputs the cosine of an input value.	Page 1449 COS(_E)
COS_E		
TAN	Outputs the tangent of an input value.	Page 1450 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 1451 ASIN(_E)
ASIN_E		
ACOS	Outputs the arc cosine (COS^{-1}) of an input value.	Page 1452 ACOS(_E)
ACOS_E		
ATAN	Outputs the arc tangent (TAN^{-1}) of an input value.	Page 1453 ATAN(_E)
ATAN_E		

Arithmetic operation functions

■Addition

Function symbol	Processing details	Reference
ADD	Outputs the sum of input values ((s1)+(s2)+···+(s28)).	Page 1454 ADD(_E)
ADD_E		

■Multiplication

Function symbol	Processing details	Reference
MUL	Outputs the product of input values ((s1)×(s2)×···×(s28)).	Page 1457 MUL(_E)
MUL_E		

■Subtraction

Function symbol	Processing details	Reference
SUB	Outputs the difference between input values ((s1)-(s2)).	Page 1459 SUB(_E)
SUB_E		

■Division

Function symbol	Processing details	Reference
DIV	Outputs the quotient of input values ((s1)÷(s2)).	Page 1462 DIV(_E)
DIV_E		

■Remainder

Function symbol	Processing details	Reference
MOD	Outputs the remainder of input values ((s1)÷(s2)).	Page 1464 MOD(_E)
MOD_E		

■Exponentiation

Function symbol	Processing details	Reference
EXPT	Outputs the exponentiation of an input value.	Page 1466 EXPT(_E)
EXPT_E		

■Assignment (move operation)

Function symbol	Processing details	Reference
MOVE	Outputs the assignment value of an input value.	Page 1467 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 1469 SHL(_E)
SHL_E		
SHR	Shifts the input value to the right by (n) bit(s), and outputs the operation result.	Page 1471 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 1473 ROL(_E)
ROL_E		
ROR	Rotates the input value to the right by (n) bit(s), and outputs the operation result.	Page 1475 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 1477 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 1480 NOT(_E)
NOT_E		

Selection functions

■Selecting a value

Function symbol	Processing details	Reference
SEL	Outputs the selected input value.	Page 1481 SEL(_E)
SEL_E		

■Selecting the maximum/minimum value

Function symbol	Processing details	Reference
MAX	Outputs the maximum input value.	Page 1483 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 1485 LIMIT(_E)
LIMIT_E		

■Multiplexer

Function symbol	Processing details	Reference
MUX	Outputs one of the input values.	Page 1488 MUX(_E)
MUX_E		

Comparison functions

■Comparing data

Function symbol	Processing details	Reference
GT	Outputs the comparison result of input values.	Page 1490 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 1492 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 1494 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 1496 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 1498 MID(_E)
MID_E		

■Concatenating string data

Function symbol	Processing details	Reference
CONCAT	Concatenates character strings, and outputs the operation result.	Page 1500 CONCAT(_E)
CONCAT_E		

■Inserting string data

Function symbol	Processing details	Reference
INSERT	Inserts a character string into another string, and outputs the operation result.	Page 1502 INSERT(_E)
INSERT_E		

■Deleting string data

Function symbol	Processing details	Reference
DELETE	Deletes the specified range in a character string, and outputs the operation result.	Page 1504 DELETE(_E)
DELETE_E		

■Replacing string data

Function symbol	Processing details	Reference
REPLACE	Replaces the specified range in a character string, and outputs the operation result.	Page 1506 REPLACE(_E)
REPLACE_E		

■Searching string data

Function symbol	Processing details	Reference
FIND	Searches a character string, and outputs the operation result.	Page 1509 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 1511 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 1513 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 1515 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 1517 DIV_TIME(_E)
DIV_TIME_E		

4.2 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 1520 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 1522 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 1524 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 1526 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 1528 CTU(_E)
CTU_E		

■Down counter

Function block symbol	Processing details	Reference
CTD	Counts down the number of rising edges of a signal.	Page 1530 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 1532 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 1535 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 1537 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 1540 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 1543 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 1545 TIMER_□_M
TIMER_100_FB_M		
TIMER_HIGH_FB_M		
TIMER_LOW_FB_M		
TIMER_CONT_FB_M		
TIMER_CONTHFB_M		

PART 3 SEQUENCE INSTRUCTIONS

This part consists of the following chapters.

5 SEQUENCE INSTRUCTIONS

5 SEQUENCE INSTRUCTIONS

5.1 Contact Instructions

Operation start, series connection, parallel connection

LD, LDI, AND, ANI, OR, ORI

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

- LD: Normally open contact operation start, LDI: Normally closed contact operation start

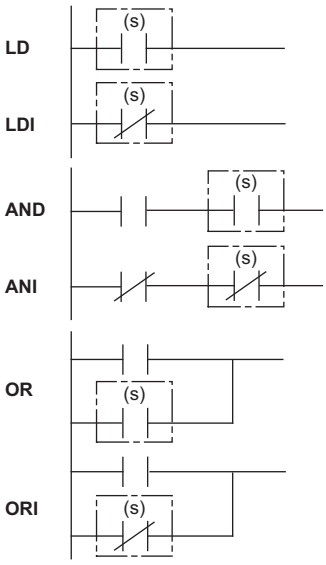
These instructions output the on/off information of the specified device as the operation result.

- 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	—	○	—	○	—	—	—	○

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

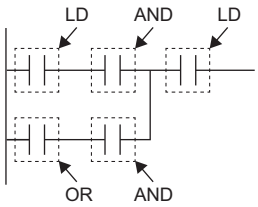
- OR is a single normally open contact parallel connection instruction and ORI is a single normally open contact parallel connection instruction. These instructions read the on/off information*1 of the specified bit device, perform an OR operation with the previous operation result, and output the operation 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

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

- 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

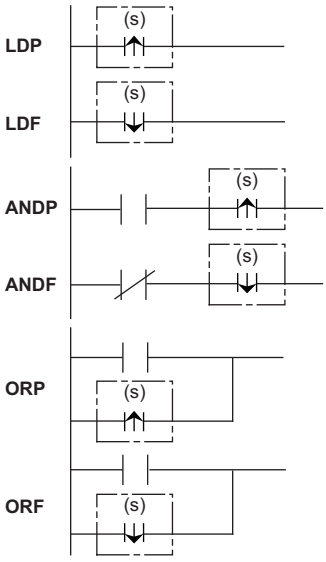
This instruction turns on only at the falling edge (on to off) of the specified bit device.

- ANDP: Rising edge pulse series connection, ANDF: Falling edge pulse series connection

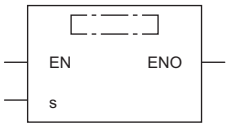
These instructions perform an AND operation with the previous operation result.

- 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	—	○	—	○	—	—	—	○

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

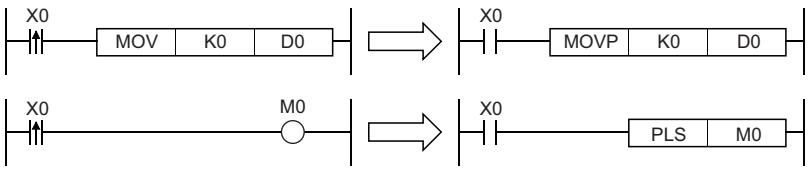
Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	—

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 150 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

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

- 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

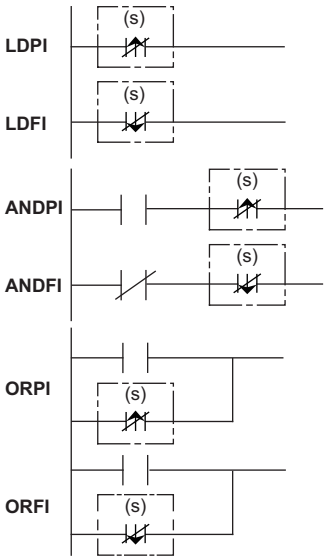
This instruction turns on when the specified bit device is at the rising edge (off to on), off, or on.

- ANDPI: Rising edge pulse NOT series connection, ANDFI: Falling edge pulse NOT series connection

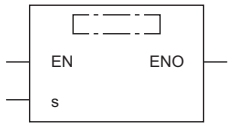
These instructions perform an AND operation with the previous operation result.

- 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	—	○	—	○	—	—	—	○

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	—

Processing details

■LDPI, LDFI

- LDFI 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 150 LD, LDI, AND, ANI, OR, ORI)

Operation error

There is no operation error.

5.2 Association Instructions

Ladder block series/parallel connection

ANB, ORB

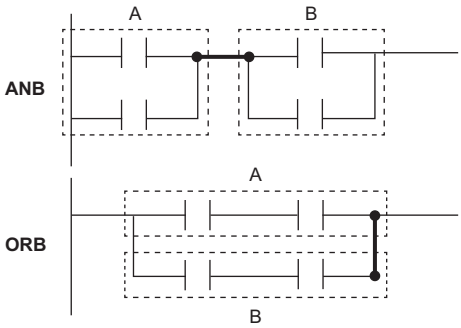
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

- 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.

Storing/reading/clearing the operation result

MPS, MRD, MPP

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

- MPS: Storing the operation result

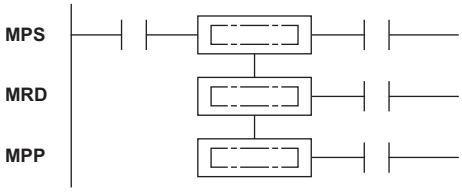
This instruction stores the operation result (on/off) immediately before the MPS instruction.

- MRD: Reading the operation result

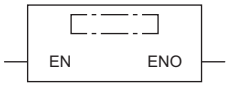
This instruction reads the operation result stored by using the MPS instruction.

- MPP: Clearing the operation result

This instruction clears the operation result stored by using the MPS instruction.

Ladder	ST
 (In the ladder display, these instructions are hidden.)	ENO:=MPS(EN); ENO:=MRD(EN); ENO:=MPP(EN);

FBD/LD



■Execution condition

Instruction	Execution condition
MPS MRD MPP	Every scan

Processing details

■MPS

- This instruction stores the operation result (on/off) immediately before the MPS instruction.
- Up to 16 MPS instructions can be used consecutively. If the MPP instruction is used in the middle of the program, the number of MPS instructions used is decremented by one.

■MRD

- This instruction reads the operation result stored by using the MPS instruction, and performs operations from the next step based on the operation result.

■MPP

- This instruction reads the operation result stored by using the MPS instruction, and performs operations from the next step based on the operation result.
- This instruction clears the operation result stored by using the MPS instruction.
- This instruction decrements the number of MPS instructions used in the program by one.

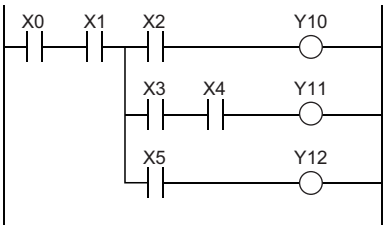
Operation error

There is no operation error.

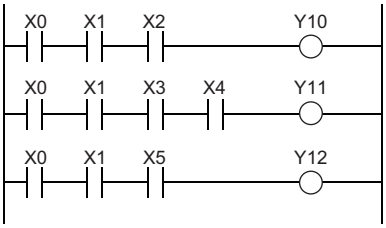
Point

- The following are the ladder program examples.

[Ladder program using the MPS, MRD, and MPP instructions]



[Ladder program not using the MPS, MRD, or MPP instruction]



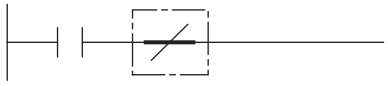
- Use the same number of MPS instructions as that of MPP instructions. If the numbers of MPS and MPP instructions are different, the ladder is not displayed correctly on the engineering tool (ladder mode).

Inverting the operation result

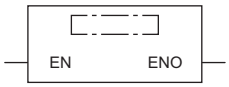
INV

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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 150 Operation start, series connection, parallel connection). The INV instruction cannot be used at the LD and OR (☞ Page 150 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 159 ANB, ORB

Converting the operation result into a pulse

MEP, MEF

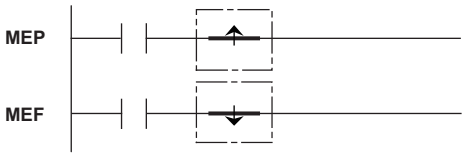
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

- MEP: Converting the operation result into a pulse (rising edge)

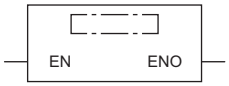
This instruction turns on at the rising edge (off to on) of the operation result up to the MEP instruction.

- MEP: 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 164 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 150 Operation start, series connection, parallel connection). The MEP or MEF instruction cannot be used at the LD and OR (☞ Page 150 Operation start, series connection, parallel connection) positions.

Converting the edge relay operation result into a pulse

EGP, EGF

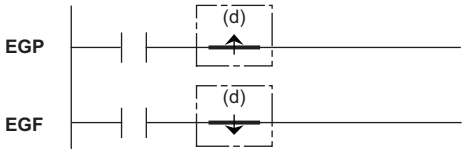
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- EGP: Converting the edge relay operation result into a pulse (rising edge)

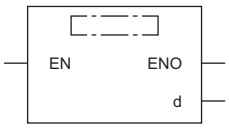
This instruction stores the operation result up to the EGP instruction in the edge relay (V). The instruction turns on at the rising edge (off to on) of the operation result.

- 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.

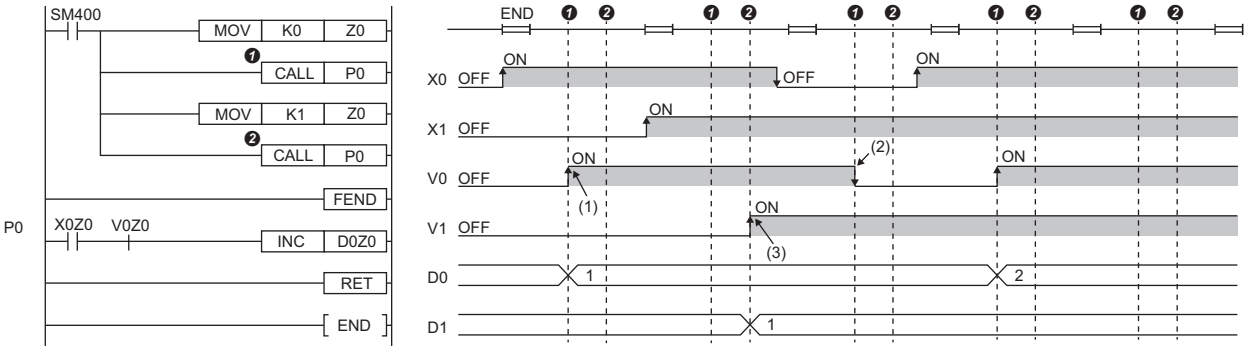
■Applicable devices

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□\□, U3E□\(H)G□	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 X0.
(2) The device turns off at the falling edge of X0.
(3) The device turns on at the rising edge of X1.

■EGF

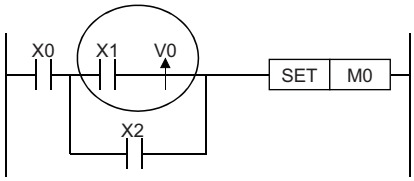
- This instruction stores the operation result up to the EGF instruction in the edge relay (V).
- The instruction turns on (continuity state) at the falling edge (on to off) of the operation result up to the EGF instruction. The instruction turns off (non-continuity state) when the operation result is 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 150 Operation start, series connection, parallel connection). The EGP or EGF instruction cannot be used at the LD and OR (☞ Page 150 Operation start, series connection, parallel connection) positions.
- The instructions cannot be used at the ladder block position shown below.



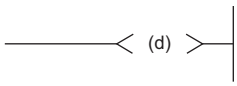
5.3 Output Instructions

Out (excluding the timer, counter, and annunciator)

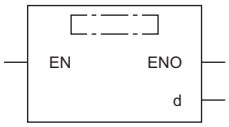
OUT

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	○	○*2	○	—	—	—	○	—	—	—	○

*1 When F is used, refer to the following.
☞ Page 178 OUT F

*2 When T or ST is used, refer to the following.
☞ Page 168 OUT T, OUTH T, OUT ST, OUTH ST
When C is used, refer to the following.
☞ Page 174 OUT C

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	○*3	—

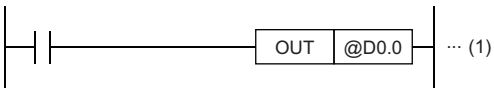
*3 When SA\T and SA\ST is used, refer to the following.
☞ Page 168 OUT T, OUTH T, OUT ST, OUTH ST
When SA\C is used, refer to the following.
☞ Page 174 OUT C

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.

Timer

OUT T, OUTH T, OUT ST, OUTH ST

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation when the systems are switched. (MELSEC iQ-R CPU Module User's Manual (Application))

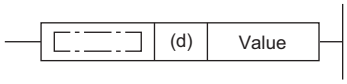
Point

When the safety timer is used in safety programs executed by the SIL2 Process CPU and the Safety CPU, unless otherwise specified, replace some words as follows:

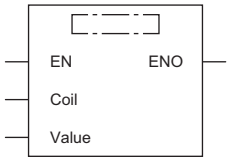
- "Timer" → "Safety timer"
- "Retentive timer" → "Safety retentive timer"
- "T" → "SA\T", "ST" → "SA\ST", "M0" → "SA\M0"
- "Scan" → "Safety cycle processing"
- "Scan time" → "Safety cycle time"

- 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) Coil	Timer device or timer type label	—	Bit	ANY_BOOL
Value	Value set for the timer	0 to 32767	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.

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d) Coil	—	—	○*1	—	—	—	—	—	—	—	—	—
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.

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d) Coil	—	○*4	—
Value	—	○*5	○*6

*4 Only SA\T and SA\ST can be used.
*5 SA\T, SA\ST, and SA\C cannot be used.
*6 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
Low-speed timer High-speed timer	Off	0	Non-continuity	Continuity	Non-continuity	Continuity
Low-speed retentive timer High-speed retentive timer	Off	Current value retained	Non-continuity	Continuity	Continuity	Non-continuity

- 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 1000ms (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.

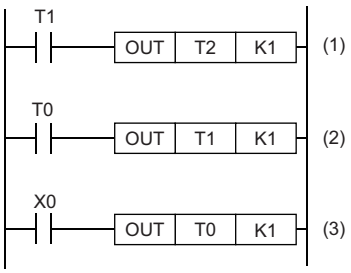
MELSEC iQ-R CPU Module User's Manual (Application)

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.
In the following cases, if a program is created in order of timer measurements, all timers turn on in the same scan.

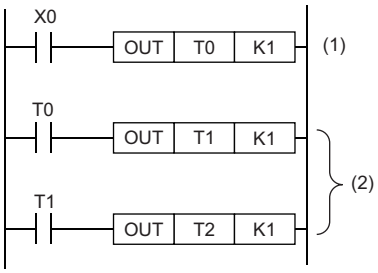
- The set value is smaller than the scan time.
- The set value is 1.

Ex.
When timers T0 to T2 are programmed in order from the one that measures last



(1) Timer T2 starts measurement from the next scan after the contact of timer T1 turns on.
(2) Timer T1 starts measurement from the next scan after the contact of timer T0 turns on.
(3) Timer T0 starts measurement when X0 turns on.

Ex.
When timers T0 to T2 are programmed in order of measurement



(1) Timer T0 starts measurement when X0 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.

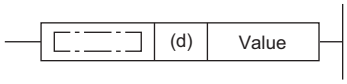
Long timer

OUT LT, OUT LST

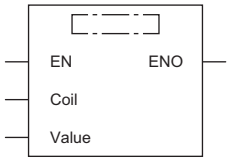
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, they do not operate in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- 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) Coil	Long timer device or long timer type label	—	Bit	ANY_BOOL
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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d) Coil	—	—	—	—	—	○*1	—	—	—	—	—	—
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.

MELSEC iQ-R CPU Module User's Manual (Application)

Precautions

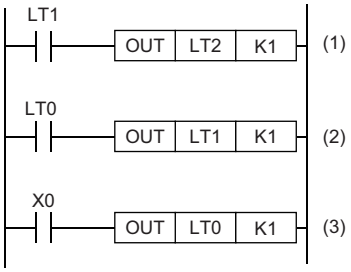
To create a program in which the operation of a long timer contact triggers the operation of another long timer, program the long timers in order from the one that operates last.

In the following cases, 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.

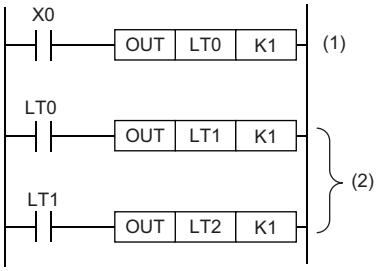
When timers LT0 to LT2 are programmed in order from the one that measures last



(1) Long timer LT2 starts measurement from the next scan after the contact of long timer LT1 turns on.
(2) Long timer LT1 starts measurement from the next scan after the contact of long timer LT0 turns on.
(3) Long timer LT0 starts measurement when X0 turns on.

Ex.

When long timers LT0 to LT2 are programmed in order of measurement



(1) Long timer LT0 starts measurement when X0 turns on.

(2) When the contact of timer LT0 turns on, the contacts of timers LT1 and LT2 also turn on.

Operation error

There is no operation error.

Counter

OUT C

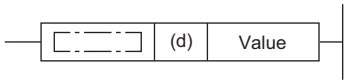
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

Point

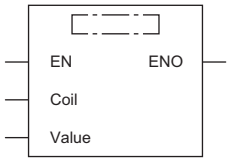
When the safety counter is used in safety programs executed by the SIL2 Process CPU and the Safety CPU, unless otherwise specified, replace a word as follows:

- "OUT C" → "OUT SA\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.

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d) Coil	—	—	○*1	—	—	—	—	—	—	—	—	—
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.

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d) Coil	—	○*4	—
Value	—	○*5	○*6

*4 Only SA\C can be used.
*5 SA\T, SA\ST, and SA\C cannot be used.
*6 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 SD 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).
- 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.
- When the set value is 0, the same processing is performed as when it is set to 1.

Operation error

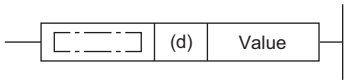
There is no operation error.

Long counter

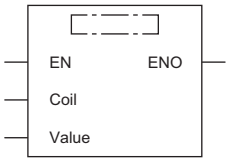
OUT LC

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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 SD 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).
- 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.
- When the set value is 0, the same processing is performed as when it is set to 1.

Operation error

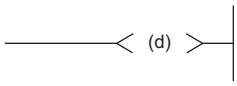
There is no operation error.

Annunciator

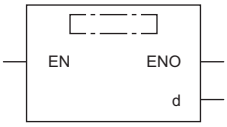
OUT F

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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 CPU module 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

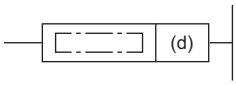
There is no operation error.

Setting devices (excluding annunciator)

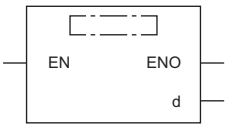
SET

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	○	—	—	—	—	—	○	—	—	—	○

*1 When F is used, refer to the following.
☞ Page 183 SET F

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

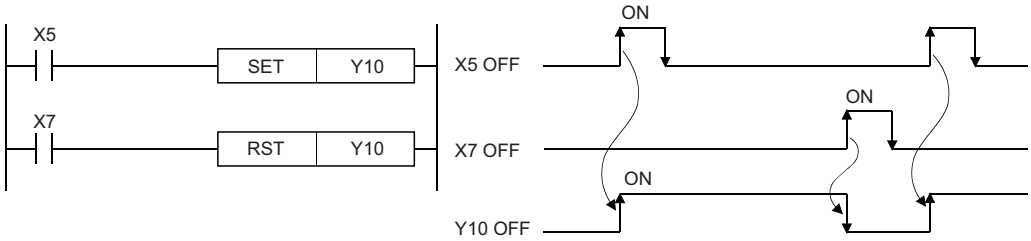
Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	—	—

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

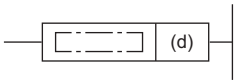
When X is used, specify a device number that is not used in actual input. If the number that is used in actual input is specified, the data of actual input is written over the input device (X) specified by the SET instruction.

Resetting devices (excluding annunciator)

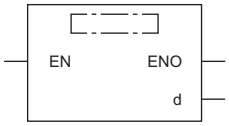
RST

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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_BOOL ANY16_U ANY16_S
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	○	○	○	○	○	○	○	—	—	—	○

*1 When F is used, refer to the following.
☞ Page 185 RST F

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

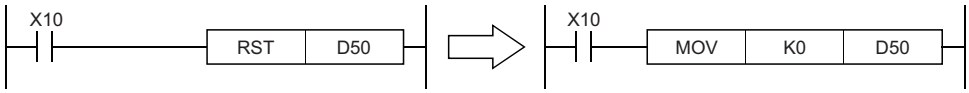
Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	○	—

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

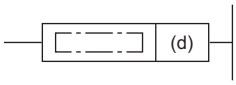
There is no operation error.

Setting annunciator

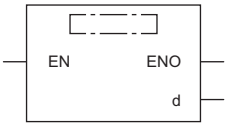
SET F

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

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

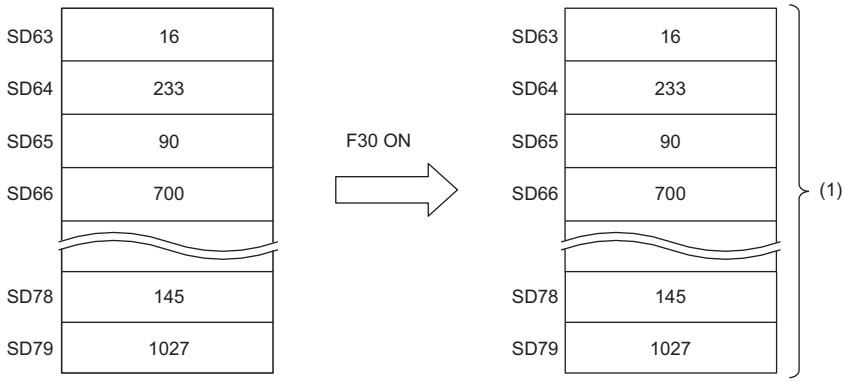
■Applicable devices

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□\□, U3E□\(H)G□	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

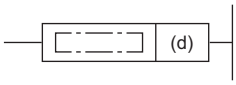
There is no operation error.

Resetting annunciator

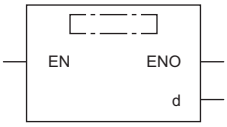
RST F

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	—	—	—	—	—	—	—	—	—	—	—

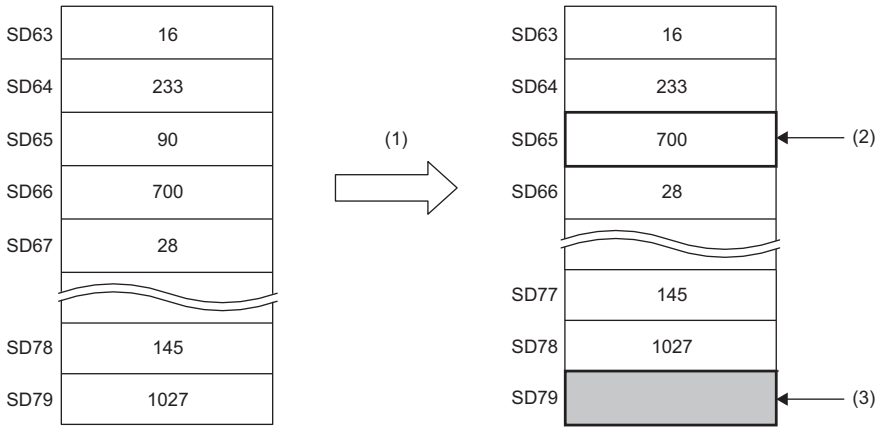
*1 Only F can be used.

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 CPU module 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

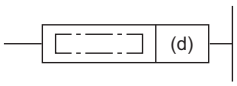
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

Point

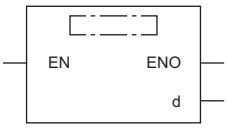
When this instruction is used in safety programs executed by the SIL2 Process CPU and the Safety CPU, unless otherwise specified, replace some words as follows:

- "X0" → "SA\X0", "X5" → "SA\X5", "M0" → "SA\M0"
- "Scan" → "Safety cycle processing"

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	—	—	—	○	—	—	—	○

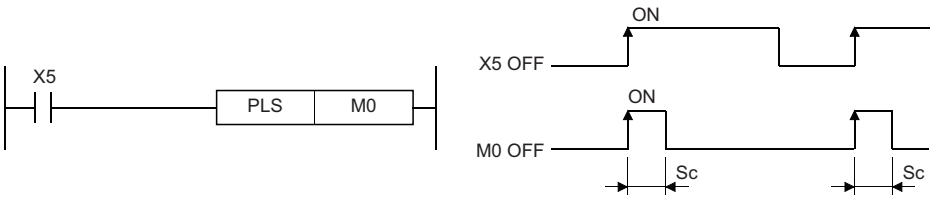
- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	○	—

Processing details

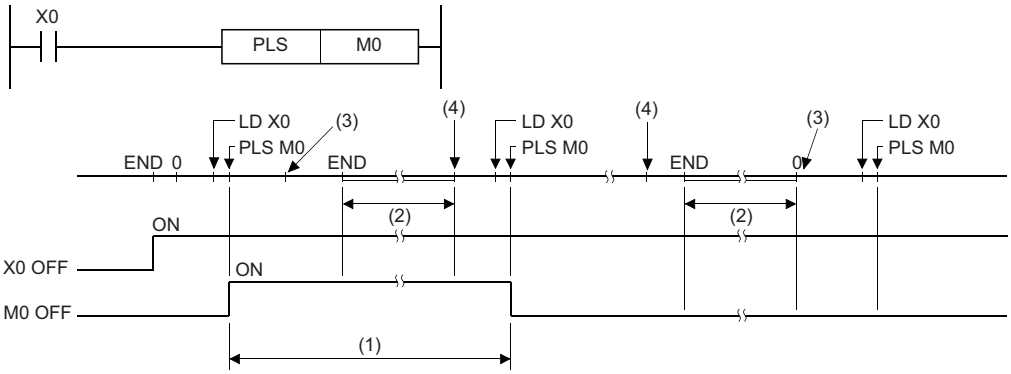
- This instruction turns on the specified device on the rising edge (off to on) of the execution command. When the execution command is in another state (staying on, falling edge (on to off), or staying off), the instruction turns off the specified device. If only one PLS instruction in the device specified 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 58 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 CPU module 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) The CPU module operation stops.
(3) Change the RUN/STOP/RESET switch of the CPU module from RUN to STOP.
(4) Change the RUN/STOP/RESET switch of the CPU module from STOP to 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. However, in safety programs executed by the SIL2 Process CPU and the Safety CPU, the PLS instruction turns on one safety cycle processing for the specified device/label until the safety program of next safety cycle processing starts and the instruction is executed. If the standard/safety shared label is used in the argument of the PLS instruction, the instruction may fail to detect the ON state of the standard/safety shared label or may detect the ON state for plural periods in the safety program or standard program that uses the corresponding standard/safety shared label depending on the timing to interrupt the safety cycle processing.

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

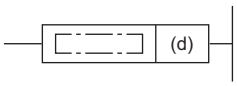
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

Point

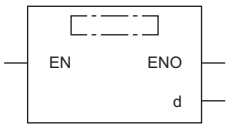
When this instruction is used in safety programs executed by the SIL2 Process CPU and the Safety CPU, unless otherwise specified, replace some words as follows:

- "X5" → "SA\X5"
- "Scan" → "Safety cycle processing"

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	—	—	—	○	—	—	—	○

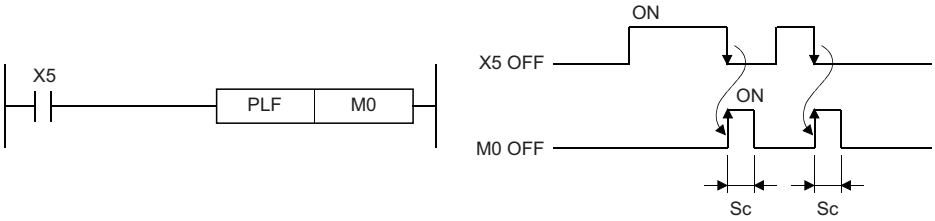
- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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 58 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 CPU module 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. However, in safety programs executed by the SIL2 Process CPU and the Safety CPU, the PLF instruction turns on one safety cycle processing for the specified device/label until the safety program of next safety cycle processing starts and the instruction is executed. If the standard/safety shared label is used in the argument of the PLF instruction, the instruction may fail to detect the ON state of the standard/safety shared label or may detect the ON state for plural periods in the safety program or standard program that uses the corresponding standard/safety shared label depending on the timing to interrupt the safety cycle processing.

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

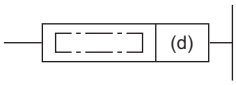
There is no operation error.

Inverting the bit device output

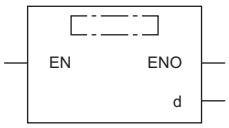
FF

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	—	—	—	○	—	—	—	○

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

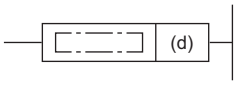
There is no operation error.

Converting the direct access output into a pulse

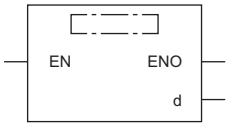
DELTA(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

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.

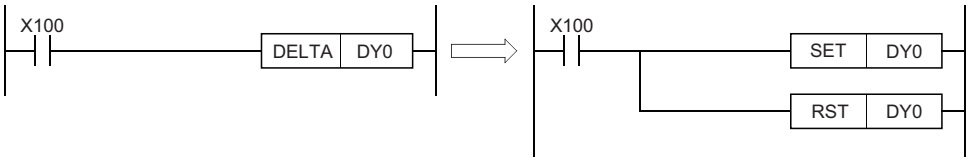
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	—	—	—	—	—	—	—	—	—	○

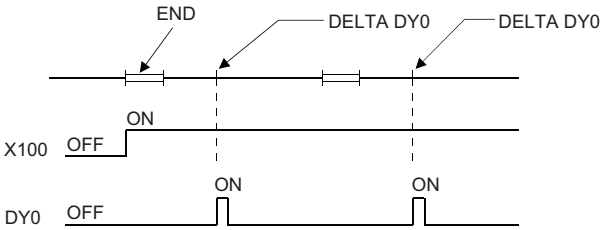
Processing details

- 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

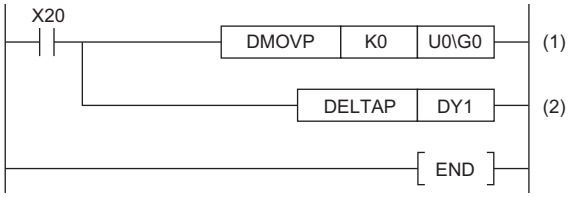
These instructions are used as an execution command (rising edge execution) for intelligent function modules.
These instructions cannot be used as an actual output command for output modules.

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 in slot 0 of the base unit when X20 turns on



(1) Store the preset value (0) in the buffer memory areas 0 and 1 of the RD62P2.
(2) Output the preset command.

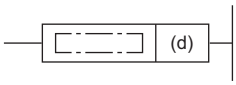
5.4 Shift Instructions

Shifting bit devices

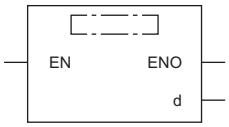
SFT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	—	—	—	○	—	—	—	○

Processing details

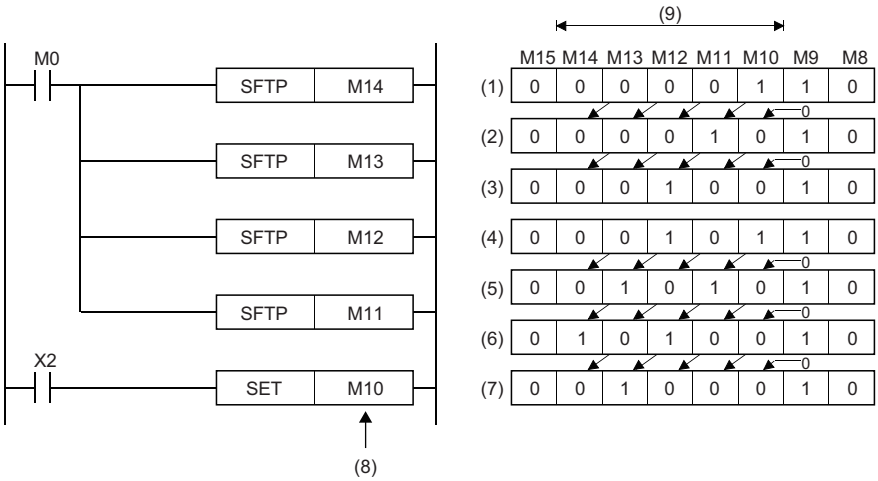
■Bit device

- 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.

Ex.

When the SFTP instruction that specifies M11 is executed, it shifts the on/off state of M10 to M11, and turns off M10.

- Turn on the shift target start device by using the SET instruction.
- When the SFT(P) instruction is used consecutively, program devices in descending order of the device numbers.



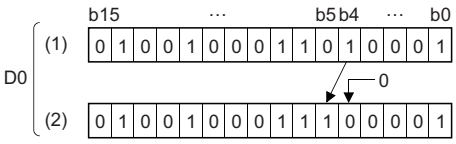
(1) X02 ON
(2) After the 1st shift input
(3) After the 2nd shift input
(4) X02 ON
(5) After the 3rd shift input
(6) After the 4th shift input
(7) After the 5th shift input
(8) Shift target start device
(9) Shift range

■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.

5.5 Master Control Instructions

Setting/resetting a master control

MC, MCR

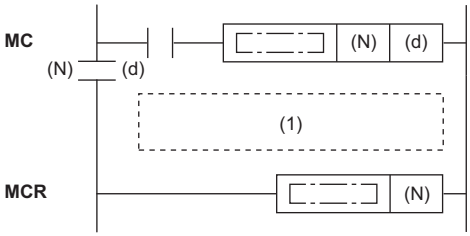
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

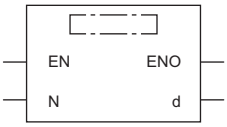
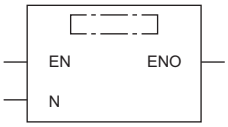
Point

When these instructions are used in safety programs executed by the SIL2 Process CPU and the Safety CPU, unless otherwise specified, replace some words as follows:

- "Timer" → "Safety timer"
- "Retentive timer" → "Safety retentive timer"
- "Counter" → "Safety counter"
- Add "SA\" to the devices "X" and "M" in Figures. (Example: "X0" → "SA\X0", "M0" → "SA\M0")
- "Scan time" → "Safety cycle time"

- MC: This instruction starts a master control.
- MCR: This instruction ends a master control.

Ladder	ST
 (1) Master control ladder	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.

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	N	DY
(N)	—	—	—	—	—	—	—	—	—	—	—	○	—
(d)	○	○	○	—	—	—	—	○	—	—	—	—	○

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

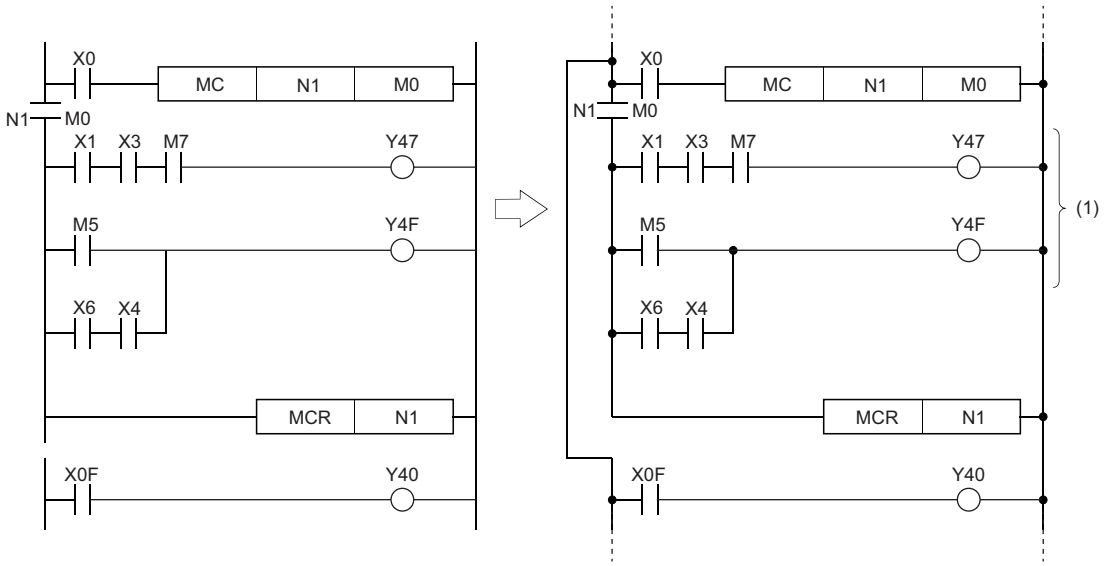
Operand	Bit	Word	Indirect specification	Constant	Others
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD		K, H	N
(N)	—	—	—	—	○
(d)	○	○	○	—	—

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 X0 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 CPU module 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

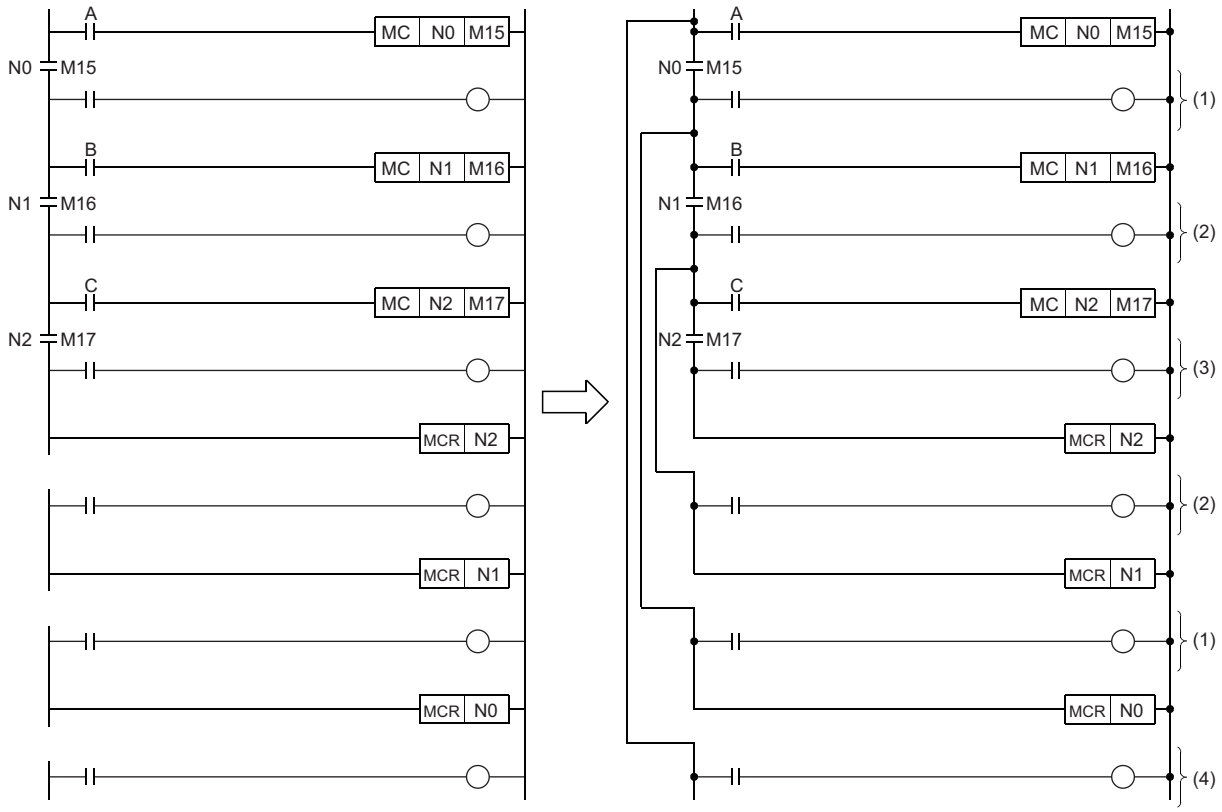
- This instruction is a master control reset instruction which indicates the end of the master control area.
- Do not place any contact instruction before the MCR instruction.
- Use the MC and MCR instructions with the same nesting number as a set. Note that if the MCR instructions are nested in one 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

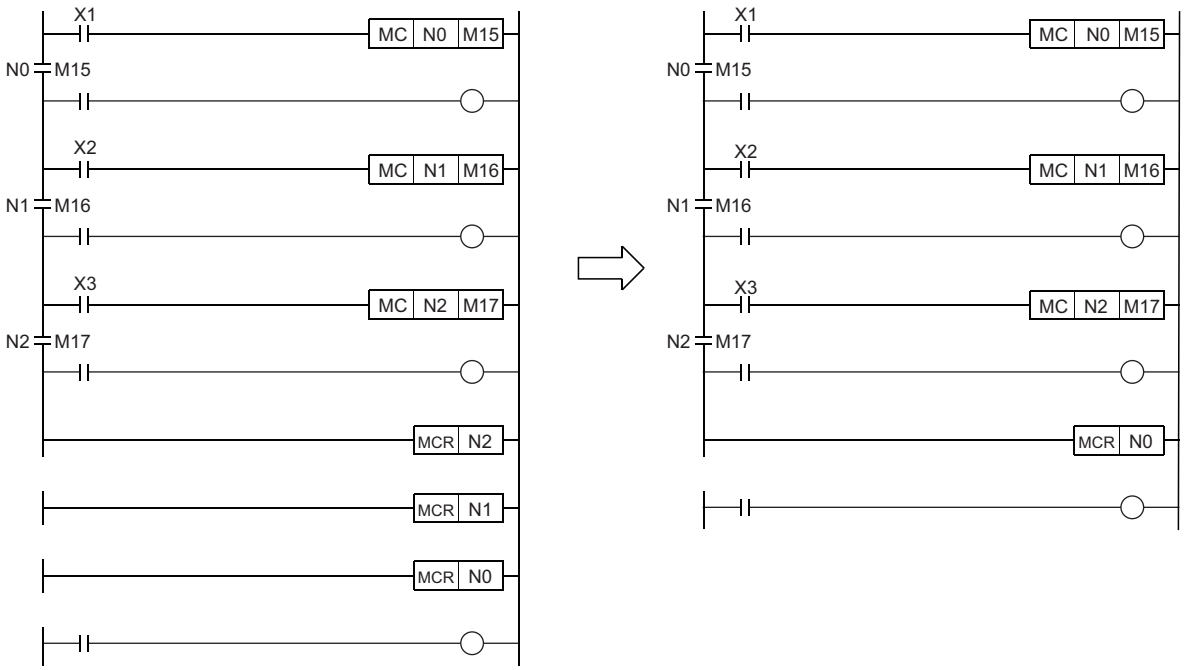
The master control instructions can be nested. Individual master control areas are distinguished by nesting (N) numbers. Nesting can be set from N0 to N14.
Using the nesting structure enables the creation of a ladder which can sequentially constrain the program execution conditions.
The following figure 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

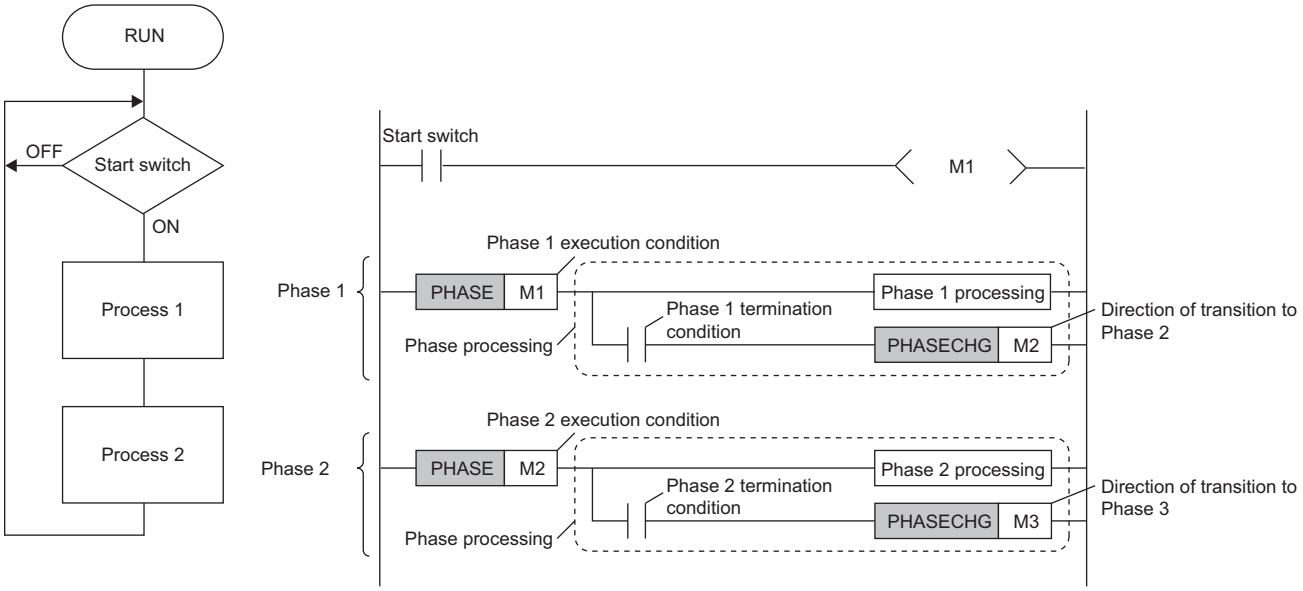
- Up to 15 nests (N0 to N14) are allowed. When nesting is performed, the MC instruction should use nesting (N) numbers in order from lower numbers and the MCR instruction should use them in order from higher numbers.
- If the MCR instructions are nested in one place, all master controls can be terminated by specifying the lowest nesting (N) number.



5.6 Phase Processing Instructions

Overview

Phase processing instructions are used when sequential processing is performed according to the process (phase) in the ladder program.
A program is executed for each process.



Point

- An example of a program expression that performs sequential processing according to the process is an SFC program. Since SFC programs allow a series of control operations to be divided into multiple steps with step transitions, and the execution order and execution conditions of the programs can be clearly expressed, the use of SFC programs is recommended when controlling complex processes.
- Since the PHASE instruction adds two steps, when using a function*1 that specifies the step number in the ladder program after the PHASE instruction, it is necessary to specify a step number that takes into account the two steps that are added (for a total of 15 steps).

*1 Functions that specify the step number include the following.
· Realtime monitor function
· Device test function with execution conditions
· Data logging function

Engineering tool setting

To use phase processing instructions, "Use Phase Processing Instructions" must be set to "Yes" in the engineering tool options. An error will occur if a program for which phase processing instructions are input is converted without this setting.

[Tool] ⇨ [Options] ⇨ [Convert] ⇨ [Basic Setting] ⇨ [Operational Setting]

Precautions

The following precautions apply in common to phase processing instructions.

- They can be used only in ladder programs. They cannot be used in SFC programs, ST programs, and FBD/LD.
- They can be used only in scan execution type programs. They cannot be used in initial execution type programs, fixed scan execution type programs, event execution type programs, and standby type programs. Note that a program for which phase processing instructions are input should not be changed to any other program execution type except the scan execution type program after it is converted by the engineering tool.
- They can be used only in main routine programs. They cannot be used in subroutine programs and interrupt programs.
- They can be used only within program blocks. They cannot be used in a function (FUN) or function block (FB).

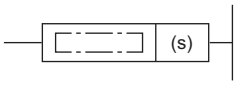
Starting the phase processing

PHASE

RnCPU RnENCPU ~~RnPCPU (Process)~~ ~~RnPCPU (Redundant)~~ ~~RnPSFCPU (Standard)~~ ~~RnPSFCPU (Safety)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "24" or later, and RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "57" or later support this instruction. Use an engineering tool with version "1.075D" or later.

This instruction starts phase processing.

Ladder	ST
	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
PHASE	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device for starting the phase processing	—	Bit	ANY_BOOL*1

*1 For array labels, labels or devices cannot be used for the array index.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(s)	○*1*3	—	○*2*3	—	—	—	—	—	—	—	—	—

*1 F cannot be used.
*2 T, ST, C, R, and ZR cannot be used.
*3 Index modification is not available.

Processing details

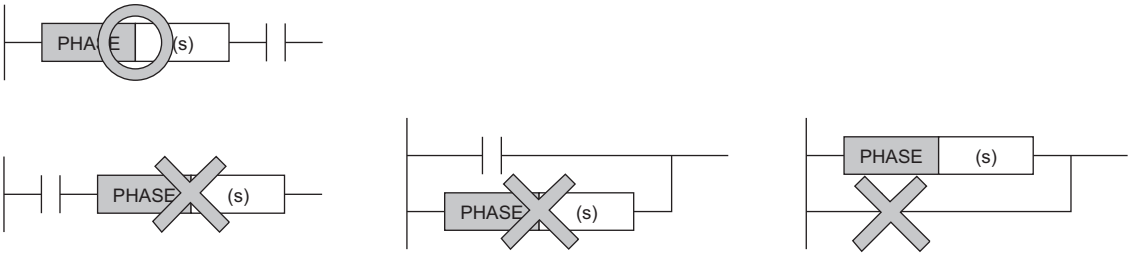
- If the device specified by (s) is turned off and on or stayed on, the operation result of the connected phase processing will be the one as programmed.
- If the device specified by (s) is turned on and off, the operation result of the connected phase processing will be as follows.

Device used for operation	Operation result (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	All turned off.
Device used by the SET/RST instructions Device used by the SFT instruction Device used by the basic/application instruction	Maintains the current status.

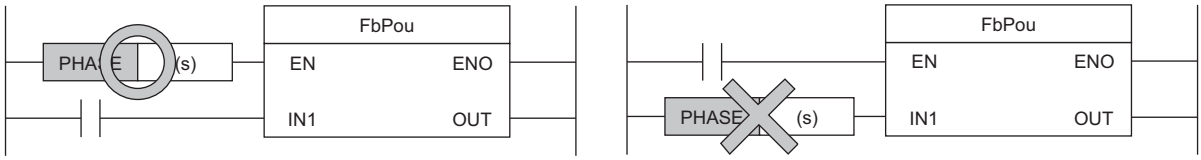
- If the device specified by (s) is stayed off, the connected phase processing is not performed.

Precautions

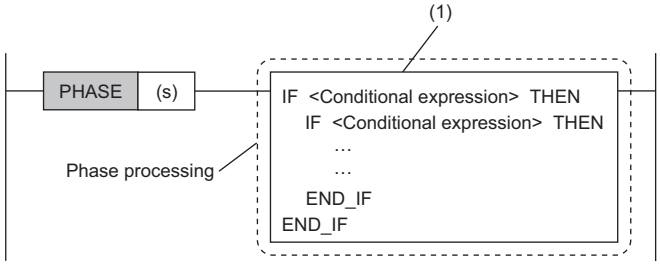
- The PHASE instruction can only be used for direct connection to a rail. Also, connection in parallel with other instructions is not possible.



- When connecting FB/FUN to the PHASE instruction, FB/FUN can be connected only to EN.



- When using control statements (IF statement, CASE statement, FOR statement, WHILE statement, REPEAT statement) in inline ST connected to the PHASE instruction, the number of levels must be within 127 levels.



(1) Within phase processing, up to 127 levels are supported for inline ST control statements.

Operation error

There is no operation error.

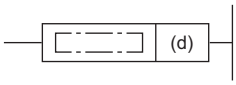
Changing the execution phase

PHASECHG

RnCPU RnENCPU ~~RnPCPU (Process)~~ ~~RnPCPU (Redundant)~~ ~~RnPSFCPU (Standard)~~ ~~RnPSFCPU (Safety)~~ ~~RnSFCPU (Standard)~~ ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "24" or later, and RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "57" or later support this instruction. Use an engineering tool with version "1.075D" or later.

This instruction ends the phase currently being executed and shifts to the specified phase.

Ladder	ST
	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
PHASECHG	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Transition destination phase device	—	Bit	ANY_BOOL*1

*1 For array labels, labels or devices cannot be used for the array index.

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(d)	○*1*3	—	○*2*3	—	—	—	—	—	—	—	—	—

*1 F cannot be used.
*2 T, ST, C, R, and ZR cannot be used.
*3 Index modification is not available.

Processing details

- This instruction ends the phase currently being executed (turns off the device specified in the previous PHASE instruction*1) and shifts to the phase specified in (d) (turns on the specified device).

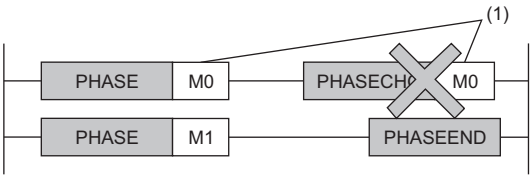
*1 The non-execution processing of the phase processing connected to the PHASE instruction is performed at the next scan after the execution of this instruction.

Precautions

- Execute the PHASECHG instruction within the phase processing. If it is executed outside the phase processing, the specified device of the PHASE instruction that was last turned on is turned off[*1]. (Phase transition (turning on the specified device) is performed.)

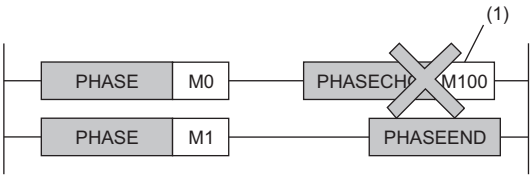
*1 This excludes cases where ladder block change during RUN between the ON execution of the PHASE instruction and the execution of the PHASECHG instruction, or writing to the running sequencer (writing any of sequence program, FB file, or global label setting file), is executed.

- Do not specify in (d) the same device as the device specified in the previous PHASE instruction. Otherwise, the phase currently being executed will not end and the phase will not shift.



(1) Specify the same device in the PHASECHG as that specified in the PHASE instruction.

- Do not specify in (d) a device that is not used in the PHASE instruction. Otherwise, the phase currently being executed will end but the phase will not shift. (The device specified is turned on.)



(1) Specify a device that is not used in the PHASE instruction.

- 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 PHASECHG instruction.

Operation error

There is no operation error.

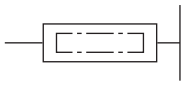
Terminating the execution phase

PHASEEND

RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "24" or later, and RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "57" or later support this instruction. Use an engineering tool with version "1.075D" or later.

This instruction ends the phase currently being executed.

Ladder	ST
	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
PHASEEND	

Processing details

- This instruction ends the phase currently being executed (turns off the device specified in the previous PHASE instruction[*1]).

*1 The non-execution processing of the phase processing connected to the PHASE instruction is performed at the next scan after the execution of this instruction.

Precautions

- Execute the PHASEEND instruction within the phase processing. If it is executed outside the phase processing, the specified device of the PHASE instruction that was last turned on is turned off[*1].

*1 This excludes cases where ladder block change during RUN between the ON execution of the PHASE instruction and the execution of the PHASEEND instruction, or writing to the running sequencer (writing any of sequence program, FB file, or global label setting file), is executed.

Operation error

There is no operation error.

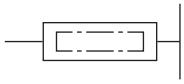
5.7 Termination Instructions

Ending the main routine program

FEND

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

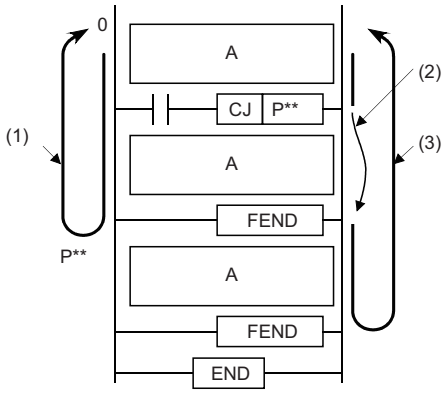
■Execution condition

Instruction	Execution condition
FEND	Every scan

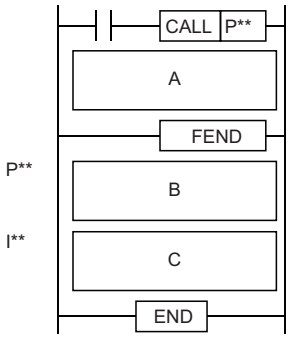
Processing details

- 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 CPU module 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

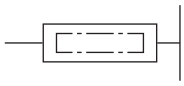
Error code (SD0)	Description
3340H	After execution of the FOR instruction, the FEND instruction is executed before the NEXT instruction.
3381H	After execution of the CALL(P), FCALL(P), ECALL(P), or EFCALL(P) instruction, the FEND instruction is executed before the RET instruction.
33A1H	Within the interrupt program specified by the interrupt pointer (I), the FEND instruction is executed before the IRET instruction.

Ending the sequence program

END

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction indicates the end of a program.

Ladder	ST
	Not supported

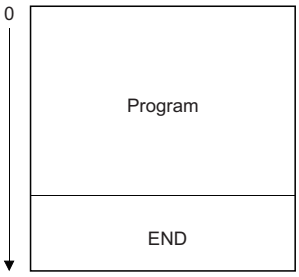
FBD/LD
Not supported

■Execution condition

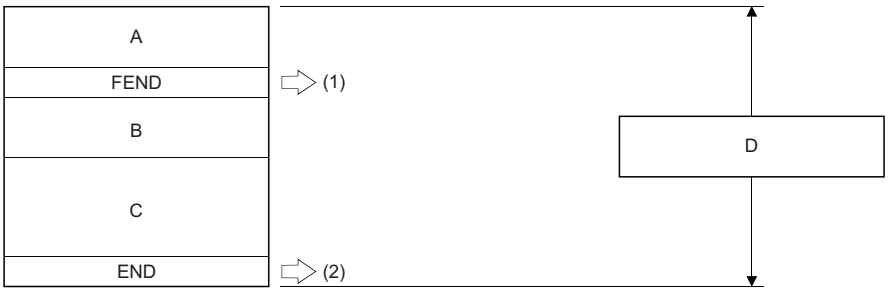
Instruction	Execution condition
END	Every scan

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 CPU module 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
3340H	After execution of the FOR instruction, the END instruction is executed before the NEXT instruction.
3381H	After execution of the CALL(P), FCALL(P), ECALL(P), or EFCALL(P) instruction, the END instruction is executed before the RET instruction.
33A1H	Within the interrupt program specified by the interrupt pointer (I), the END instruction is executed before the IRET instruction.

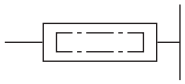
5.8 Stop Instruction

Stopping the sequence program

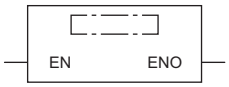
STOP

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This instruction stops the operation of the CPU module. (The operation of this instruction is the same as setting the switch of the CPU module 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 CPU module when the execution command turns on. (The operation of this instruction is the same as setting the switch of the CPU module to the STOP position.)
- To restart the operation of the CPU module 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
3340H	After execution of the FOR instruction, the STOP instruction is executed before the NEXT instruction.
3381H	After execution of the CALL(P), FCALL(P), ECALL(P), EFCALL(P), or XCALL instruction, the STOP instruction is executed before the RET instruction.
33A1H	Within the interrupt program specified by the interrupt pointer (I), the STOP instruction is executed before the IRET instruction.
33A3H	The STOP instruction is executed within a fixed scan execution type program.
33A4H	The STOP instruction is executed within an event execution type program.

5.9 No Operation Instruction

No operation (NOP)

NOP

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction is used to insert a space for debugging.

Ladder	ST
—	Not supported

FBD/LD
Not supported

■Execution condition

Instruction	Execution condition
NOP	Every scan

Processing details

- This instruction is a no-operation instruction and has no impact on the previous operations.
- The instruction is used for the following purposes:
  - To insert a space for debugging
  - To delete an instruction without changing the number of steps (The relevant instruction is replaced with the NOP instruction.)
  - To delete an instruction temporarily

Point

For inserting or deleting the NOP instruction, refer to the following.

GX Works3 Operating Manual

Operation error

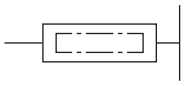
There is no operation error.

No operation (NOPLF)

NOPLF

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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.

PART 4 BASIC INSTRUCTIONS

This part consists of the following chapters.

6 BASIC INSTRUCTIONS

6 BASIC INSTRUCTIONS

6.1 Comparison Operation Instructions

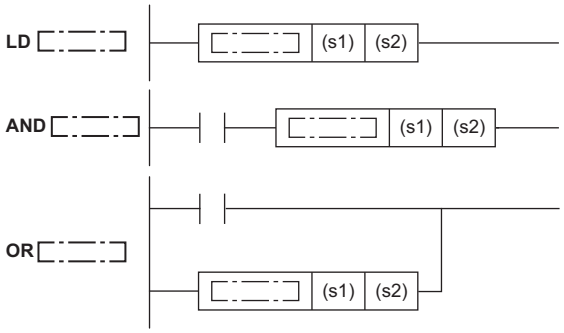
Comparing 16-bit binary data

LD□(_U), AND□(_U), OR□(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions compare the two sets of 16-bit binary data specified. (Devices are used as normally open contacts.)

Ladder



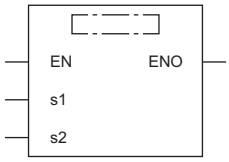
(□ is to be replaced by any of the following: =(_U), <>(_U), >(_U), <=(_U), <(_U), >=(_U).)

ST*1

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.)*2

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).)*2

*1 The engineering tool with version "1.035M" or later supports the ST.
*2 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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.

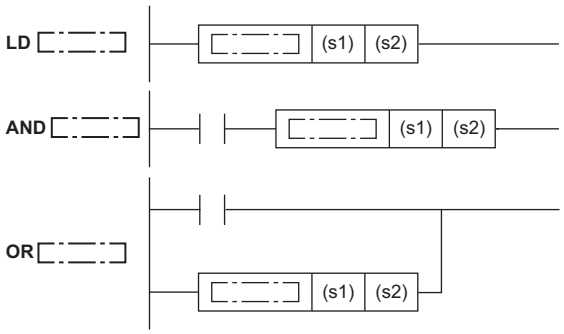
Comparing 32-bit binary data

LDD□(_U), ANDD□(_U), ORD□(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions compare the two sets of 32-bit binary data specified. (Devices are used as normally open contacts.)

Ladder



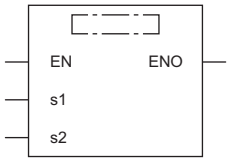
(□ is to be replaced by any of the following: D=(_U), D<>(_U), D>(_U), D<=(_U), D<(_U), D>=(_U).)

ST*1

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.)*2

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).)*2

*1 The engineering tool with version "1.035M" or later supports the ST.
*2 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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.

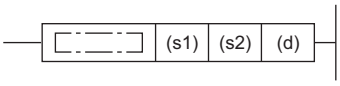
Outputting a comparison result of 16-bit binary data

CMP(P)(_U)

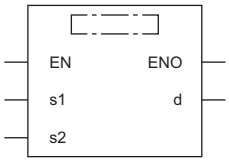
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions compare the 16-bit binary data specified by (s1) with the 16-bit binary data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST	
	ENO:=CMP(EN,s1,s2,d); ENO:=CMPP(EN,s1,s2,d);	ENO:=CMP_U(EN,s1,s2,d); ENO:=CMPP_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
CMP CMP_U	
CMPP CMPP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	CMP(P)	Comparison data or the device where the comparison data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	CMP(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	CMP(P)	Comparison data or the device where the comparison data is stored	-32768 to 32767	16-bit signed binary	ANY16_S
	CMP(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(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

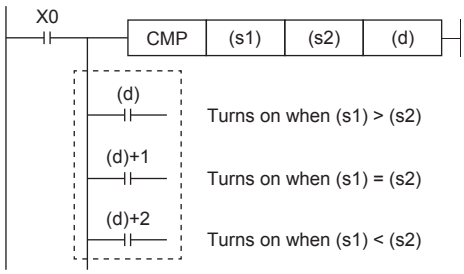
■Applicable devices

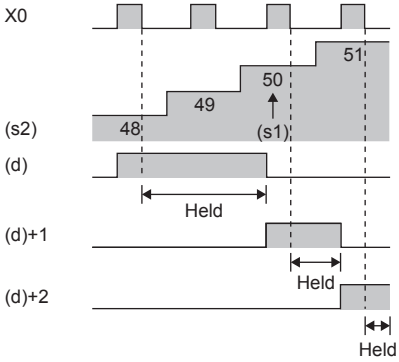
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(s2)	○*1	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	○*2	—	—	—	—	○	—	—	—	—

*1 FX and FY cannot be used.
*2 T, ST, and C cannot be used.

Processing details

- These instructions compare the 16-bit binary data specified by (s1) with the 16-bit binary data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.





Operation error

There is no operation error.

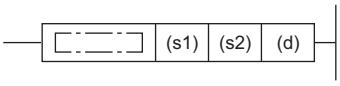
Outputting a comparison result of 32-bit binary data

DCMP(P)(_U)

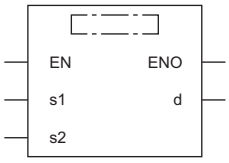
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions compare the 32-bit binary data specified by (s1) with the 32-bit binary data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST	
	ENO:=DCMP(EN,s1,s2,d); ENO:=DCMPP(EN,s1,s2,d);	ENO:=DCMP_U(EN,s1,s2,d); ENO:=DCMPP_U(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DCMP DCMP_U	
DCMPP DCMPP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DCMP(P)	Comparison data or the device where the comparison data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DCMP(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	DCMP(P)	Comparison data or the device where the comparison data is stored	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DCMP(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(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

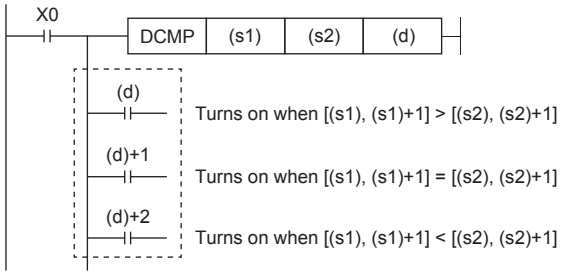
■Applicable devices

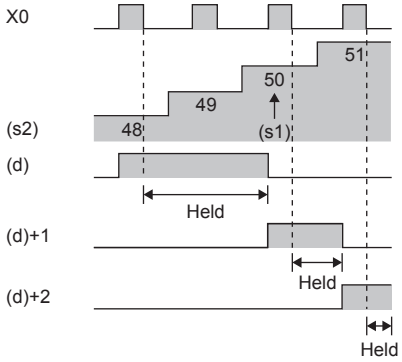
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○*1	○	○	○	○	○	○	○	○	—	—	—
(s2)	○*1	○	○	○	○	○	○	○	○	—	—	—
(d)	○	—	○*2	—	—	—	—	○	—	—	—	—

*1 FX and FY cannot be used.
*2 T, ST, and C cannot be used.

Processing details

- These instructions compare the 32-bit binary data specified by (s1) with the 32-bit binary data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.





Operation error

There is no operation error.

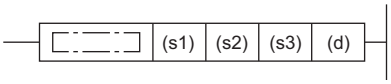
Outputting a band comparison result of 16-bit binary data

ZCP(P)(_U)

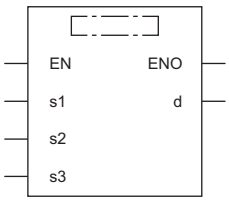
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions compare the band between the 16-bit binary data specified by lower limit value (s1) and the 16-bit binary data specified by upper limit value (s2) with the 16-bit binary data in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST	
	ENO:=ZCP(EN,s1,s2,s3,d); ENO:=ZCPP(EN,s1,s2,s3,d);	ENO:=ZCP_U(EN,s1,s2,s3,d); ENO:=ZCPP_U(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ZCP ZCP_U	
ZCPP ZCPP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	ZCP(P)	Lower limit value or the start device for storing the lower limit value	-32768 to 32767	16-bit signed binary	ANY16_S
	ZCP(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s2)	ZCP(P)	Upper limit value or the start device for storing the upper limit value	-32768 to 32767	16-bit signed binary	ANY16_S
	ZCP(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(s3)	ZCP(P)	Comparison data or the start device for storing the comparison data	-32768 to 32767	16-bit signed binary	ANY16_S
	ZCP(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(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

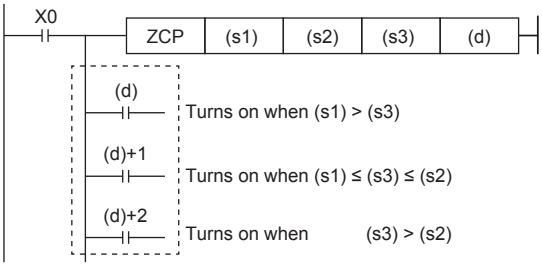
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(s2)	○*1	○	○	○	○	—	—	○	○	—	—	—
(s3)	○*1	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	○*2	—	—	—	—	○	—	—	—	—

*1 FX and FY cannot be used.
*2 T, ST, and C cannot be used.

Processing details

- These instructions compare the band between the 16-bit binary data specified by lower limit value (s1) and the 16-bit binary data specified by upper limit value (s2) with the 16-bit binary data in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.



Precautions

- Set (s1) to a value less than (s2). If (s1) is set to a value greater than (s2), (s2) is treated as the same value as (s1).

Operation error

There is no operation error.

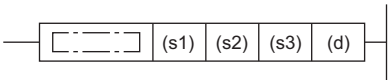
Outputting a band comparison result of 32-bit binary data

DZCP(P)(_U)

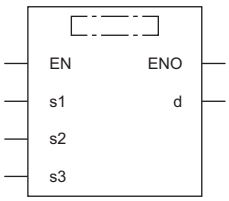
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions compare the band between the 32-bit binary data specified by lower limit value (s1) and the 32-bit binary data specified by upper limit value (s2) with the 32-bit binary data in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST	
	ENO:=DZCP(EN,s1,s2,s3,d); ENO:=DZCPP(EN,s1,s2,s3,d);	ENO:=DZCP_U(EN,s1,s2,s3,d); ENO:=DZCPP_U(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DZCP DZCP_U	
DZCPP DZCPP_U	

Setting data

■Description, range, data type

Operand		Description	Range	Data type	Data type (label)
(s1)	DZCP(P)	Lower limit value or the start device for storing the lower limit value	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DZCP(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s2)	DZCP(P)	Upper limit value or the start device for storing the upper limit value	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DZCP(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(s3)	DZCP(P)	Comparison data or the start device for storing the comparison data	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
	DZCP(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(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

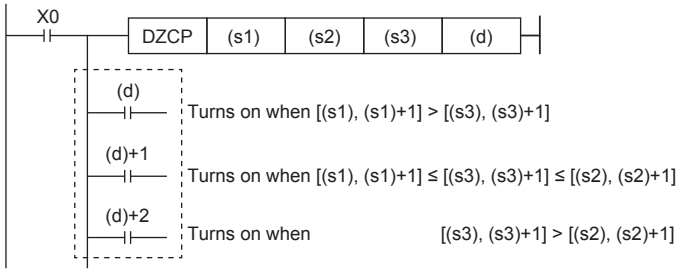
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○*1	○	○	○	○	○	○	○	○	—	—	—
(s2)	○*1	○	○	○	○	○	○	○	○	—	—	—
(s3)	○*1	○	○	○	○	○	○	○	○	—	—	—
(d)	○	—	○*2	—	—	—	—	○	—	—	—	—

*1 FX and FY cannot be used.
*2 T, ST, and C cannot be used.

Processing details

- These instructions compare the band between the 32-bit binary data specified by lower limit value (s1) and the 32-bit binary data specified by upper limit value (s2) with the 32-bit binary data in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.



Precautions

- Set (s1) to a value less than (s2). If (s1) is set to a value greater than (s2), (s2) is treated as the same value as (s1).

Operation error

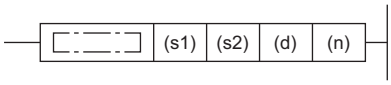
There is no operation error.

Comparing 16-bit binary block data

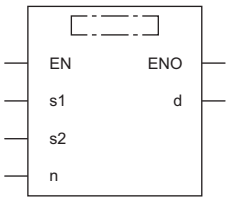
BKCMP□(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions compare the two sets of 16-bit binary block data specified.

Ladder	ST*1	
 (□ 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).)	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: EQ, NE, GT, LE, LT, GE.)*2	

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).)*2

*1 The engineering tool with version "1.035M" or later supports the ST.
*2 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

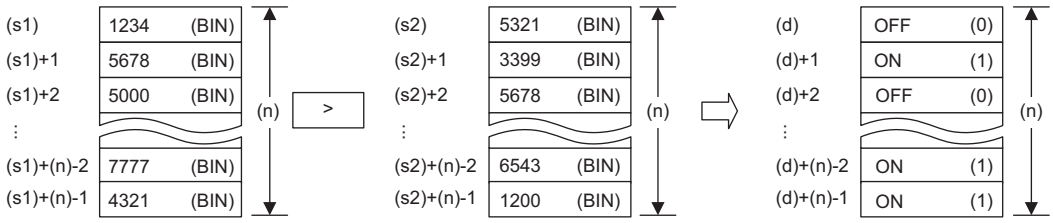
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

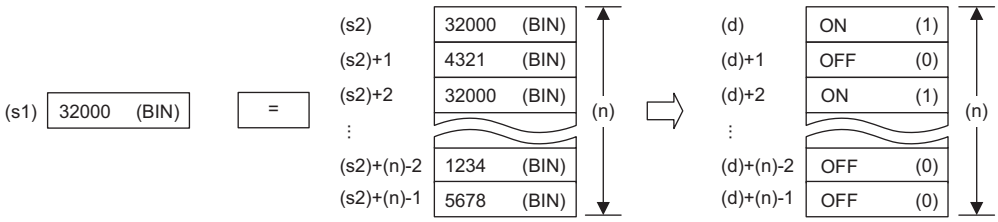
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

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

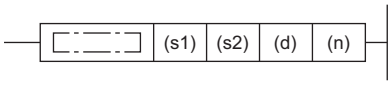
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping.
	The device ranges starting from the ones specified by (s2) and (d) are overlapping.

Comparing 32-bit binary block data

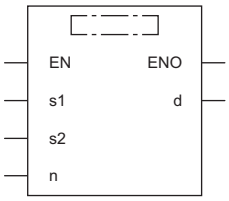
DBKCMP□(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions compare the two sets of 32-bit binary block data specified.

Ladder	ST*1	
 (□ 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).)	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: EQ, NE, GT, LE, LT, GE.)*2	

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).)*2

*1 The engineering tool with version "1.035M" or later supports the ST.
*2 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

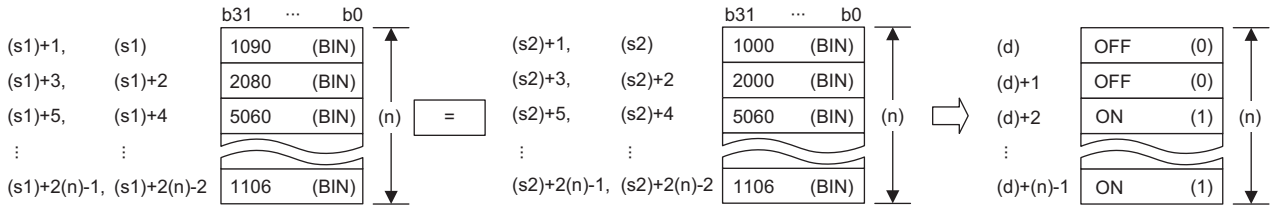
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

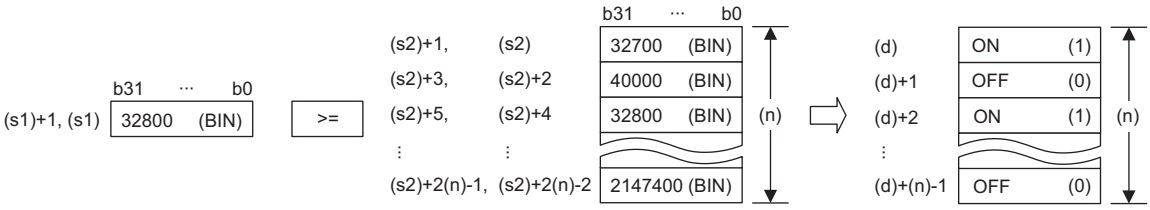
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

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)	

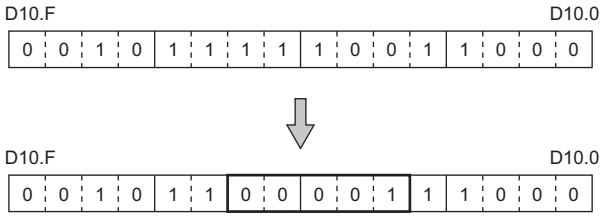
- When the comparison operation results stored in (n) points from the device specified by (d) are all on (1), SM704 turns on.
- If (n) is 0, no processing is performed.

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.



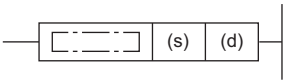
6.2 Arithmetic Operation Instructions

Adding 16-bit binary data

+(P)(_U) [when two operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions add the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 233 +(P)(_U) [when three operands are set])

FBD/LD
Not supported (☞ Page 233 +(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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

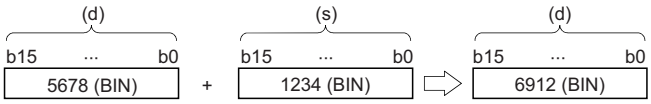
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

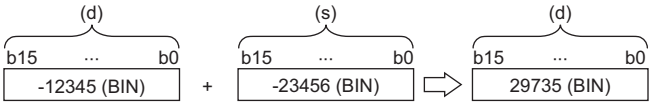
Processing details

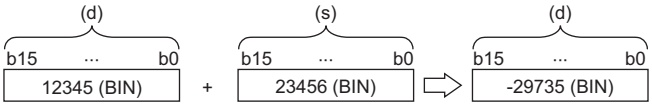
- These instructions add the 16-bit binary data in the device specified by (d) and the 16-bit binary data in the device specified by (s), and store the operation result in the device specified by (d).



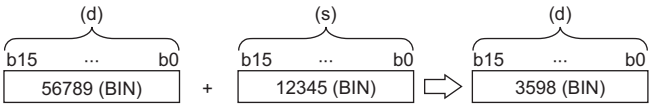
- If an overflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[+(P) instruction]





[+(P)_U instruction]



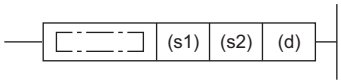
Operation error

There is no operation error.

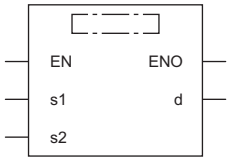
+(P)(_U) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

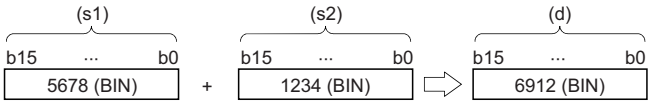
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

Processing details

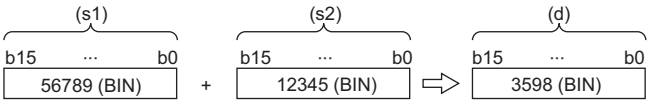
- These instructions add the 16-bit binary data in the device specified by (s1) and the 16-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



- 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

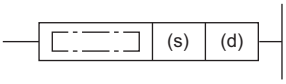
There is no operation error.

Subtracting 16-bit binary data

-(P)(_U) [when two operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform subtraction between the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 237 -(P)(_U) [when three operands are set])

FBD/LD
Not supported (☞ Page 237 -(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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

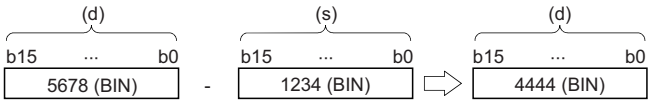
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

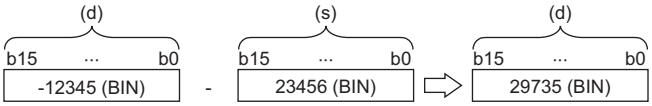
Processing details

- 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).

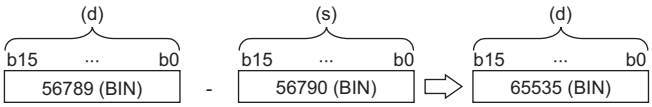


- If an underflow occurs in the result, the borrow bit is ignored. In this case, SM700 does not turn on.

[-(P) instruction]



[-(P)_U instruction]



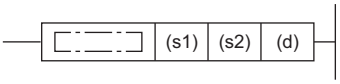
Operation error

There is no operation error.

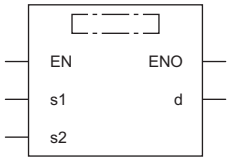
-(P)(_U) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

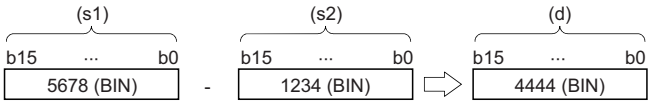
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

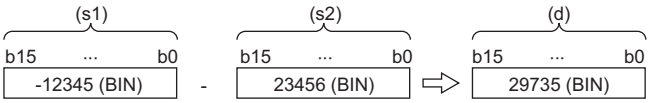
Processing details

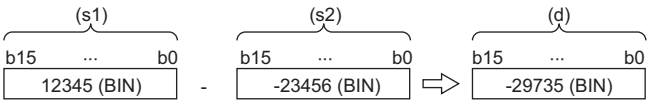
- These instructions subtract the 16-bit binary data in the device specified by (s2) from the 16-bit binary data in the device specified by (s1), and store the operation result in the device specified by (d).



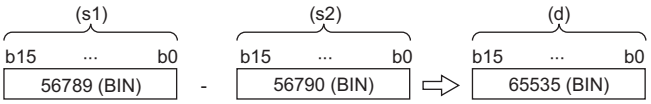
- If an underflow occurs in the result, the borrow bit is ignored. In this case, SM700 does not turn on.

[-(P) instruction]





[-(P)_U instruction]



Operation error

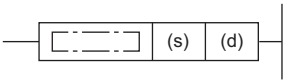
There is no operation error.

Adding 32-bit binary data

D+(P)(_U) [when two operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions add the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 241 D+(P)(_U) [when three operands are set])

FBD/LD
Not supported (☞ Page 241 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

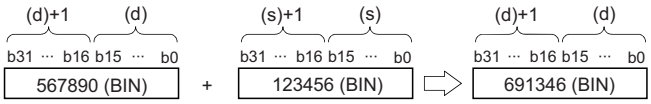
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

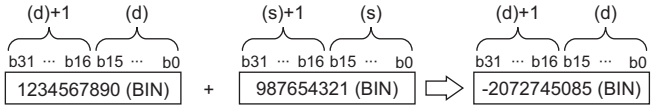
Processing details

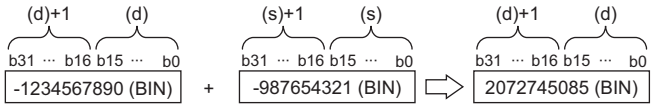
- These instructions add the 32-bit binary data in the device specified by (d) and the 32-bit binary data in the device specified by (s), and store the operation result in the device specified by (d).



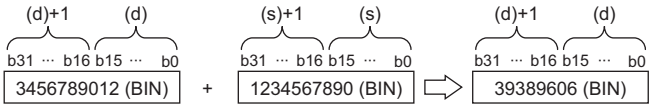
- If an overflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[D+(P) instruction]





[D+(P)_U instruction]



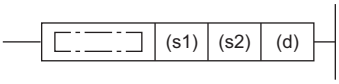
Operation error

There is no operation error.

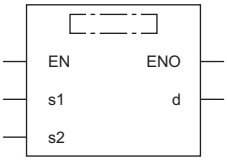
D+(P)(_U) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

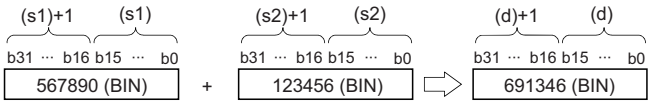
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

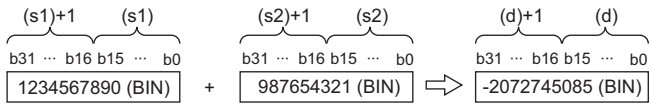
Processing details

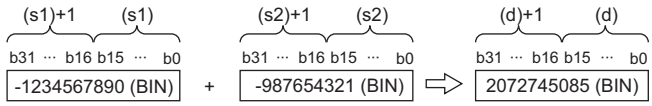
- These instructions add the 32-bit binary data in the device specified by (s1) and the 32-bit binary data in the device specified by (s2), and store the operation result in the device specified by (d).



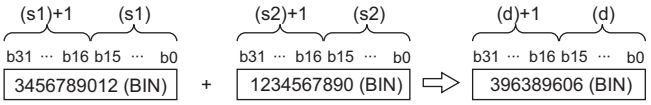
- If an overflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[D+(P) instruction]





[D+(P)_U instruction]



Operation error

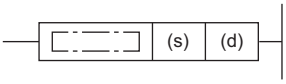
There is no operation error.

Subtracting 32-bit binary data

D-(P)(_U) [when two operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform subtraction between the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 245 D-(P)(_U) [when three operands are set])

FBD/LD
Not supported (☞ Page 245 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

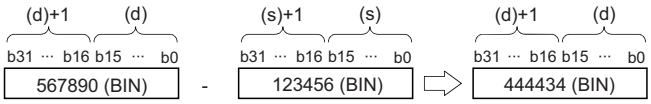
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	○
(d)	○	○	—

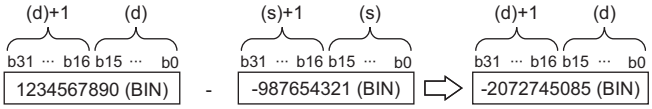
Processing details

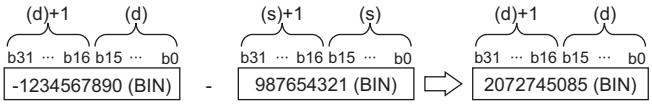
- These instructions subtract the 32-bit binary data in the device specified by (s) from the 32-bit binary data in the device specified by (d) and, and store the operation result in the device specified by (d).



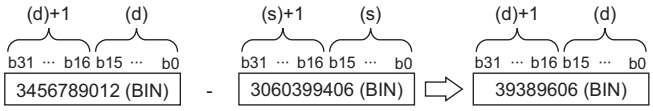
- If an underflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[D-(P) instruction]





[D-(P)_U instruction]



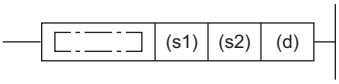
Operation error

There is no operation error.

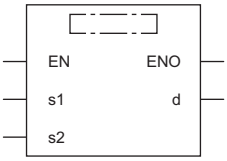
D-(P)(_U) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

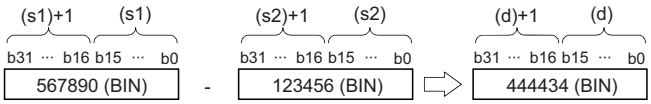
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s1)	○	○	○
(s2)	○	○	○
(d)	○	○	—

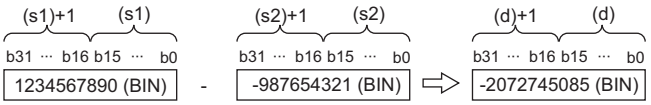
Processing details

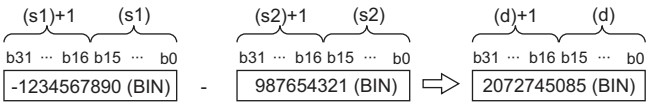
- These instructions subtracts the 32-bit binary data in the device specified by (s2) from the 32-bit binary data in the device specified by (s1), and store the operation result in the device specified by (d).



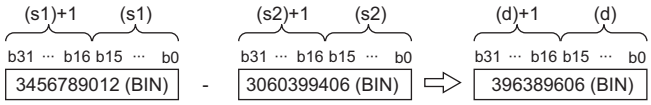
- If an underflow occurs in the result, the carry bit is ignored. In this case, SM700 does not turn on.

[D-(P) instruction]





[D-(P)_U instruction]



Operation error

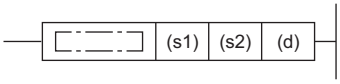
There is no operation error.

Multiplying 16-bit binary data

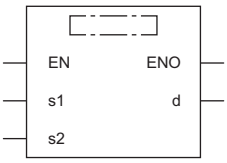
*(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions multiply the two sets of 16-bit binary data specified.

Ladder	ST*1	
	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.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

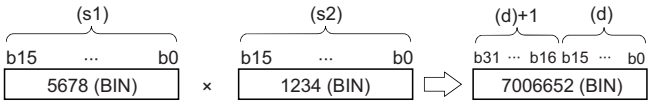
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

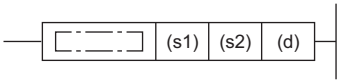
There is no operation error.

Dividing 16-bit binary data

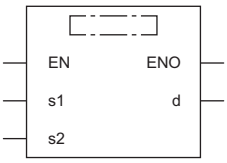
/(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform division between the two sets of 16-bit binary data specified.

Ladder	ST*1	
	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.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

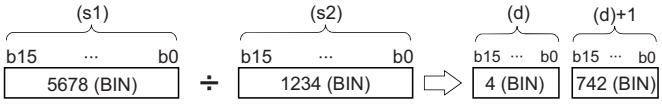
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

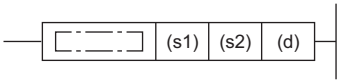
Error code (SD0)	Description
3400H	The value (divisor) in the device specified by (s2) is 0.

Multiplying 32-bit binary data

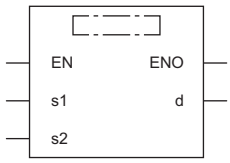
D*(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions multiply the two sets of 32-bit binary data specified.

Ladder	ST*1	
	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.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

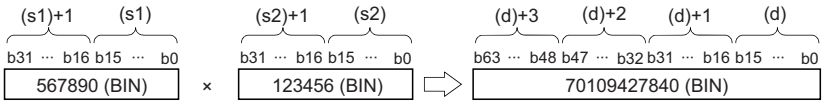
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	—	○	—	—	○	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

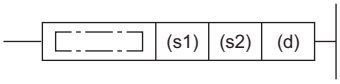
There is no operation error.

Dividing 32-bit binary data

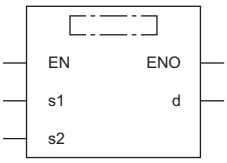
D/(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform division between the two sets of 32-bit binary data specified.

Ladder	ST*1	
	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.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

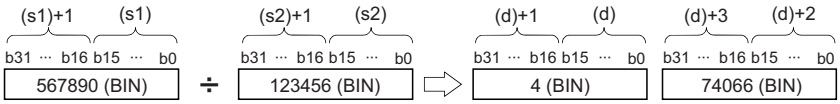
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	—	○	—	—	○	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

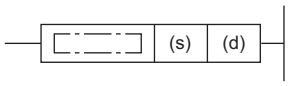
Error code (SD0)	Description
3400H	The value (divisor) in the device specified by (s2) is 0.

Adding BCD 4-digit data

B+(P) [when two operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions add the two sets of BCD 4-digit data specified.

Ladder	ST
	Not supported (☞ Page 256 B+(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 256 B+(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
B+	
B+P	

Setting data

■Description, range, data type

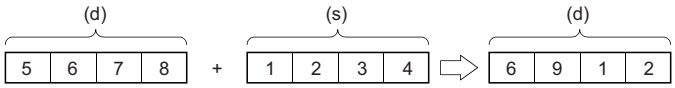
Operand	Description	Range	Data type	Data type (label)
(s)	Second addend data or the device where the second addend data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Device where the first addend data is stored	0 to 9999	BCD 4-digit	ANY16

■Applicable devices

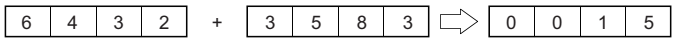
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

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.



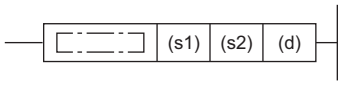
Operation error

Error code (SD0)	Description
3405H	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.

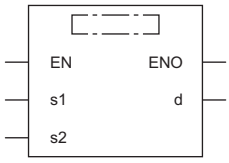
B+(P) [when three operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

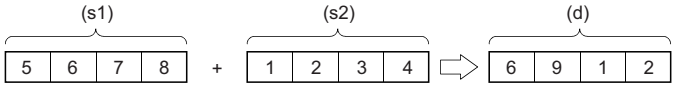
Operand	Description	Range	Data type	Data type (label)
(s1)	First addend data or the device where the first addend data is stored	0 to 9999	BCD 4-digit	ANY16
(s2)	Second addend data or the device where the second addend data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Device for storing the operation result	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

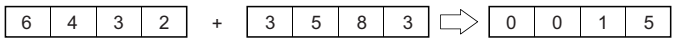
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

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

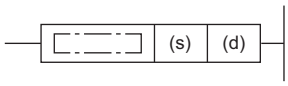
Error code (SD0)	Description
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 9999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 9999.

Subtracting BCD 4-digit data

B-(P) [when two operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions perform subtraction between the two sets of BCD 4-digit data specified.

Ladder	ST
	Not supported (☞ Page 259 B-(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 259 B-(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
B-	
B-P	

Setting data

■Description, range, data type

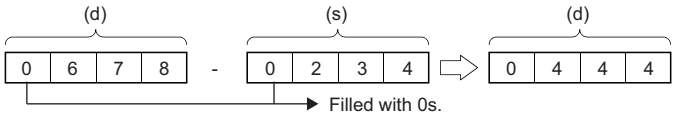
Operand	Description	Range	Data type	Data type (label)
(s)	Subtrahend data or the device where subtrahend data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Device where minuend data is stored	0 to 9999	BCD 4-digit	ANY16

■Applicable devices

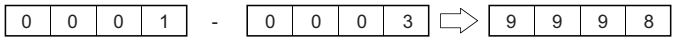
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

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.



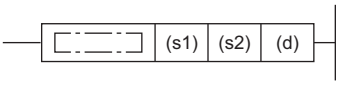
Operation error

Error code (SD0)	Description
3405H	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.

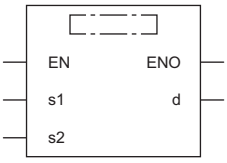
B-(P) [when three operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

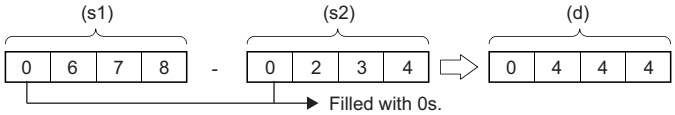
Operand	Description	Range	Data type	Data type (label)
(s1)	Minuend data or the device where minuend data is stored	0 to 9999	BCD 4-digit	ANY16
(s2)	Subtrahend data or the device where subtrahend data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Device for storing the operation result	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

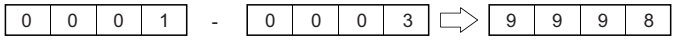
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐, U3E☐\(H)G☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

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

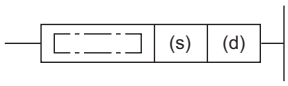
Error code (SD0)	Description
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 9999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 9999.

Adding BCD 8-digit data

DB+(P) [when two operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions add the two sets of BCD 8-digit data specified.

Ladder	ST
	Not supported (☞ Page 263 DB+(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 263 DB+(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
DB+	
DB+P	

Setting data

■Description, range, data type

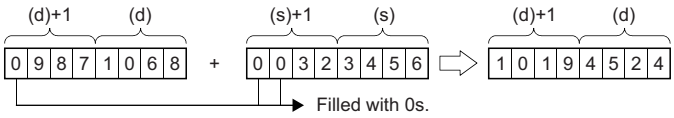
Operand	Description	Range	Data type	Data type (label)
(s)	Second addend data or the start device where the second addend data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device where the first addend data is stored	0 to 99999999	BCD 8-digit	ANY32

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

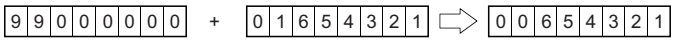
Processing details

- 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.



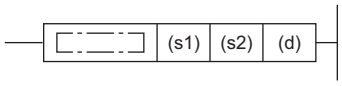
Operation error

Error code (SD0)	Description
3405H	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.

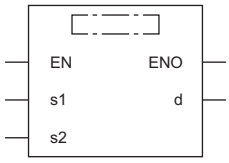
DB+(P) [when three operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

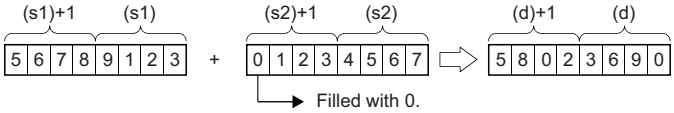
Operand	Description	Range	Data type	Data type (label)
(s1)	First addend data or the start device where the first addend data is stored	0 to 99999999	BCD 8-digit	ANY32
(s2)	Second addend data or the start device where the second addend data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for 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☐\☐, U3E☐\(H)G☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

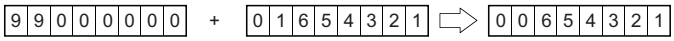
Processing details

- 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

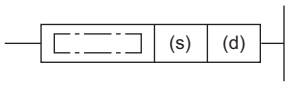
Error code (SD0)	Description
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 99999999.

Subtracting BCD 8-digit data

DB-(P) [when two operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions perform subtraction between the two sets of BCD 8-digit data specified.

Ladder	ST
	Not supported (☞ Page 267 DB-(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 267 DB-(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
DB-	
DB-P	

Setting data

■Description, range, data type

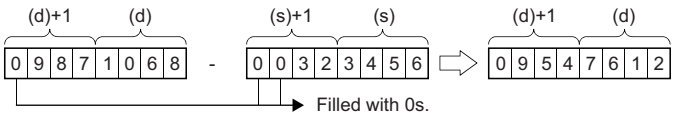
Operand	Description	Range	Data type	Data type (label)
(s)	Subtrahend data or the start device where subtrahend data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Minuend data or the start device where minuend data is stored	0 to 99999999	BCD 8-digit	ANY32

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

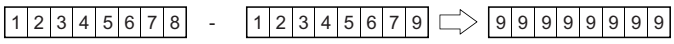
Processing details

- 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.



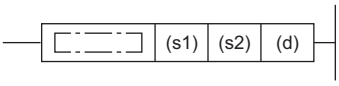
Operation error

Error code (SD0)	Description
3405H	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.

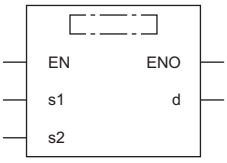
DB-(P) [when three operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

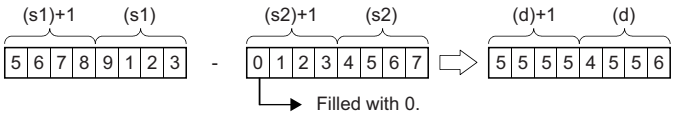
Operand	Description	Range	Data type	Data type (label)
(s1)	Minuend data or the start device where minuend data is stored	0 to 99999999	BCD 8-digit	ANY32
(s2)	Subtrahend data or the start device where subtrahend data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the operation 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

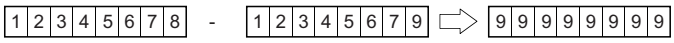
Processing details

- 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

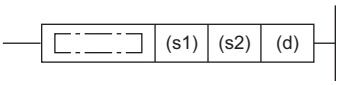
Error code (SD0)	Description
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 99999999.

Multiplying BCD 4-digit data

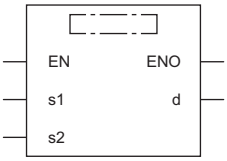
B*(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

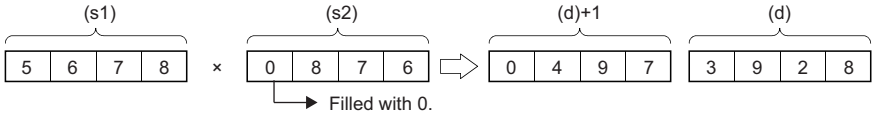
Operand	Description	Range	Data type	Data type (label)
(s1)	Multiplicand data or the device where multiplicand data is stored	0 to 9999	BCD 4-digit	ANY16
(s2)	Multiplier data or the device where multiplier data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the operation result	—	BCD 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□\□, U3E□\(H)G□	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

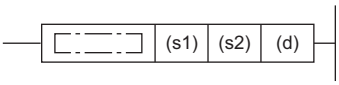
Error code (SD0)	Description
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 9999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 9999.

Dividing BCD 4-digit data

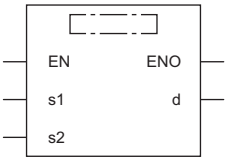
B/(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

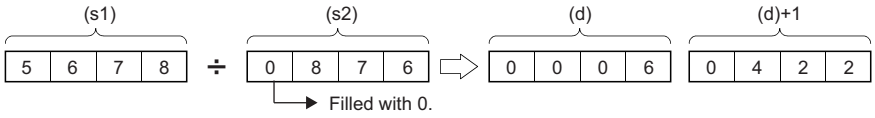
Operand	Description	Range	Data type	Data type (label)
(s1)	Dividend data or the device where dividend data is stored	0 to 9999	BCD 4-digit	ANY16
(s2)	Divisor data or the device where divisor data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the operation result	—	BCD 8-digit	ANY16_ARRAY (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☐\☐, U3E☐\(H)G☐	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

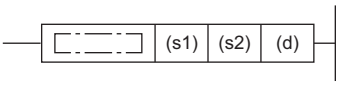
Error code (SD0)	Description
3400H	The value (divisor) in the device specified by (s2) is 0.
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 9999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 9999.

Multiplying BCD 8-digit data

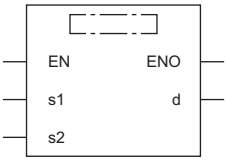
DB*(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

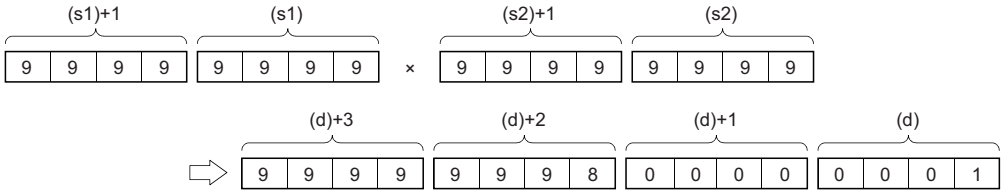
Operand	Description	Range	Data type	Data type (label)
(s1)	Multiplicand data or the start device where multiplicand data is stored	0 to 99999999	BCD 8-digit	ANY32
(s2)	Multiplier data or the start device where multiplier data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the 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□\□, U3E□\(H)G□	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

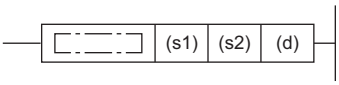
Error code (SD0)	Description
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 99999999.

Dividing BCD 8-digit data

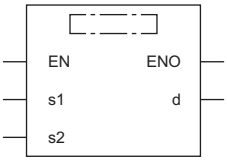
DB/(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

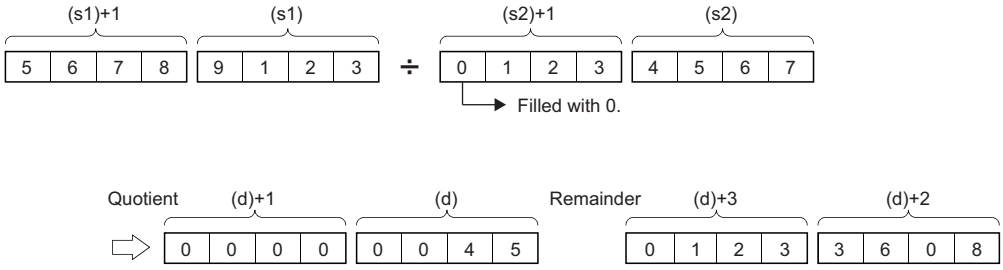
Operand	Description	Range	Data type	Data type (label)
(s1)	Dividend data or the start device where dividend data is stored	0 to 99999999	BCD 8-digit	ANY32
(s2)	Divisor data or the start device where divisor data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the operation result	—	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☐\☐, U3E☐\(H)G☐	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

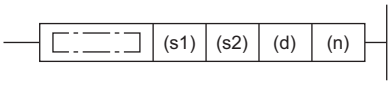
Error code (SD0)	Description
3400H	The value (divisor) in the device specified by (s2) is 0.
3405H	The BCD data in the device specified by (s1) is out of the range, 0 to 99999999.
	The BCD data in the device specified by (s2) is out of the range, 0 to 99999999.

Adding 16-bit binary block data

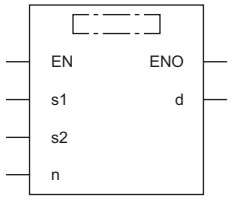
BK+(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions add the two 16-bit binary data blocks specified.

Ladder	ST*1	
	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.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■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

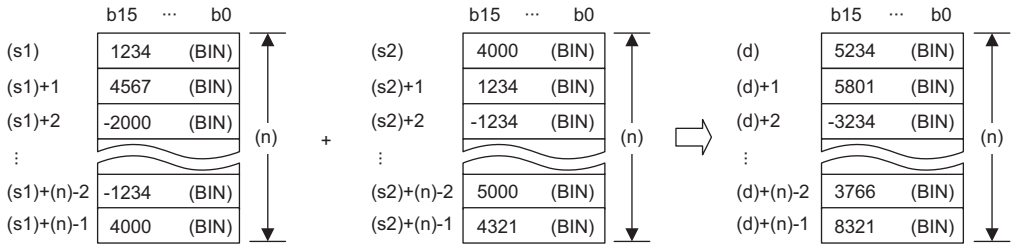
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

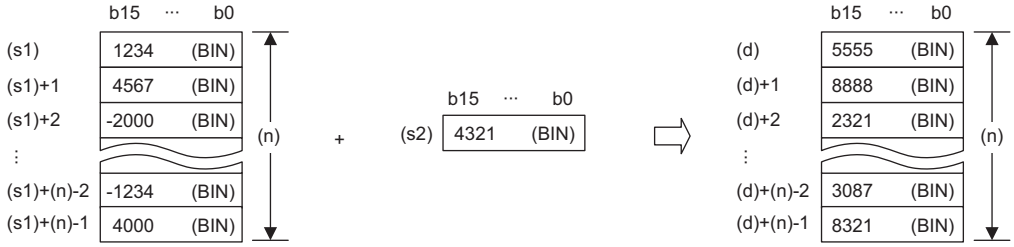
- These instructions add the (n) points of 16-bit binary data from the device specified by (s1) and the (n) points of 16-bit binary data from the device specified by (s2) or the constant, and store the operation result in the device specified by (d) and later.
- Specify data in units of 16 bits.

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

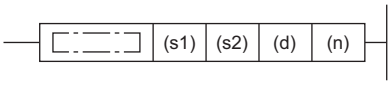
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are overlapping (except when the same device is specified for (s2) and (d)).

Subtracting 16-bit binary block data

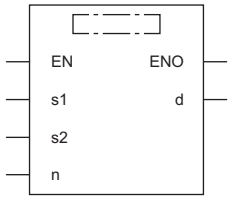
BK-(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions perform subtraction between the two 16-bit binary data blocks specified.

Ladder	ST*1	
	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.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■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

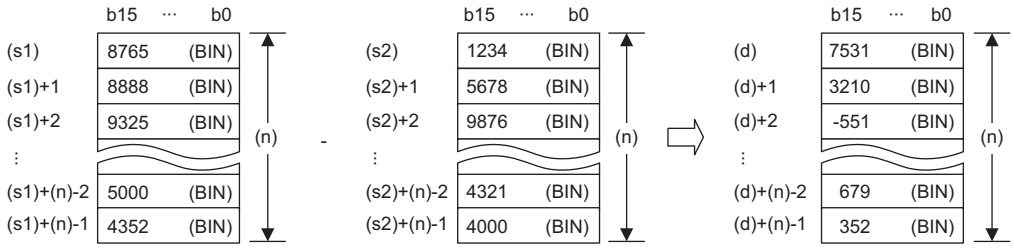
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐, U3E☐\(H)G☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

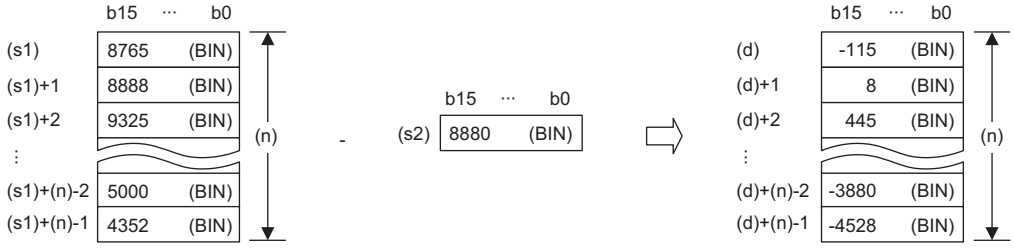
- These instructions subtract the (n) points of 16-bit binary data from the device specified by (s2) or the constant from the (n) points of 16-bit binary data from the device specified by (s1), and store the operation result in the device specified by (d) and later.
- Specify data in units of 16 bits.

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

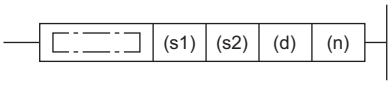
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are overlapping (except when the same device is specified for (s2) and (d)).

Adding 32-bit binary block data

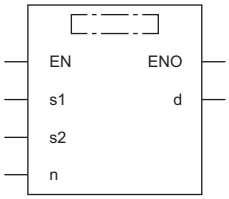
DBK+(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions add the two 32-bit binary data blocks specified.

Ladder	ST*1	
	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.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■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

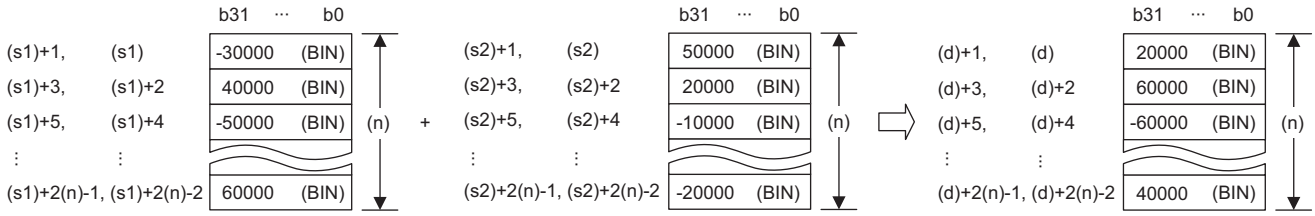
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

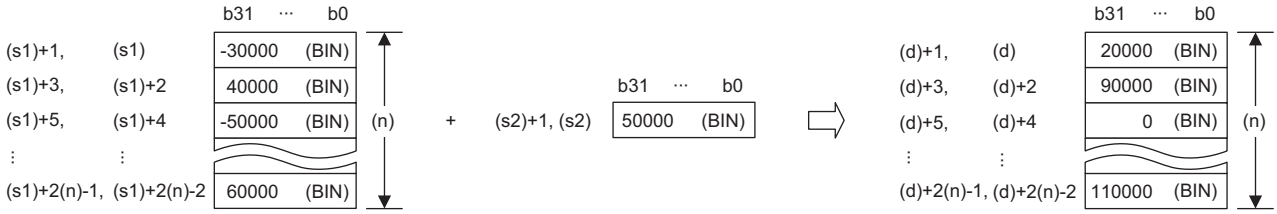
- These instructions add the (n) points of 32-bit binary data from the device specified by (s1) and the (n) points of 32-bit binary data from the device specified by (s2) or the constant, and store the operation result in the device specified by (d) and later.
- Specify data in units of 32 bits.

Ex.

When a device is specified by (s2) (signed value specification)



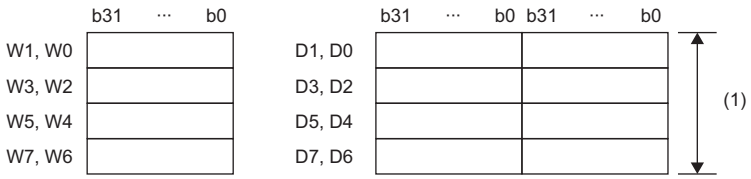
When a constant is specified by (s2) (signed value specification)



- 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.

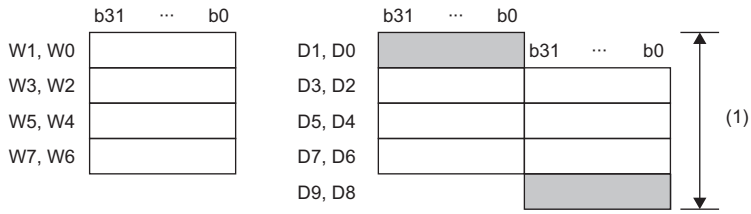
Ex.

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

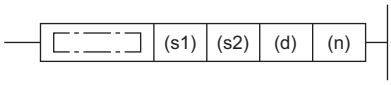
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are overlapping (except when the same device is specified for (s2) and (d)).

Subtracting 32-bit binary block data

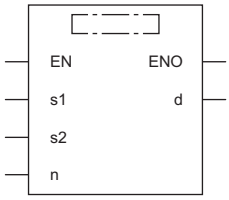
DBK-(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions perform subtraction between the two 32-bit binary data blocks specified.

Ladder	ST*1	
	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.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■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

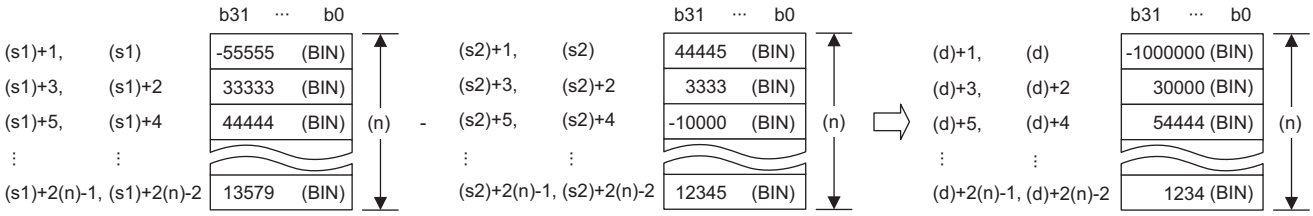
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐, U3E☐\(H)G☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

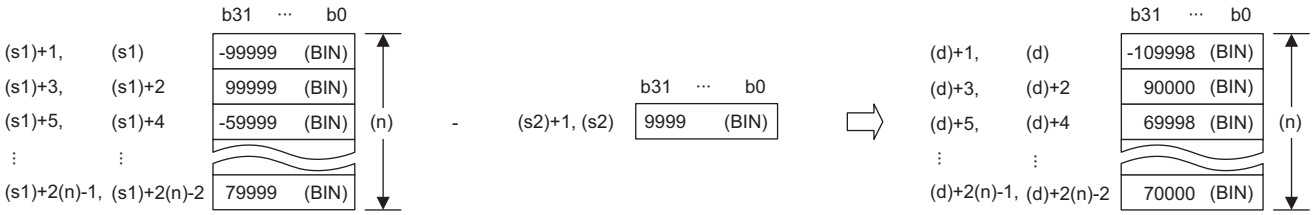
- These instructions subtract the (n) points of 32-bit binary data from the device specified by (s2) or the constant from the (n) points of 32-bit binary data from the device specified by (s1), and store the operation result in the device specified by (d) and later.
- Specify data in units of 32 bits.

Ex.

When a device is specified by (s2) (signed value specification)



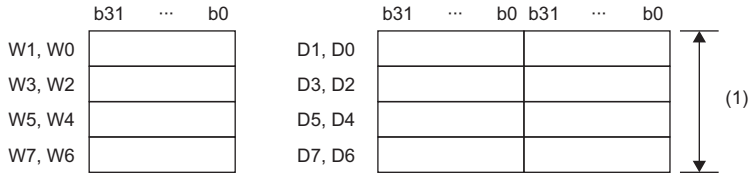
When a constant is specified by (s2) (signed value specification)



- 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.

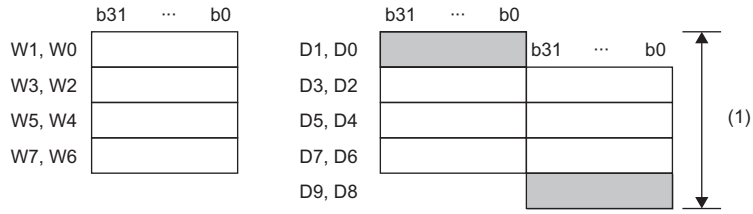
Ex.

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)	K0 (H00000000) - K1 (H00000001) ⇒ K4294967295 (HFFFFFFFF)
K-2147483647 (H80000001) - K2 (H00000002) ⇒ K2147483647 (H7FFFFFFF)	

Operation error

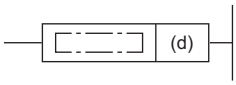
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are overlapping (except when the same device is specified for (s2) and (d)).

Incrementing 16-bit binary data

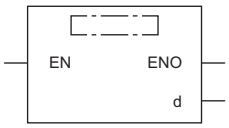
INC(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

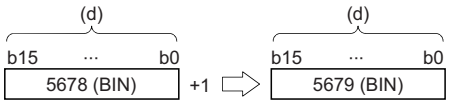
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

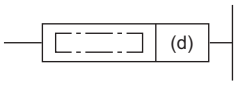
There is no operation error.

Decrementing 16-bit binary data

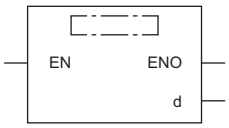
DEC(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

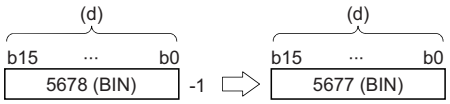
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

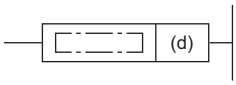
There is no operation error.

Incrementing 32-bit binary data

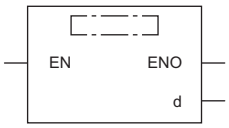
DINC(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

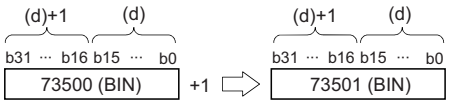
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	○	—

Processing details

- These instructions increment the 32-bit binary data in the device specified by (d) by one.



- When the DINC(P) 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

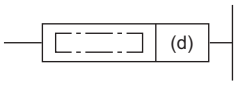
There is no operation error.

Decrementing 32-bit binary data

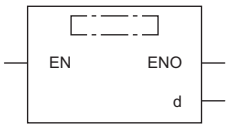
DDEC(P)(_U)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

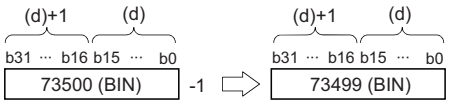
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(d)	○	○	—

Processing details

- These instructions decrement the 32-bit binary data in the device specified by (d) by one.



- When the DDEC(P) 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) 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.

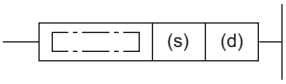
6.3 Logical Operation Instructions

Performing an AND operation on 16-bit data

WAND(P) [when two operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform an AND operation on the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 297 WAND(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 297 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

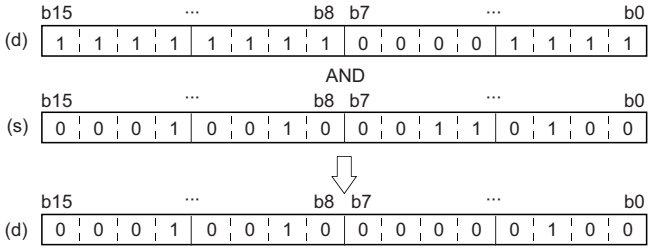
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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.

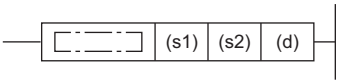
Operation error

There is no operation error.

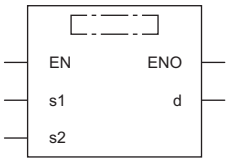
WAND(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

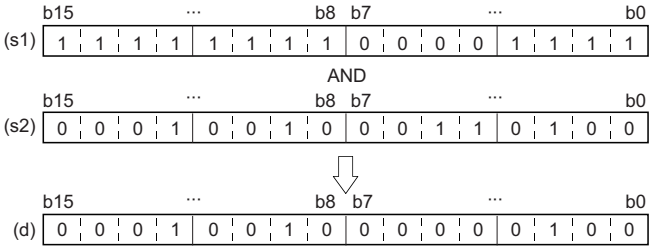
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

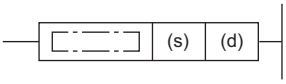
There is no operation error.

Performing an AND operation on 32-bit data

DAND(P) [when two operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform an AND operation on the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 301 DAND(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 301 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

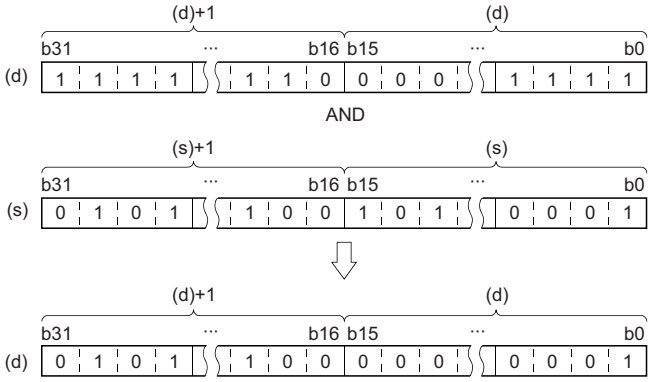
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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.

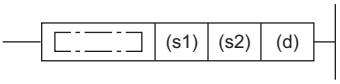
Operation error

There is no operation error.

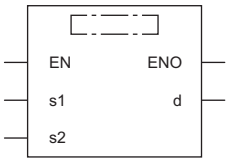
DAND(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

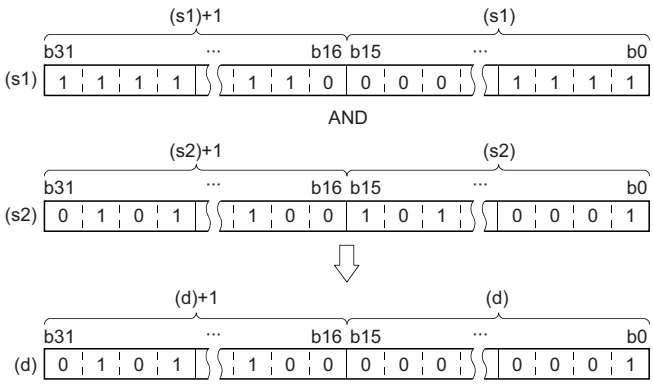
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

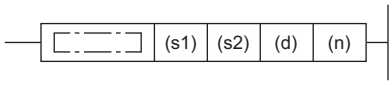
There is no operation error.

Performing an AND operation on 16-bit block data

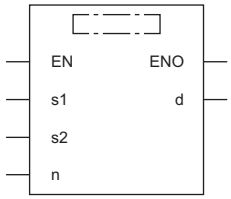
BKAND(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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.

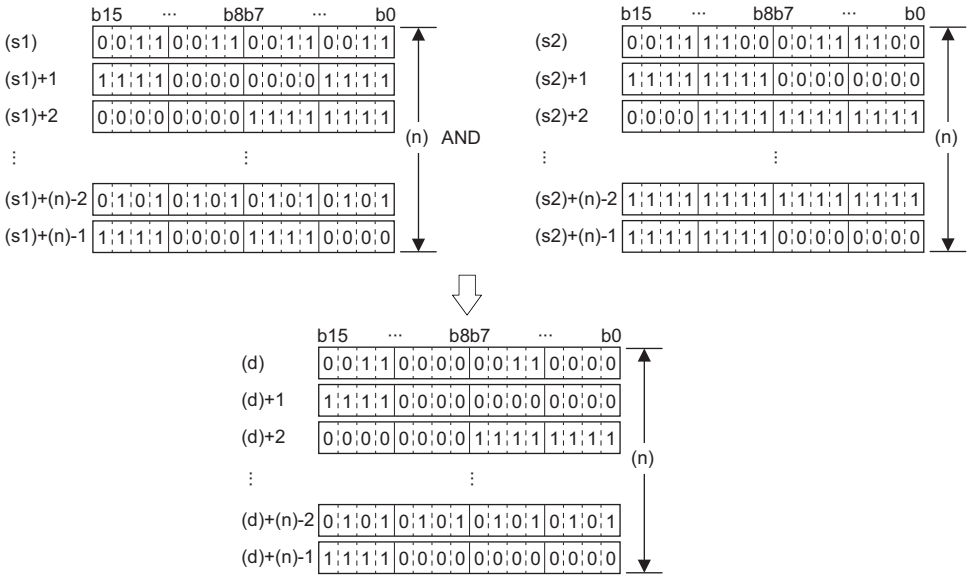
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)*1	—	—	○	—	—	—	—	○	—	—	—	—
(s2)*1	—	—	○	—	—	—	—	○	○	—	—	—
(d)*1	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

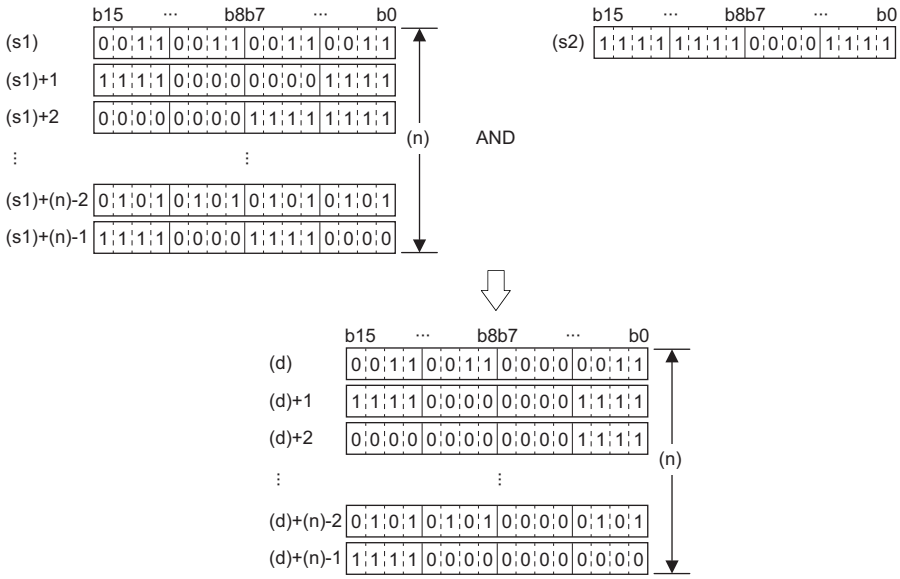
*1 The same device number can be specified for (s1) and (d) or (s2) and (d).

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

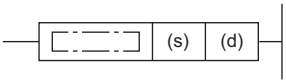
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are partially overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are partially overlapping (except when the same device is specified for (s2) and (d)).

Performing an OR operation on 16-bit data

WOR(P) [when two operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform an OR operation on the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 307 WOR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 307 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

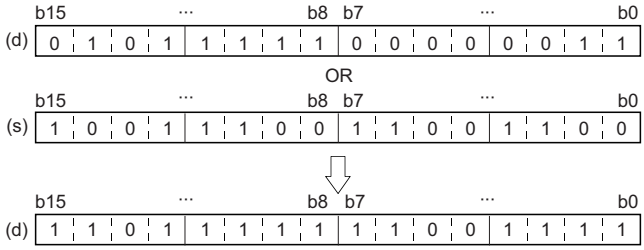
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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.

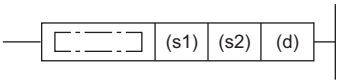
Operation error

There is no operation error.

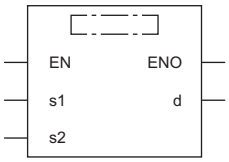
WOR(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

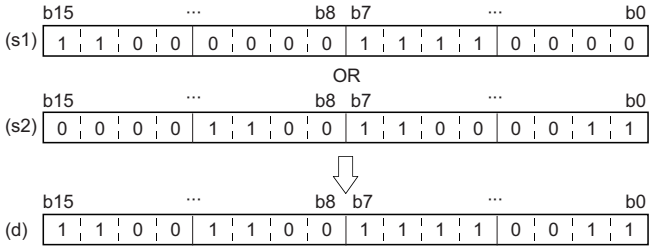
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

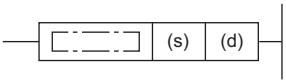
There is no operation error.

Performing an OR operation on 32-bit data

DOR(P) [when two operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform an OR operation on the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 311 DOR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 311 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

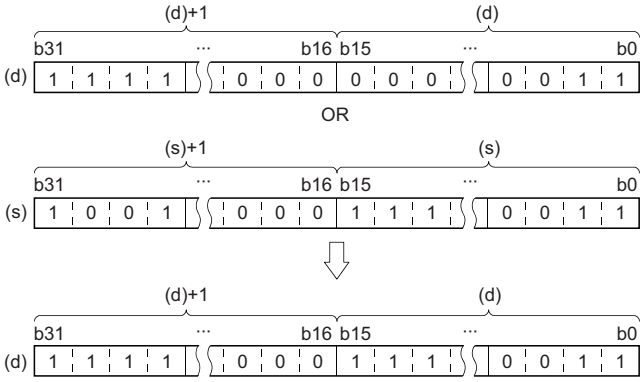
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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.

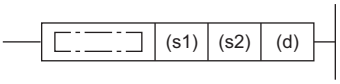
Operation error

There is no operation error.

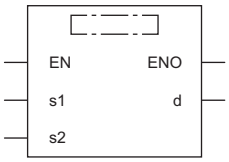
DOR(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

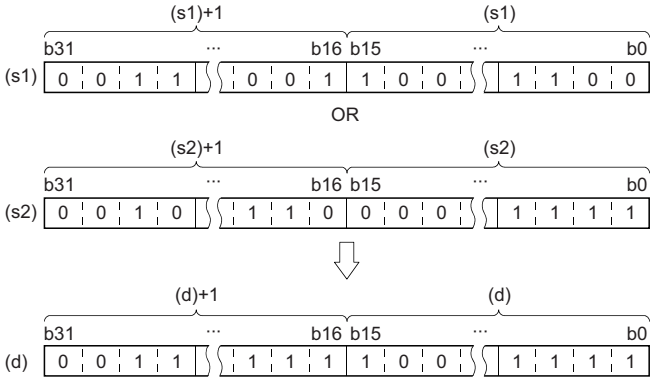
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

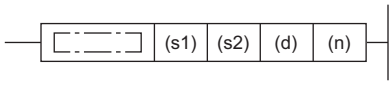
There is no operation error.

Performing an OR operation on 16-bit block data

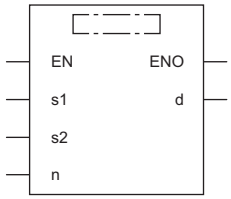
BKOR(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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.

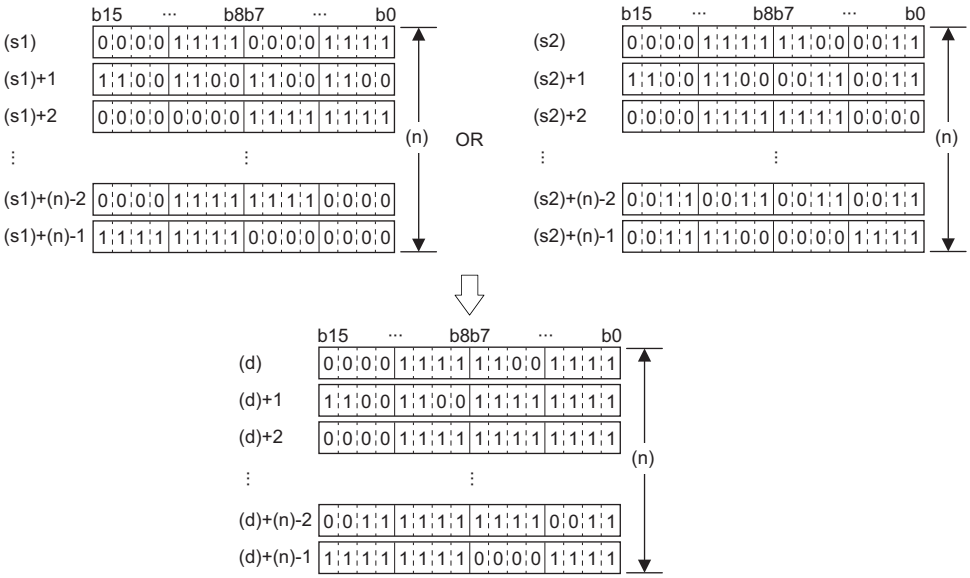
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)*1	—	—	○	—	—	—	—	○	—	—	—	—
(s2)*1	—	—	○	—	—	—	—	○	○	—	—	—
(d)*1	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

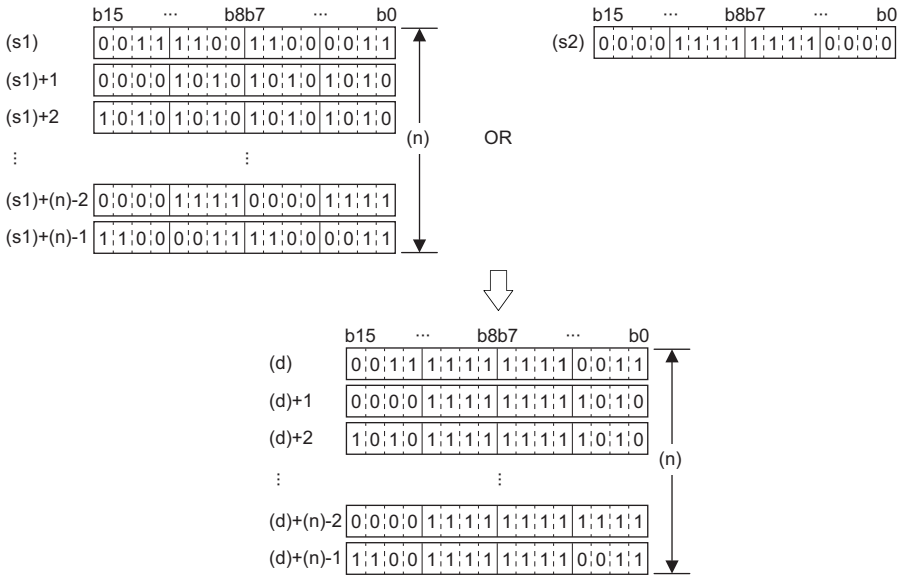
*1 The same device number can be specified for (s1) and (d) or (s2) and (d).

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

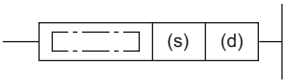
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are partially overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are partially overlapping (except when the same device is specified for (s2) and (d)).

Performing an XOR operation on 16-bit data

WXOR(P) [when two operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform an XOR operation on the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 317 WXOR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 317 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

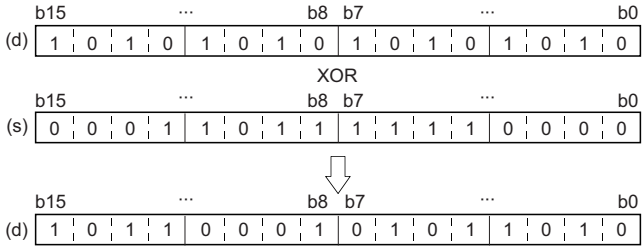
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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.

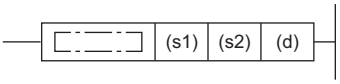
Operation error

There is no operation error.

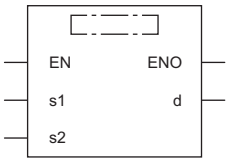
WXOR(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

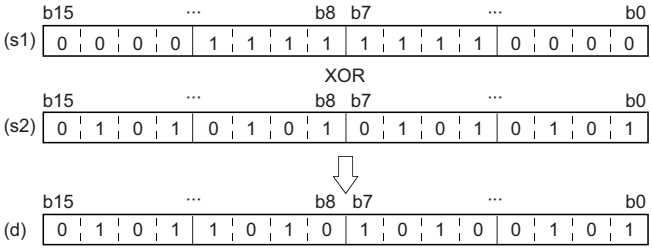
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

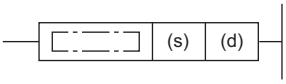
There is no operation error.

Performing an XOR operation on 32-bit data

DXOR(P) [when two operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform an XOR operation on the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 321 DXOR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 321 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

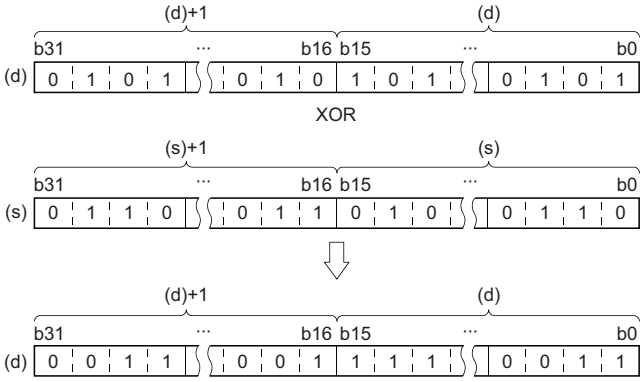
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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.

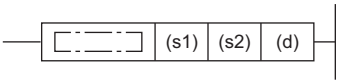
Operation error

There is no operation error.

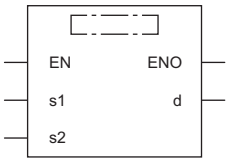
DXOR(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

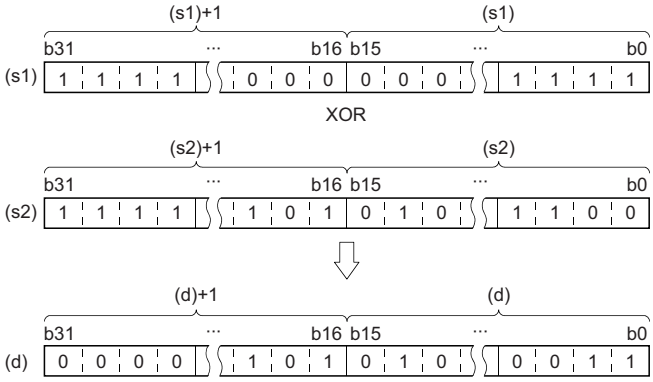
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

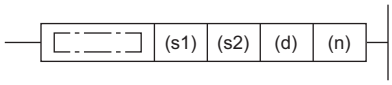
There is no operation error.

Performing an XOR operation on 16-bit block data

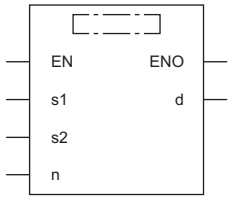
BKXOR(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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.

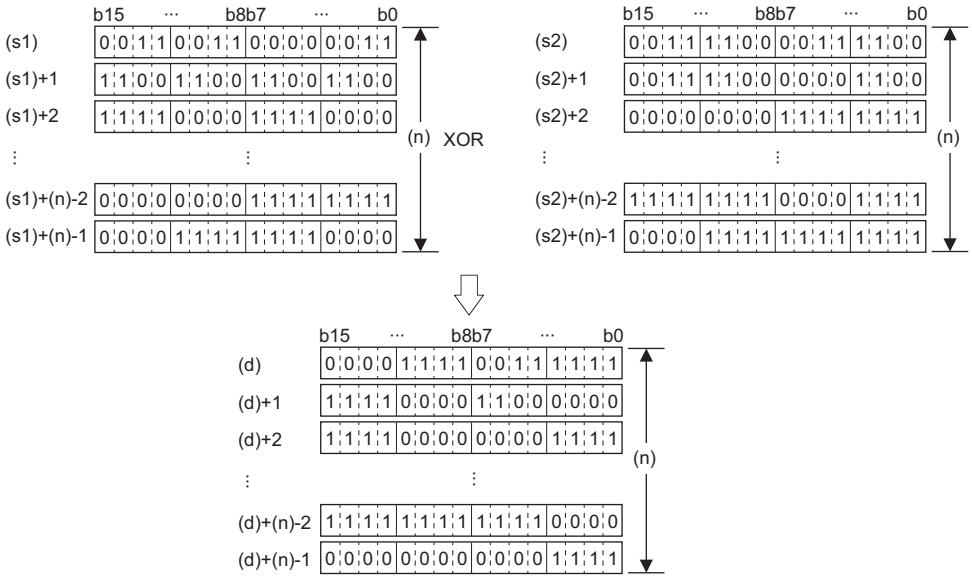
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)*1	—	—	○	—	—	—	—	○	—	—	—	—
(s2)*1	—	—	○	—	—	—	—	○	○	—	—	—
(d)*1	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

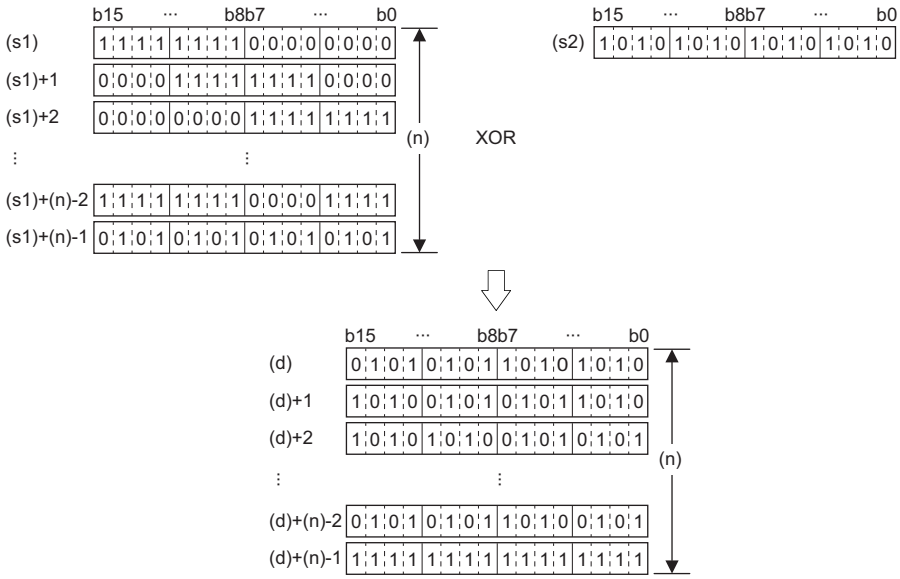
*1 The same device number can be specified for (s1) and (d) or (s2) and (d).

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

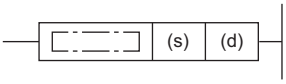
Error code (SD0)	Description
2821H	The device ranges starting from the ones specified by (s1) and (d) are partially overlapping (except when the same device is specified for (s1) and (d)).
	The device ranges starting from the ones specified by (s2) and (d) are partially overlapping (except when the same device is specified for (s2) and (d)).

Performing an XNOR operation on 16-bit data

WXNR(P) [when two operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform an XNOR operation on the two sets of 16-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 327 WXNR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 327 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

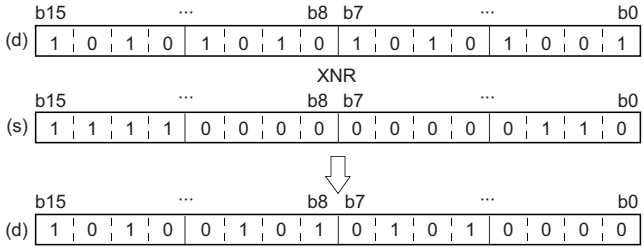
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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.

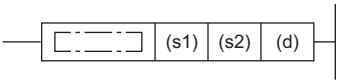
Operation error

There is no operation error.

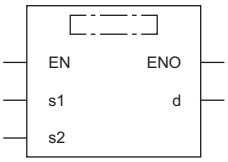
WXNR(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

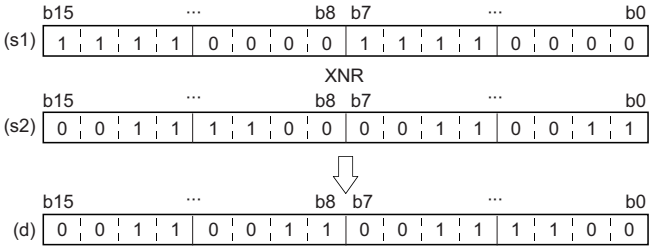
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

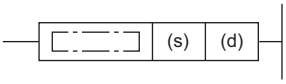
There is no operation error.

Performing an XNOR operation on 32-bit data

DXNR(P) [when two operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions perform an XNOR operation on the two sets of 32-bit binary data specified.

Ladder	ST
	Not supported (☞ Page 331 DXNR(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 331 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

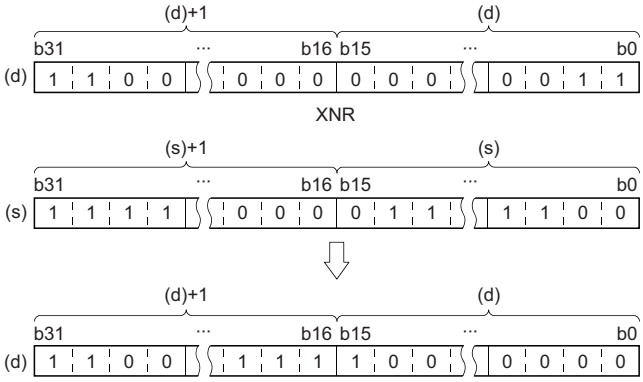
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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.

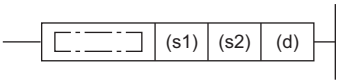
Operation error

There is no operation error.

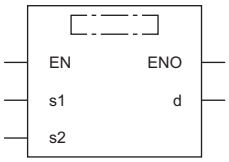
DXNR(P) [when three operands are set]

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

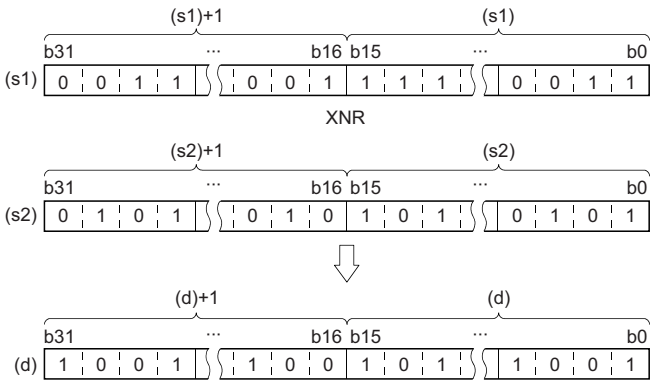
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

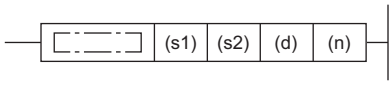
There is no operation error.

Performing an XNOR operation on 16-bit block data

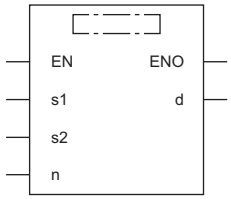
BKXNR(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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.

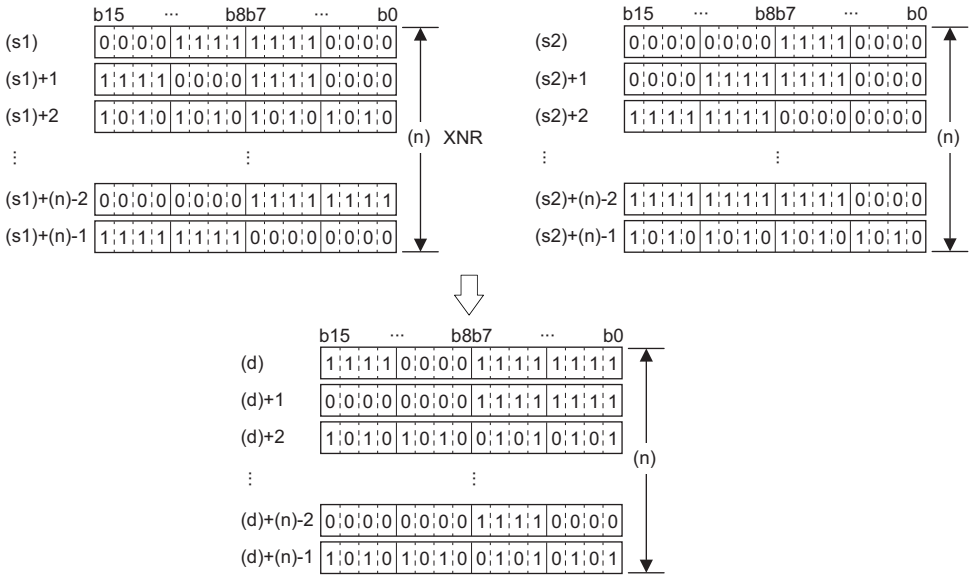
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)*1	—	—	○	—	—	—	—	○	—	—	—	—
(s2)*1	—	—	○	—	—	—	—	○	○	—	—	—
(d)*1	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

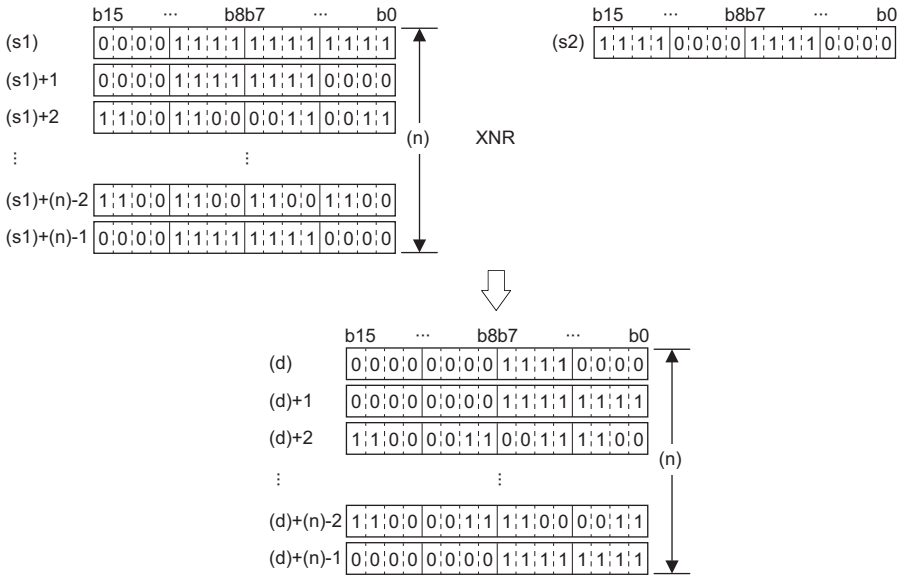
*1 The same device number can be specified for (s1) and (d) or (s2) and (d).

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)).

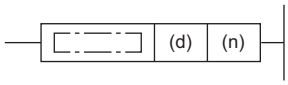
6.4 Bit Processing Instructions

Setting a bit in the word device

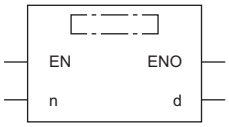
BSET(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

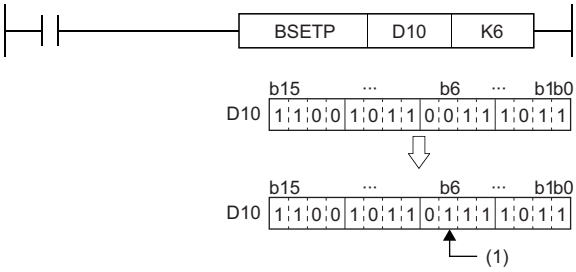
Operand	Description	Range	Data type	Data type (label)
(d)	Set target device	—	16-bit signed binary	ANY16
(n)	Set target bit position	0 to 15	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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

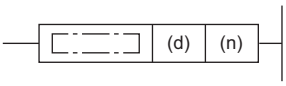
There is no operation error.

Resetting a bit in the word device

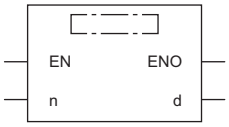
BRST(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

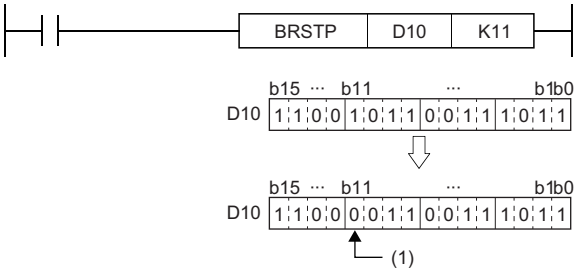
Operand	Description	Range	Data type	Data type (label)
(d)	Reset target device	—	16-bit signed binary	ANY16
(n)	Reset target bit position	0 to 15	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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

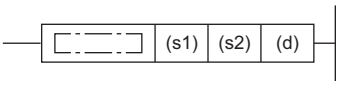
There is no operation error.

Performing a 16-bit test

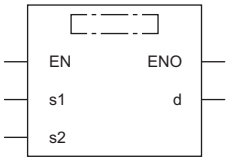
TEST(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

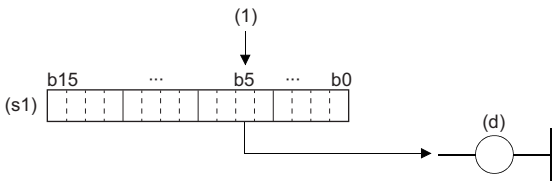
Operand	Description	Range	Data type	Data type (label)
(s1)	Device where the extract target bit data is stored	—	16-bit signed binary	ANY16
(s2)	Extract target bit position	0 to 15	16-bit unsigned binary	ANY16
(d)	Device for storing the extracted bit data	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
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□\□, U3E□\(H)G□	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

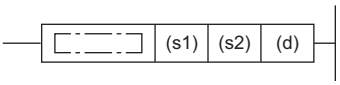
There is no operation error.

Performing a 32-bit test

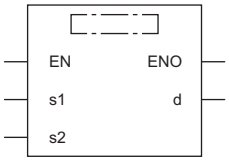
DTEST(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

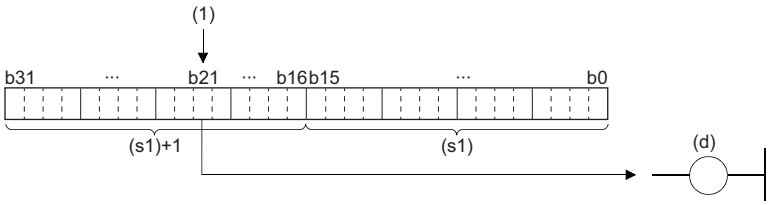
Operand	Description	Range	Data type	Data type (label)
(s1)	Device where the extract target bit data is stored	—	32-bit signed binary	ANY32
(s2)	Extract target bit position	0 to 31	16-bit unsigned binary	ANY16
(d)	Device for storing the extracted bit data	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
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□\□, U3E□\(H)G□	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

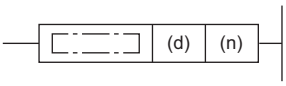
There is no operation error.

Batch-resetting bit devices

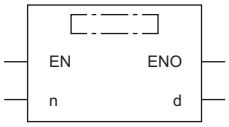
BKRST(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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.

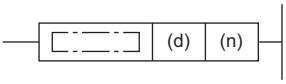
6.5 Shift Instructions

Shifting 16-bit binary data to the right by n bit(s)

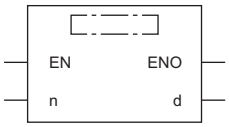
SFR(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	○	○	○	○	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

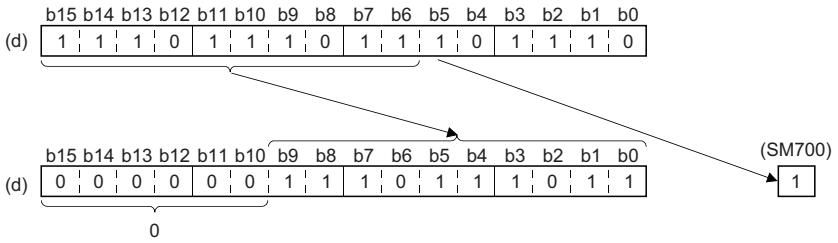
*1 FX and FY cannot be used.

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, a value in a bit to the right of the shift target area is stored.

Ex.

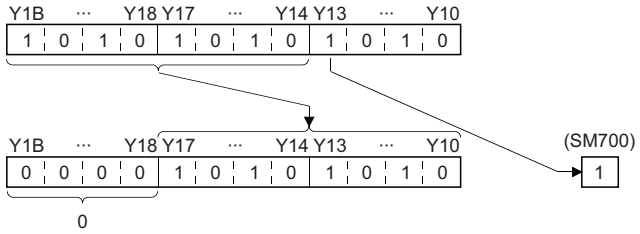
When (n)=6



- 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

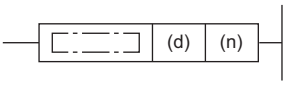
There is no operation error.

Shifting 16-bit binary data to the left by n bit(s)

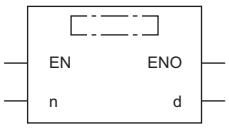
SFL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	○	○	○	○	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

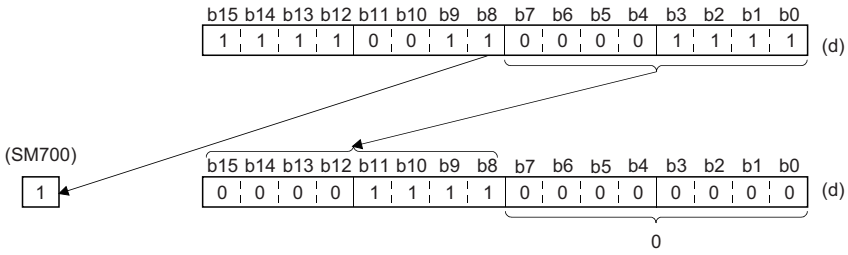
*1 FX and FY cannot be used.

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, a value in a bit to the left of the shift target area is stored.

Ex.

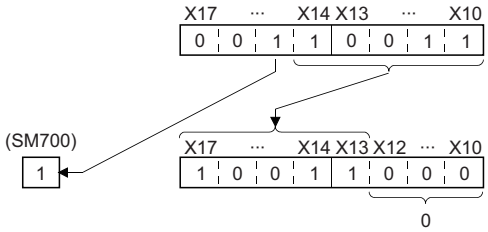
When (n)=8



- 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

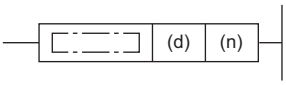
There is no operation error.

Shifting n-bit data to the right by one bit

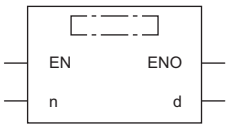
BSFR(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

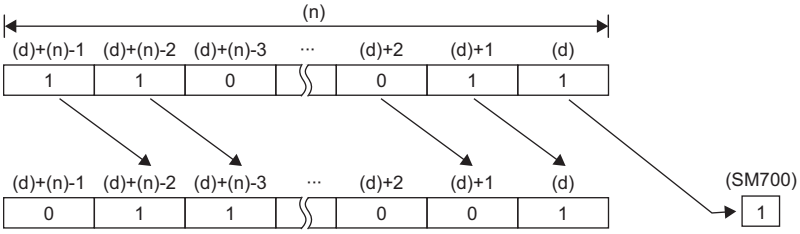
■Applicable devices

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□\□, U3E□\(H)G□	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, 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

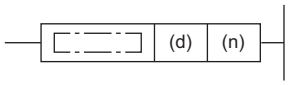
There is no operation error.

Shifting n-bit data to the left by one bit

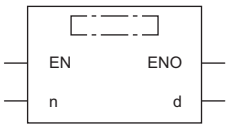
BSFL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

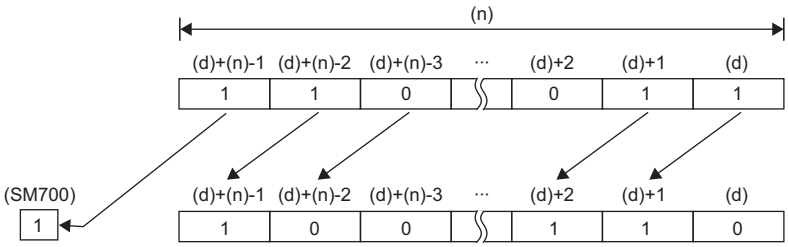
■Applicable devices

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□\□, U3E□\(H)G□	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, 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

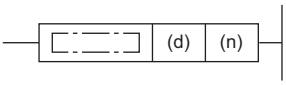
There is no operation error.

Shifting n-word data to the right by one word

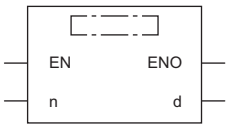
DSFR(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

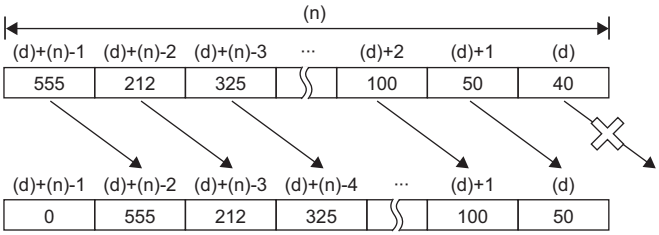
■Applicable devices

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□\□, U3E□\(H)G□	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

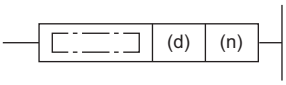
There is no operation error.

Shifting n-word data to the left by one word

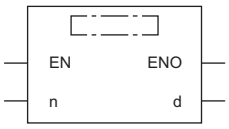
DSFL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

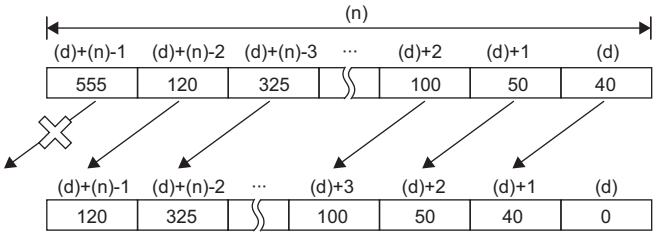
■Applicable devices

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□\□, U3E□\(H)G□	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.



Operation error

There is no operation error.

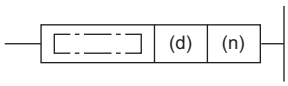
Shifting n double word(s) of data to the right by one double word

DDSFR(P)

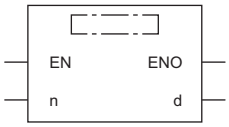
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift the (n) double word(s) of data starting from the specified device to the right by one double word. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=DDSFR(EN,n,d); ENO:=DDSFRP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DDSFR	
DDSFRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target start device	—	Double word	ANY32
(n)	Amount of data to be shifted	0 to 65535	16-bit unsigned 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

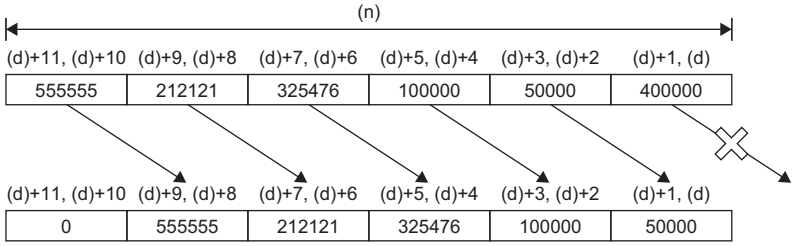
*1 FX and FY cannot be used.

Processing details

- These instructions shift the (n) double word(s) of data starting from the device specified by (d) to the right by one double word.
- One double word from the most significant bit is filled with 0.
- If (n) is 0, no processing is performed.

Ex.

When (n)=6



Operation error

There is no operation error.

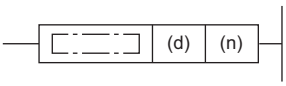
Shifting n double word(s) of data to the left by one double word

DDSFL(P)

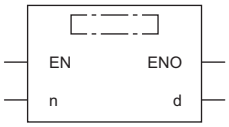
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift the (n) double word(s) of data starting from the specified device to the left by one double word. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=DDSFL(EN,n,d); ENO:=DDSFLP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DDSFL	
DDSFLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target start device	—	Double word	ANY32
(n)	Amount of data to be shifted	0 to 65535	16-bit unsigned 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

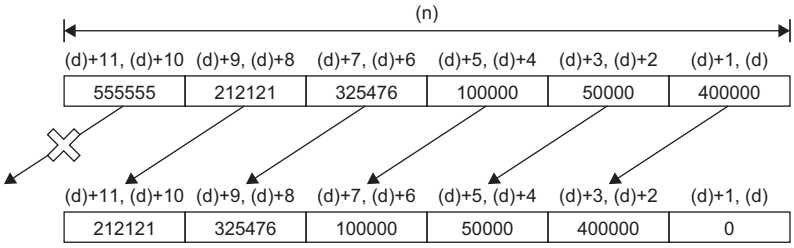
*1 FX and FY cannot be used.

Processing details

- These instructions shift the (n) double word(s) of data starting from the device specified by (d) to the left by one double word.
- One double word from the least significant bit is filled with 0.
- If (n) is 0, no processing is performed.

Ex.

When (n)=6



Operation error

There is no operation error.

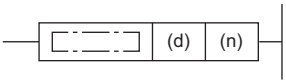
Shifting n point(s) of single-precision real number data to the right by one point

ESFR(P)

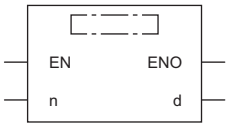
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift the (n) points of single-precision real number data starting from the specified device to the right by one point. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=ESFR(EN,n,d); ENO:=ESFRP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ESFR	
ESFRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target start device	—	Single-precision real number	ANYREAL_32
(n)	Amount of data to be shifted	0 to 65535	16-bit unsigned 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

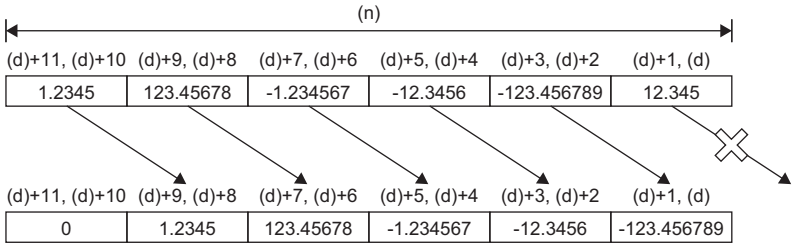
*1 FX and FY cannot be used.

Processing details

- These instructions shift the (n) points of single-precision real number data starting from the device specified by (d) to the right by one point.
- One point of single-precision real number data from the most significant bit is filled with 0.
- If (n) is 0, no processing is performed.

Ex.

When (n)=6



Operation error

There is no operation error.

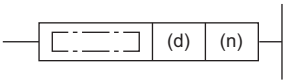
Shifting n point(s) of single-precision real number data to the left by one point

ESFL(P)

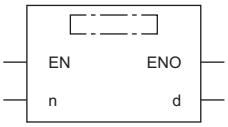
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift the (n) points of single-precision real number data starting from the specified device to the left by one point. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=ESFL(EN,n,d); ENO:=ESFLP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ESFL	
ESFLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target start device	—	Single-precision real number	ANYREAL_32
(n)	Amount of data to be shifted	0 to 65535	16-bit unsigned 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

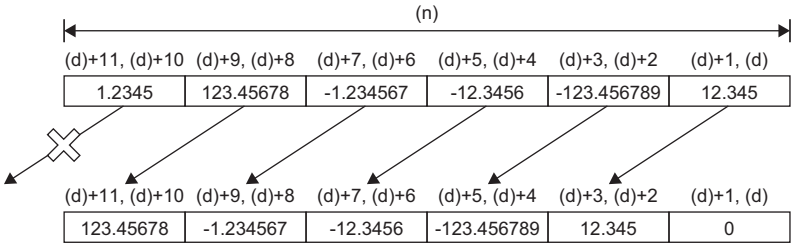
*1 FX and FY cannot be used.

Processing details

- These instructions shift the (n) points of single-precision real number data starting from the device specified by (d) to the left by one point.
- One point of single-precision real number data from the least significant bit is filled with 0.
- If (n) is 0, no processing is performed.

Ex.

When (n)=6



Operation error

There is no operation error.

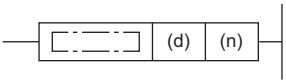
Shifting n point(s) of double-precision real number data to the right by one point

EDSFR(P)

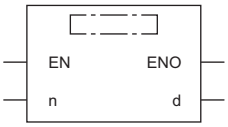
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift the (n) points of double-precision real number data starting from the specified device to the right by one point. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=EDSFR(EN,n,d); ENO:=EDSFRP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EDSFR	
EDSFRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target start device	—	Double-precision real number	ANYREAL_64
(n)	Amount of data to be shifted	0 to 65535	16-bit unsigned 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

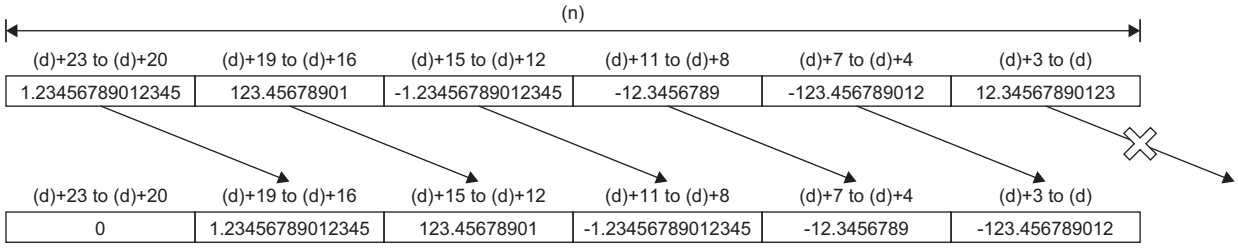
*1 FX and FY cannot be used.

Processing details

- These instructions shift the (n) points of double-precision real number data starting from the device specified by (d) to the right by one point.
- One point of double-precision real number data from the most significant bit is filled with 0.
- If (n) is 0, no processing is performed.

Ex.

When (n)=6



Operation error

There is no operation error.

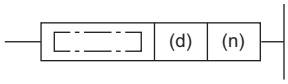
Shifting n point(s) of double-precision real number data to the left by one point

EDSFL(P)

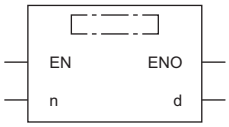
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift the (n) points of double-precision real number data starting from the specified device to the left by one point. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=EDSFL(EN,n,d); ENO:=EDSFLP(EN,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EDSFL	
EDSFLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target start device	—	Double-precision real number	ANYREAL_64
(n)	Amount of data to be shifted	0 to 65535	16-bit unsigned 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

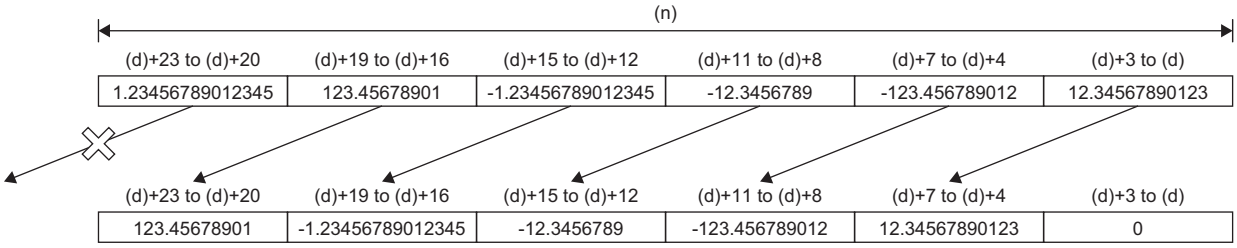
*1 FX and FY cannot be used.

Processing details

- These instructions shift the (n) points of double-precision real number data starting from the device specified by (d) to the left by one point.
- One point of double-precision real number data from the least significant bit is filled with 0.
- If (n) is 0, no processing is performed.

Ex.

When (n)=6



Operation error

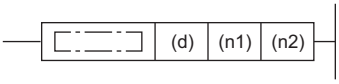
There is no operation error.

Shifting n-bit data to the right by n bit(s)

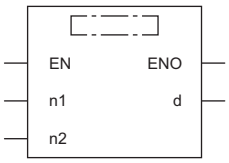
SFTBR(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*2	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*2	○	○	○	○	—	—	○	○	—	—	—

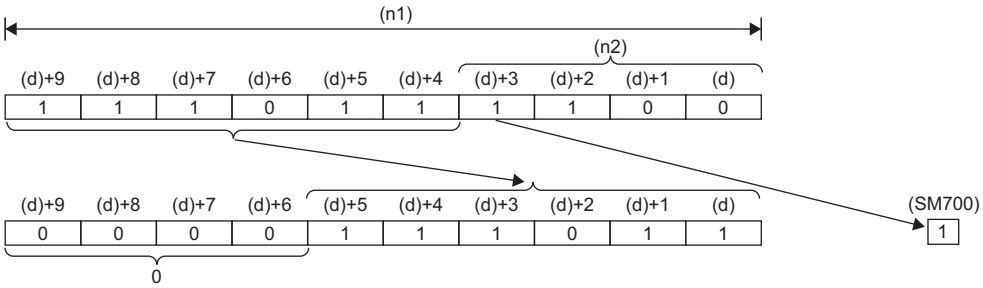
*1 T, C, and ST cannot be used.
*2 FX and FY cannot be used.

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, 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

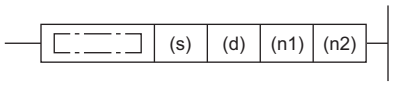
Error code (SD0)	Description
3405H	The value specified by (n1) is out of the range, 0 to 64.

SFTR(P)

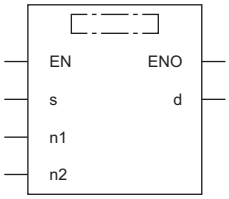
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions shift bit data to the right by the n2 bit(s) within the (n1) bits of data area starting from the specified device. In the empty area after the shift, specified data is stored.

Ladder	ST
	ENO:=SFTR(EN,s,n1,n2,d); ENO:=SFTRP(EN,s,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTR	
SFTRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device stored in the empty area after the shift	—	Bit	ANY_BOOL
(d)	Shift target start device	—	Bit	ANY_BOOL
(n1)*1	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16_U
(n2)*1	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Set values so that (n2)≤(n1).

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(s)	○	—	○*1	—	—	—	—	○	○*2	—	—	—
(d)	○	—	○*1	—	—	—	—	○	—	—	—	—
(n1)	○*3	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*3	○	○	○	○	—	—	○	○	—	—	—

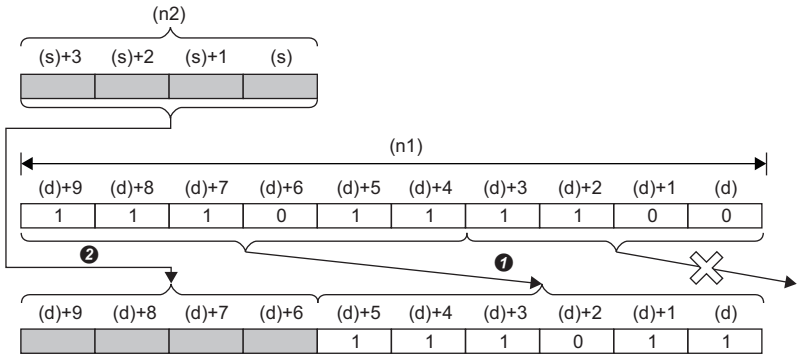
*1 T, ST, C, and FD cannot be used.
*2 Only 0 or 1 can be used.
*3 FX and FY cannot be used.

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 the empty area after the shift, (n2) point(s) of data starting from (s) is/are stored.
- When constant 0 is specified for (s), 0s are stored in (n2) points from the most significant bit after the shift.
- When constant 1 is specified for (s), 1s are stored in (n2) points from the most significant bit after the shift.
- When (n2) is 0, the processing is not performed.

Ex.

When (n1)=10 and (n2)=4



❶ Shift to the right by (n2)-bit

❷ Copy

Operation error

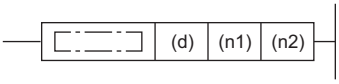
Error code (SD0)	Description
2821H	The range for (n2) points from (s) and that for (n1) points from (d) are overlapping.
3405H	A constant other than 0 or 1 is specified when the constant (s) is specified.
	The values specified in (n1) and (n2) are such that (n1)<(n2).

Shifting n-bit data to the left by n bit(s)

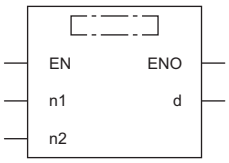
SFTBL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○*1	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*2	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*2	○	○	○	○	—	—	○	○	—	—	—

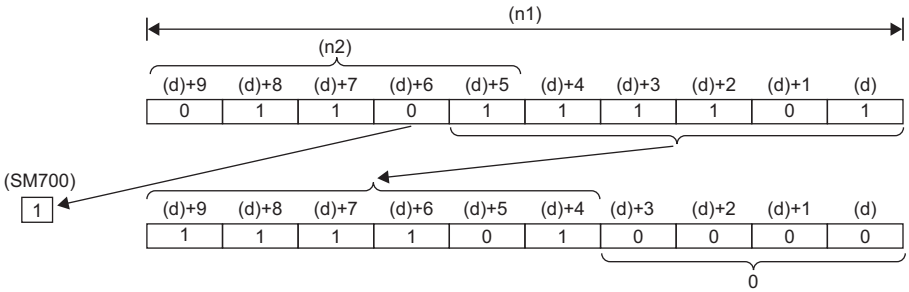
*1 T, C, and ST cannot be used.
*2 FX and FY cannot be used.

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, 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

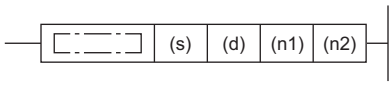
Error code (SD0)	Description
3405H	The value specified by (n1) is out of the range, 0 to 64.

SFTL(P)

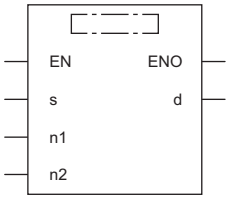
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions shift bit data to the left by the (n2) bit(s) of area within the (n1) bits of data area starting from the specified device. In the empty area after the shift, specified data is stored.

Ladder	ST
	ENO:=SFTL(EN,s,n1,n2,d); ENO:=SFTLP(EN,s,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTL	
SFTLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device stored in the empty area after the shift	—	Bit	ANY_BOOL
(d)	Shift target start device	—	Bit	ANY_BOOL
(n1)*1	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16_U
(n2)*1	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Set values so that (n2)≤(n1).

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(s)	○	—	○*1	—	—	—	—	○	○*2	—	—	—
(d)	○	—	○*1	—	—	—	—	○	—	—	—	—
(n1)	○*3	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*3	○	○	○	○	—	—	○	○	—	—	—

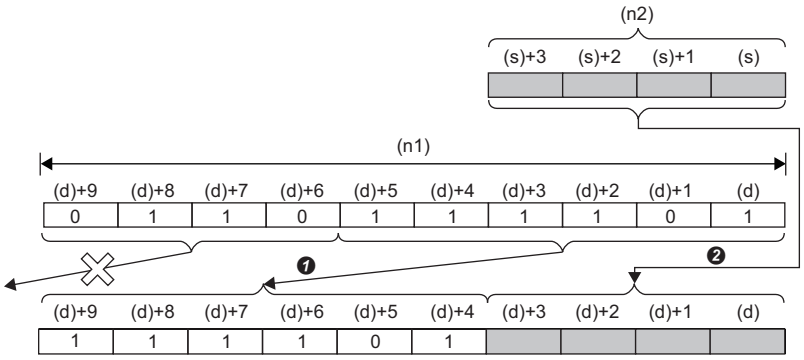
*1 T, ST, C, and FD cannot be used.
*2 Only 0 or 1 can be used.
*3 FX and FY cannot be used.

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 the empty area after the shift, (n2) point(s) of data starting from (s) is/are stored.
- When constant 0 is specified for (s), 0s are stored in (n2) points from the least significant bit after the shift.
- When constant 1 is specified for (s), 1s are stored in (n2) points from the least significant bit after the shift.
- When (n2) is 0, the processing is not performed.

Ex.

When (n1)=10 and (n2)=4



❶ Shift to the left by (n2)-bit

❷ Copy

Operation error

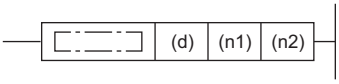
Error code (SD0)	Description
2821H	The range for (n2) points from (s) and that for (n1) points from (d) are overlapping.
3405H	A constant other than 0 or 1 is specified when the constant (s) is specified.
	The values specified in (n1) and (n2) are such that (n1)<(n2).

Shifting n-word data to the right by n word(s)

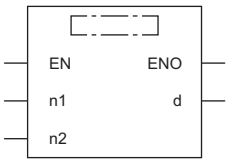
SFTWR(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

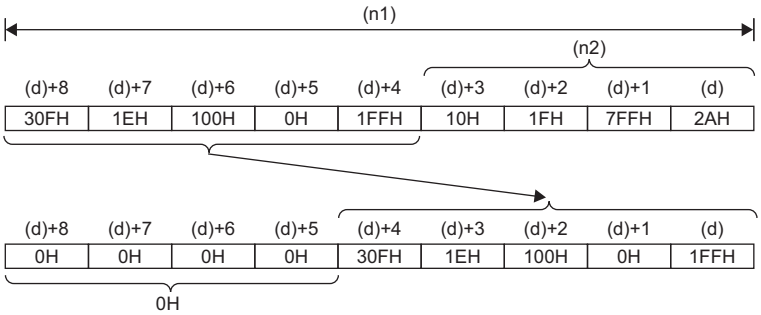
*1 FX and FY cannot be used.

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

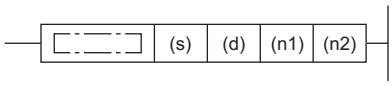
There is no operation error.

WSFR(P)

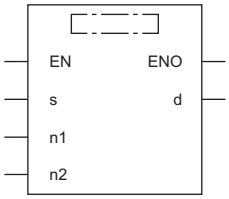
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions shift word data to the right by the (n2) word(s) within the (n1) words of data area starting from the specified device. In the empty area after the shift, specified data is stored.

Ladder	ST
	ENO:=WSFR(EN,s,n1,n2,d); ENO:=WSFRP(EN,s,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
WSFR	
WSFRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device stored in the empty area after the shift	—	16-bit unsigned binary	ANY16
(d)	Shift target start device	—	Word	ANY16
(n1)*1	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16_U
(n2)*1	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Set values so that (n2)≤(n1).

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(s)	○*1	—	○	—	—	—	—	○	○	—	—	—
(d)	○*1	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

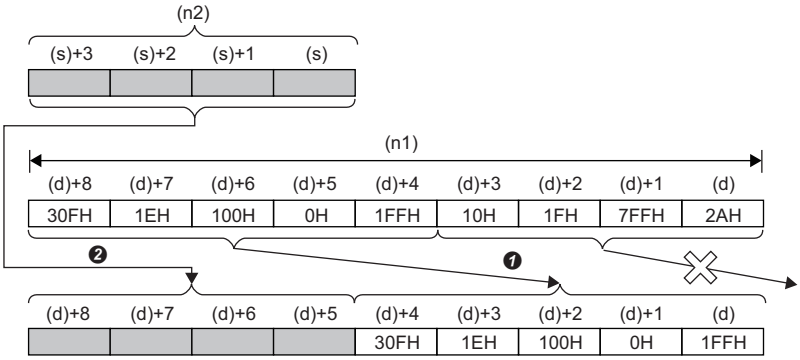
*1 FX and FY cannot be used.

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). In the empty area after the shift, (n2) point(s) of data starting from (s) is/are stored.
- When constant is specified for (s), the specified values are stored in (n2) points from the most significant bit after the shift.
- When (n2) is 0, the processing is not performed.

Ex.

When (n1)=9 and (n2)=4



❶ Shift to the right by (n2)-word

❷ Copy

Operation error

Error code (SD0)	Description
2821H	The range for (n2) points from (s) and that for (n1) points from (d) are overlapping.
3405H	The values specified in (n1) and (n2) are such that (n1)<(n2).

Precautions

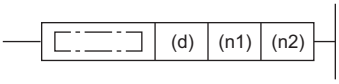
When specifying number of digits of bit for (s) and (d), set the same number of digits both for (s) and (d).

Shifting n-word data to the left by n word(s)

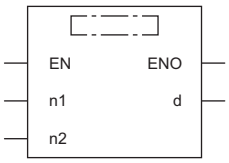
SFTWL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

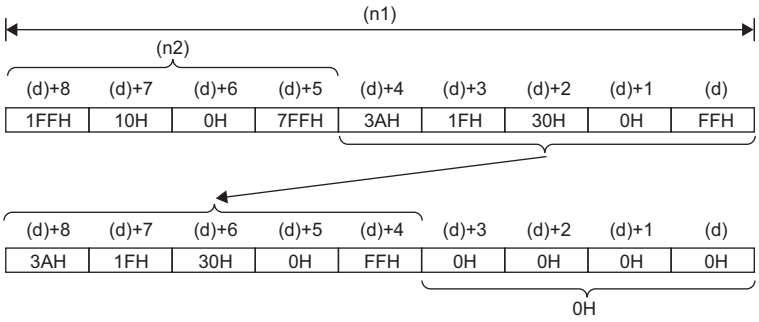
*1 FX and FY cannot be used.

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

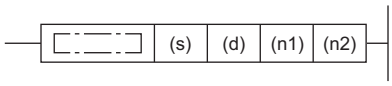
There is no operation error.

WSFL(P)

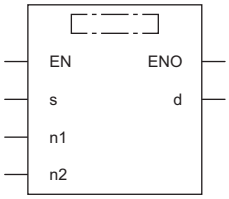
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions shift word data to the left by the (n2) word(s) within the (n1) words of data area starting from the specified device. In the empty area after the shift, specified data is stored.

Ladder	ST
	ENO:=WSFL(EN,s,n1,n2,d); ENO:=WSFLP(EN,s,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
WSFL	
WSFLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device stored in the empty area after the shift	—	16-bit unsigned binary	ANY16
(d)	Shift target start device	—	Word	ANY16
(n1)*1	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16_U
(n2)*1	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Set values so that (n2)≤(n1).

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(s)	○*1	—	○	—	—	—	—	○	○	—	—	—
(d)	○*1	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

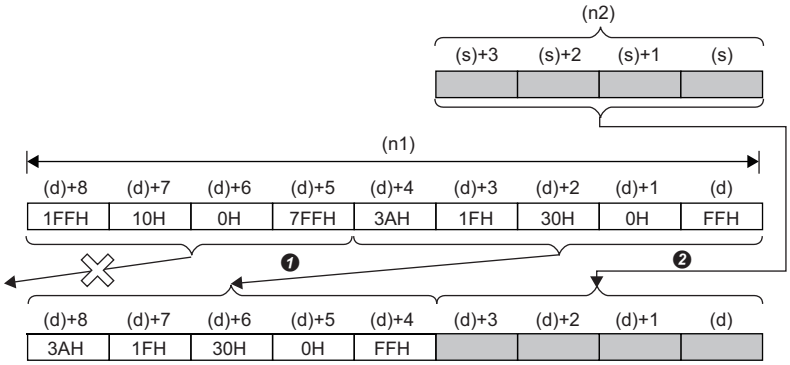
*1 FX and FY cannot be used.

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). In the empty area after the shift, (n2) point(s) of data starting from (s) is/are stored.
- When constant is specified for (s), the specified values are stored in (n2) points from the least significant bit after the shift.
- When (n2) is 0, the processing is not performed.

Ex.

When (n1)=9 and (n2)=4



❶ Shift to the left by (n2)-word

❷ Copy

Operation error

Error code (SD0)	Description
2821H	The range for (n2) points from (s) and that for (n1) points from (d) are overlapping.
3405H	The values specified in (n1) and (n2) are such that (n1)<(n2).

Precautions

When specifying number of digits of bit for (s) and (d), set the same number of digits both for (s) and (d).

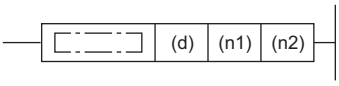
Shifting n double word(s) of data to the right by n double word(s)

SFTDWR(P)

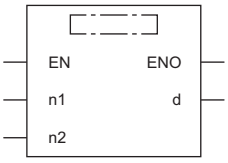
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift double word(s) of data to the right by the (n2) double word(s) within the (n1) double words of data area starting from the specified device. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=SFTDWR(EN,n1,n2,d); ENO:=SFTDWRP(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTDWR	
SFTDWRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target start device	—	Double word	ANY32
(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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

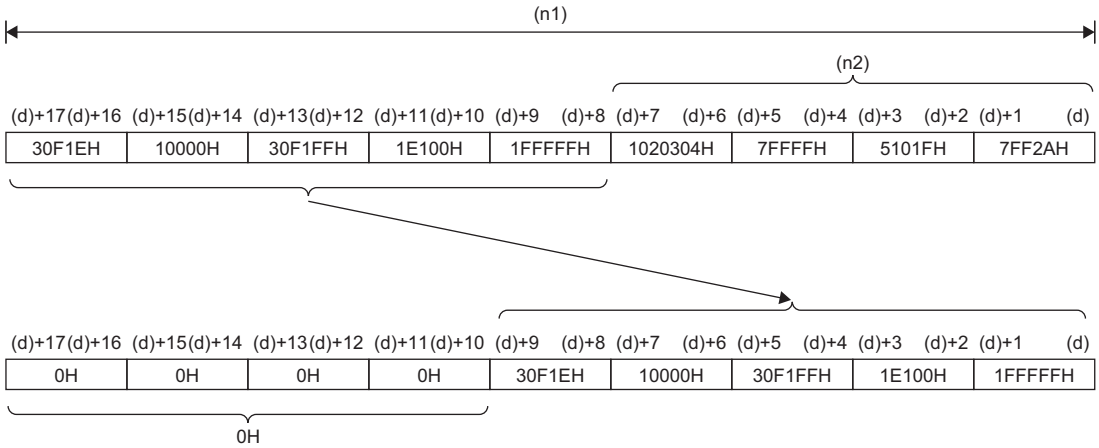
*1 FX and FY cannot be used.

Processing details

- These instructions shift double word(s) of data to the right by the (n2) double word(s) within the (n1) double words of data area starting from the device specified by (d).
- The (n2) double word(s) from the most significant bit is/are filled with 0H(s).
- If (n1) or (n2) is 0, no processing is performed.
- In the case of (n1)≤(n2), (n1) double words of data starting from the device specified by (d) become all 0Hs.

Ex.

When (n1)=9 and (n2)=4



Operation error

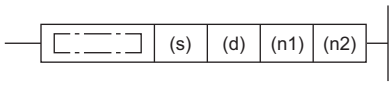
There is no operation error.

DWSFTR(P)

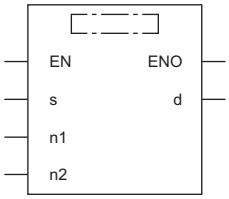
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift double word(s) of data to the right by the (n2) double word(s) within the (n1) double words of data area starting from the specified device. In the empty area after the shift, specified data is stored.

Ladder	ST
	ENO:=DWSFTR(EN,s,n1,n2,d); ENO:=DWSFTRP(EN,s,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DWSFTR	
DWSFTRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device stored in the empty area after the shift	—	Double word	ANY32
(d)	Shift target start device	—	Double word	ANY32
(n1)*1	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16
(n2)*1	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16

*1 Set values so that (n2)≤(n1).

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(s)	○*1	—	○	—	—	—	—	○	○	—	—	—
(d)	○*1	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

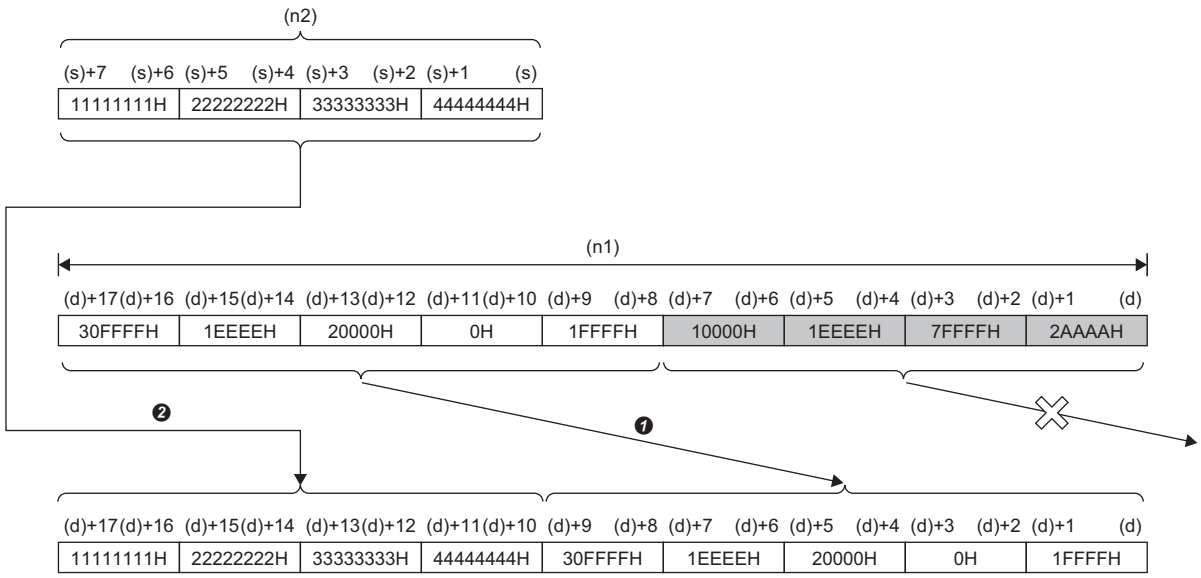
*1 FX and FY cannot be used.

Processing details

- These instructions shift double word(s) of data to the right by the (n2) double word(s) within the (n1) double words of data area starting from the device specified by (d). In the empty area after the shift, (n2) point(s) of data starting from (s) is/are stored.
- When constant is specified for (s), the specified values are stored in (n2) points from the most significant bit after the shift.
- When (n2) is 0, the processing is not performed.

Ex.

When (n1)=9 and (n2)=4



❶ Shift to the right by (n2)-double word

❷ Copy

Operation error

Error code (SD0)	Description
2821H	The range for (n2) points from (s) and that for (n1) points from (d) are overlapping.
3405H	The values specified in (n1) and (n2) are such that (n1)<(n2).

Precautions

When specifying number of digits of bit for (s) and (d), set the same number of digits both for (s) and (d).

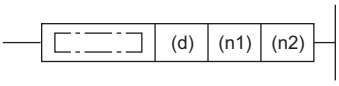
Shifting n double word(s) of data to the left by n double word(s)

SFTDWL(P)

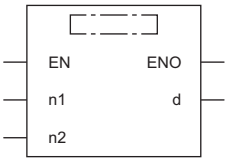
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift double word(s) of data to the left by the (n2) double word(s) within the (n1) double words of data area starting from the specified device. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=SFTDWL(EN,n1,n2,d); ENO:=SFTDWLP(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTDWL	
SFTDWLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target start device	—	Double word	ANY32
(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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

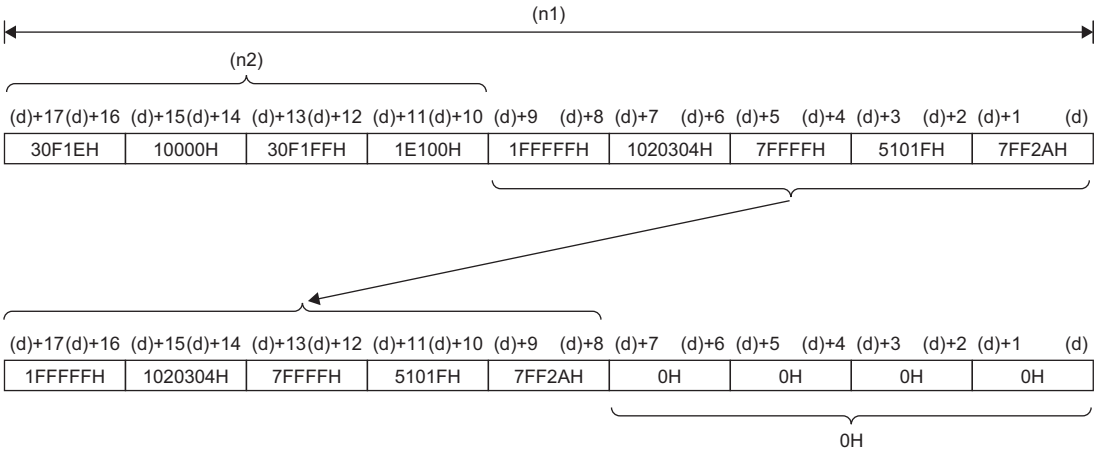
*1 FX and FY cannot be used.

Processing details

- These instructions shift double word(s) of data to the left by the (n2) double word(s) within the (n1) double words of data area starting from the device specified by (d).
- The (n2) double word(s) from the least significant bit is/are filled with 0H(s).
- If (n1) or (n2) is 0, no processing is performed.
- In the case of (n1)≤(n2), (n1) double words of data starting from the device specified by (d) become all 0Hs.

Ex.

When (n1)=9 and (n2)=4



Operation error

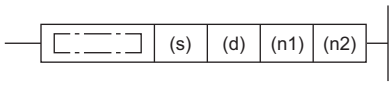
There is no operation error.

DWSFTL(P)

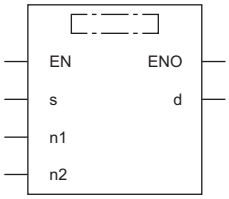
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift double word(s) of data to the left by the (n2) double word(s) within the (n1) double words of data area starting from the specified device. In the empty area after the shift, specified data is stored.

Ladder	ST
	ENO:=DWSFTL(EN,s,n1,n2,d); ENO:=DWSFTLP(EN,s,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DWSFTL	
DWSFTLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device stored in the empty area after the shift	—	Double word	ANY32
(d)	Shift target start device	—	Double word	ANY32
(n1)*1	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16
(n2)*1	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16

*1 Set values so that (n2)≤(n1).

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(s)	○*1	—	○	—	—	—	—	○	○	—	—	—
(d)	○*1	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

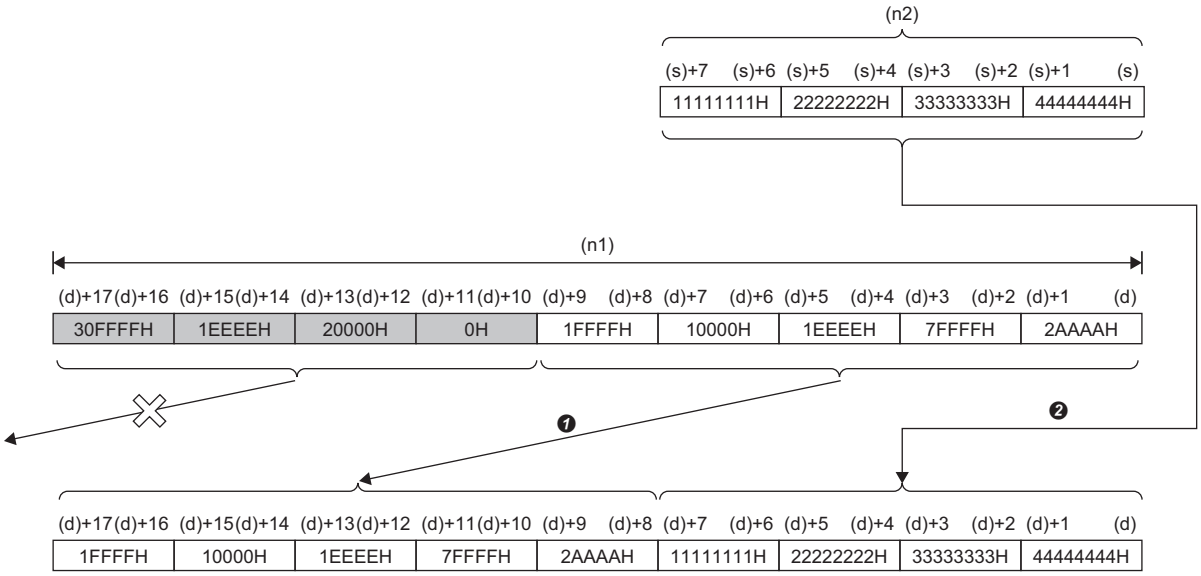
*1 FX and FY cannot be used.

Processing details

- These instructions shift double word(s) of data to the left by the (n2) double word(s) within the (n1) double words of data area starting from the device specified by (d). In the empty area after the shift, (n2) point(s) of data starting from (s) is/are stored.
- When constant is specified for (s), the specified values are stored in (n2) points from the least significant bit after the shift.
- When (n2) is 0, the processing is not performed.

Ex.

When (n1)=9 and (n2)=4



❶ Shift to the left by (n2)-double word

❷ Copy

Operation error

Error code (SD0)	Description
2821H	The range for (n2) points from (s) and that for (n1) points from (d) are overlapping.
3405H	The values specified in (n1) and (n2) are such that (n1)<(n2).

Precautions

When specifying number of digits of bit for (s) and (d), set the same number of digits both for (s) and (d).

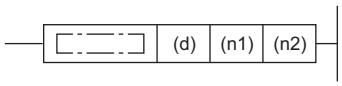
Shifting n point(s) of single-precision real number data to the right by n point(s)

SFTER(P)

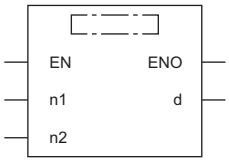
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift point(s) of single-precision real number data to the right by the (n2) point(s) within the single-precision real number (n1) point(s) of data area starting from the specified device. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=SFTER(EN,n1,n2,d); ENO:=SFTERP(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTER	
SFTERP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target start device	—	Single-precision real number	ANYREAL_32
(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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

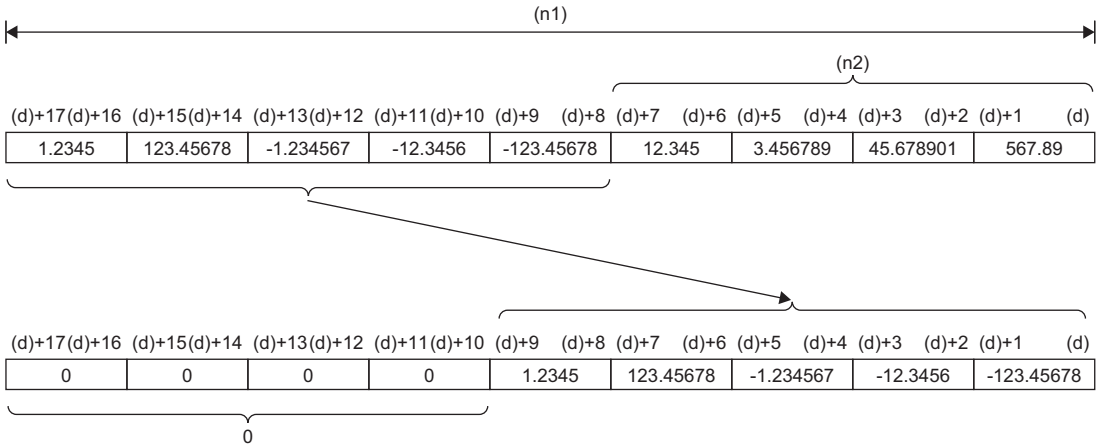
*1 FX and FY cannot be used.

Processing details

- These instructions shift point(s) of single-precision real number data to the right by the (n2) point(s) within the single-precision real number (n1) point(s) of data area starting from the device specified by (d).
- The (n2) point(s) from the most significant bit is/are filled with 0(s).
- If (n1) or (n2) is 0, no processing is performed.
- In case of (n1)≤(n2), (n1) points of data starting from the device specified by (d) become all 0.

Ex.

When (n1)=9 and (n2)=4



Operation error

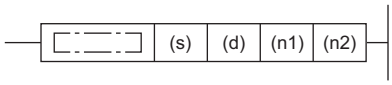
There is no operation error.

ESFTR(P)

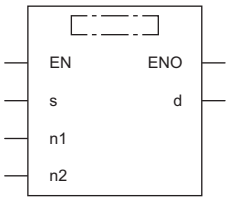
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift point(s) of single-precision real number data to the right by the (n2) point(s) within the single-precision real number (n1) point(s) of data area starting from the specified device. In the empty area after the shift, specified data is stored.

Ladder	ST
	ENO:=ESFTR(EN,s,n1,n2,d); ENO:=ESFTRP(EN,s,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ESFTR	
ESFTRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device stored in the empty area after the shift	—	Single-precision real number	ANYREAL_32
(d)	Shift target start device	—	Single-precision real number	ANYREAL_32
(n1)*1	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16
(n2)*1	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16

*1 Set values so that (n2)≤(n1).

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(s)	—	—	○	—	—	—	—	○	—	○	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

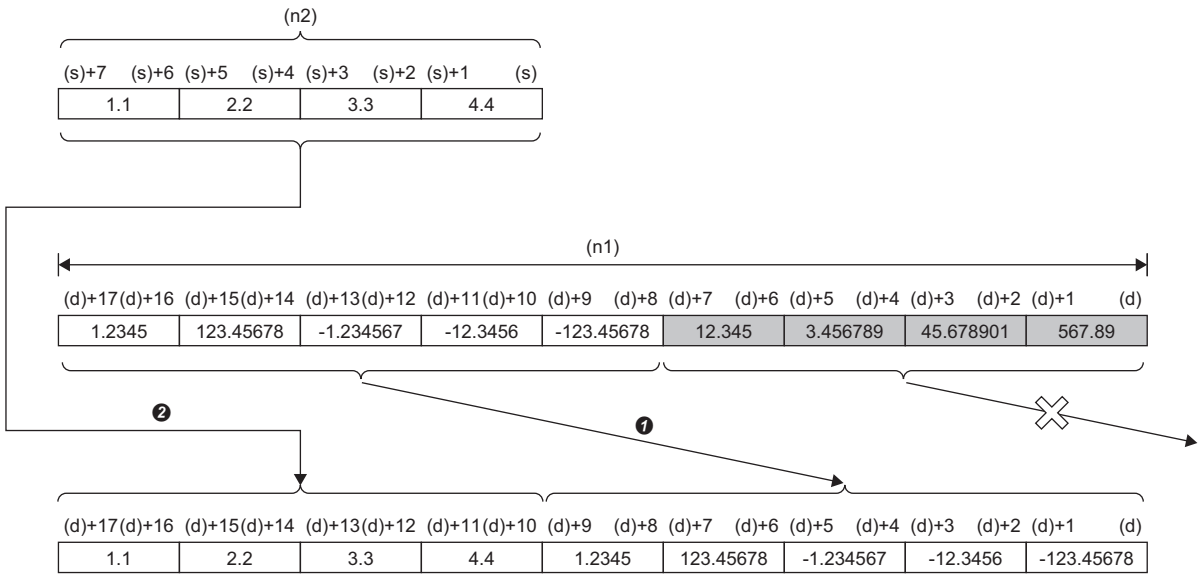
*1 FX and FY cannot be used.

Processing details

- These instructions shift point(s) of single-precision real number data to the right by the (n2) point(s) within the single-precision real number (n1) point(s) of data area starting from the device specified by (d). In the empty area after the shift, (n2) point(s) of data starting from (s) is/are stored.
- When constant is specified for (s), the specified values are stored in (n2) points from the most significant bit after the shift.
- When (n2) is 0, the processing is not performed.

Ex.

When (n1)=9 and (n2)=4



❶ Shift to the right by (n2)-point

❷ Copy

Operation error

Error code (SD0)	Description
2821H	The range for (n2) points from (s) and that for (n1) points from (d) are overlapping.
3405H	The values specified in (n1) and (n2) are such that (n1)<(n2).

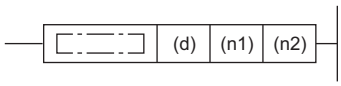
Shifting n point(s) of single-precision real number data to the left by n point(s)

SFTEL(P)

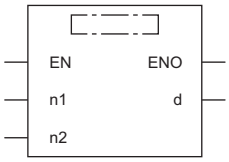
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift point(s) of single-precision real number data to the left by the (n2) point(s) within the single-precision real number (n1) point(s) of data area starting from the specified device. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=SFTEL(EN,n1,n2,d); ENO:=SFTELP(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTEL	
SFTELP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target start device	—	Single-precision real number	ANYREAL_32
(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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

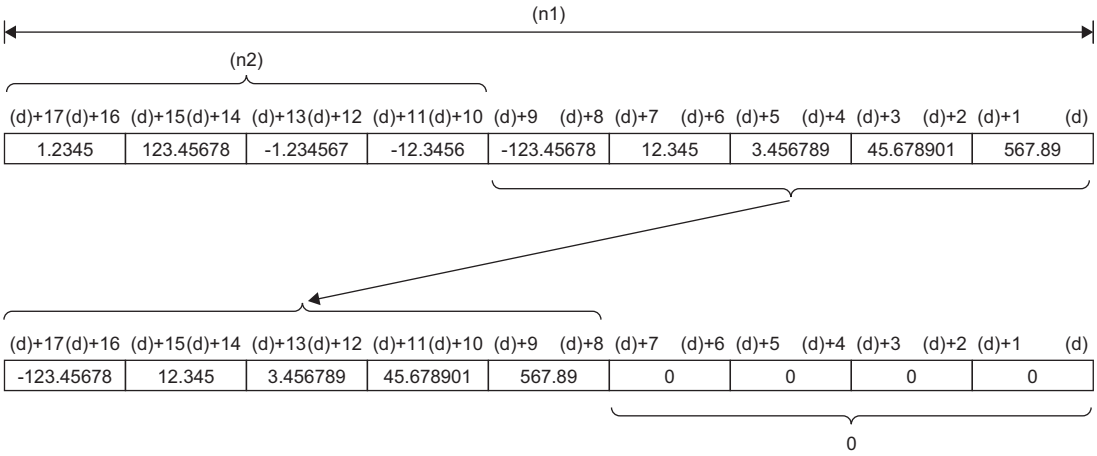
*1 FX and FY cannot be used.

Processing details

- These instructions shift point(s) of single-precision real number data to the left by the (n2) point(s) within the single-precision real number (n1) point(s) of data area starting from the device specified by (d).
- The (n2) point(s) from the least significant bit is/are filled with 0(s).
- If (n1) or (n2) is 0, no processing is performed.
- In case of (n1)≤(n2), (n1) points of data starting from the device specified by (d) become all 0.

Ex.

When (n1)=9 and (n2)=4



Operation error

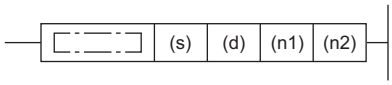
There is no operation error.

ESFTL(P)

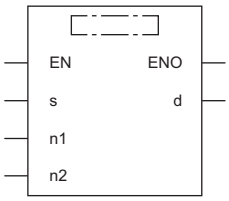
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift point(s) of single-precision real number data to the left by the (n2) point(s) within the single-precision real number (n1) point(s) of data area starting from the specified device. In the empty area after the shift, specified data is stored.

Ladder	ST
	ENO:=ESFTL(EN,s,n1,n2,d); ENO:=ESFTLP(EN,s,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ESFTL	
ESFTLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device stored in the empty area after the shift	—	Single-precision real number	ANYREAL_32
(d)	Shift target start device	—	Single-precision real number	ANYREAL_32
(n1)*1	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16
(n2)*1	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16

*1 Set values so that (n2)≤(n1).

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(s)	—	—	○	—	—	—	—	○	—	○	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

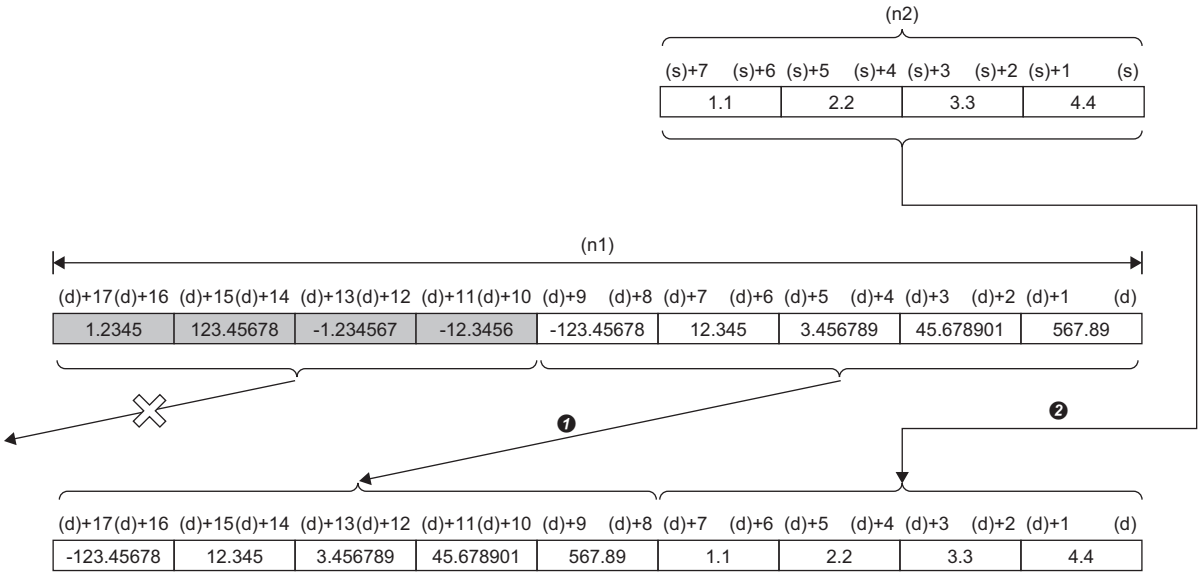
*1 FX and FY cannot be used.

Processing details

- These instructions shift point(s) of single-precision real number data to the left by the (n2) point(s) within the single-precision real number (n1) point(s) of data area starting from the device specified by (d). In the empty area after the shift, (n2) point(s) of data starting from (s) is/are stored.
- When constant is specified for (s), the specified values are stored in (n2) points from the least significant bit after the shift.
- When (n2) is 0, the processing is not performed.

Ex.

When (n1)=9 and (n2)=4



❶ Shift to the left by (n2)-point

❷ Copy

Operation error

Error code (SD0)	Description
2821H	The range for (n2) points from (s) and that for (n1) points from (d) are overlapping.
3405H	The values specified in (n1) and (n2) are such that (n1)<(n2).

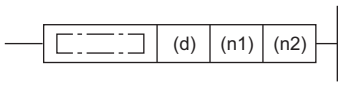
Shifting n point(s) of double-precision real number data to the right by n point(s)

SFTEDR(P)

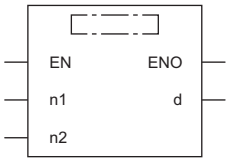
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift point(s) of double-precision real number data to the right by the (n2) point(s) within the double-precision real number (n1) point(s) of data area starting from the specified device. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=SFTEDR(EN,n1,n2,d); ENO:=SFTEDRP(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTEDR	
SFTEDRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target start device	—	Double-precision real number	ANYREAL_64
(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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

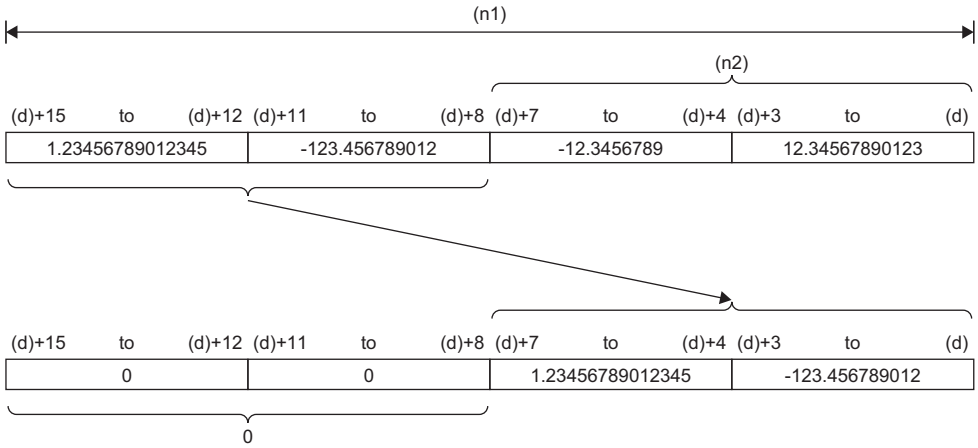
*1 FX and FY cannot be used.

Processing details

- These instructions shift point(s) of double-precision real number data to the right by the (n2) point(s) within the double-precision real number (n1) point(s) of data area starting from the device specified by (d).
- The (n2) point(s) from the most significant bit is/are filled with 0(s).
- If (n1) or (n2) is 0, no processing is performed.
- In case of (n1)≤(n2), (n1) points of data starting from the device specified by (d) become all 0.

Ex.

When (n1)=4 and (n2)=2



Operation error

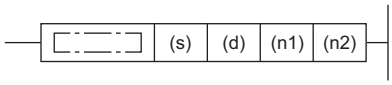
There is no operation error.

EDSFTR(P)

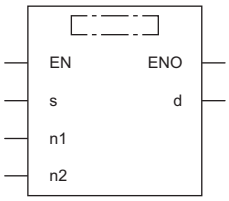
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift point(s) of double-precision real number data to the right by the (n2) point(s) within the double-precision real number (n1) point(s) of data area starting from the specified device. In the empty area after the shift, specified data is stored.

Ladder	ST
	ENO:=EDSFTR(EN,s,n1,n2,d); ENO:=EDSFTRP(EN,s,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EDSFTR	
EDSFTRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device stored in the empty area after the shift	—	Double-precision real number	ANYREAL_64
(d)	Shift target start device	—	Double-precision real number	ANYREAL_64
(n1)*1	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16
(n2)*1	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16

*1 Set values so that (n2)≤(n1).

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(s)	—	—	○	—	—	—	—	○	—	○	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

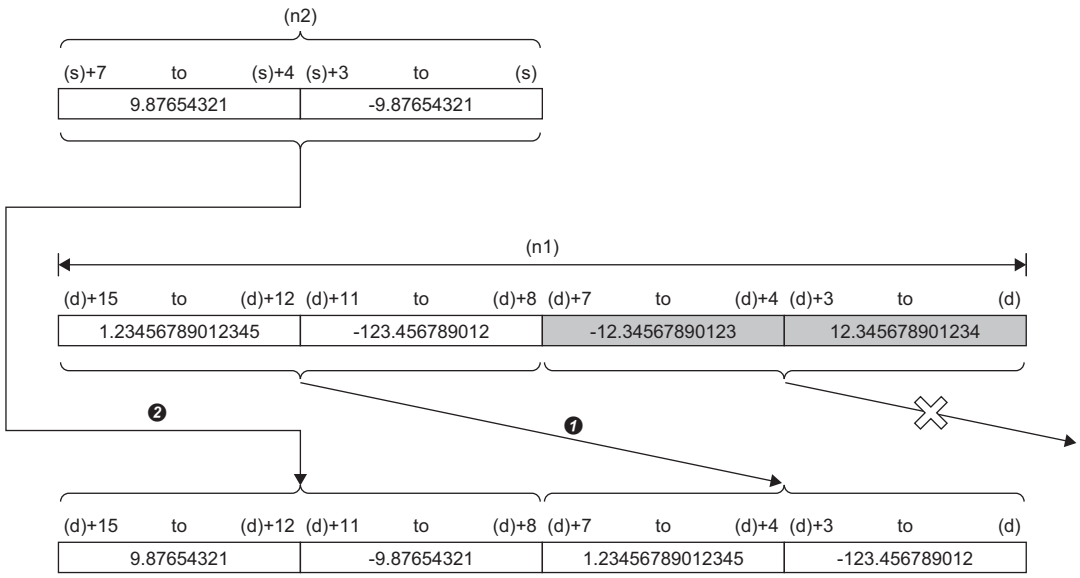
*1 FX and FY cannot be used.

Processing details

- These instructions shift point(s) of double-precision real number data to the right by the (n2) point(s) within the double-precision real number (n1) point(s) of data area starting from the device specified by (d). In the empty area after the shift, (n2) point(s) of data starting from (s) is/are stored.
- When constant is specified for (s), the specified values are stored in (n2) points from the most significant bit after the shift.
- When (n2) is 0, the processing is not performed.

Ex.

When (n1)=4 and (n2)=2



❶ Shift to the right by (n2)-point

❷ Copy

Operation error

Error code (SD0)	Description
2821H	The range for (n2) points from (s) and that for (n1) points from (d) are overlapping.
3405H	The values specified in (n1) and (n2) are such that (n1)<(n2).

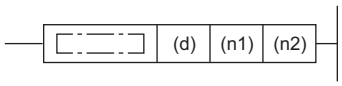
Shifting n point(s) of double-precision real number data to the left by n point(s)

SFTEDL(P)

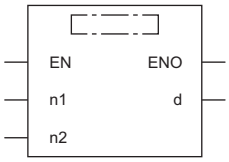
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift point(s) of double-precision real number data to the left by the (n2) point(s) within the double-precision real number (n1) point(s) of data area starting from the specified device. In the empty area after the shift, 0 is stored.

Ladder	ST
	ENO:=SFTEDL(EN,n1,n2,d); ENO:=SFTEDLP(EN,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SFTEDL	
SFTEDLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Shift target start device	—	Double-precision real number	ANYREAL_64
(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

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

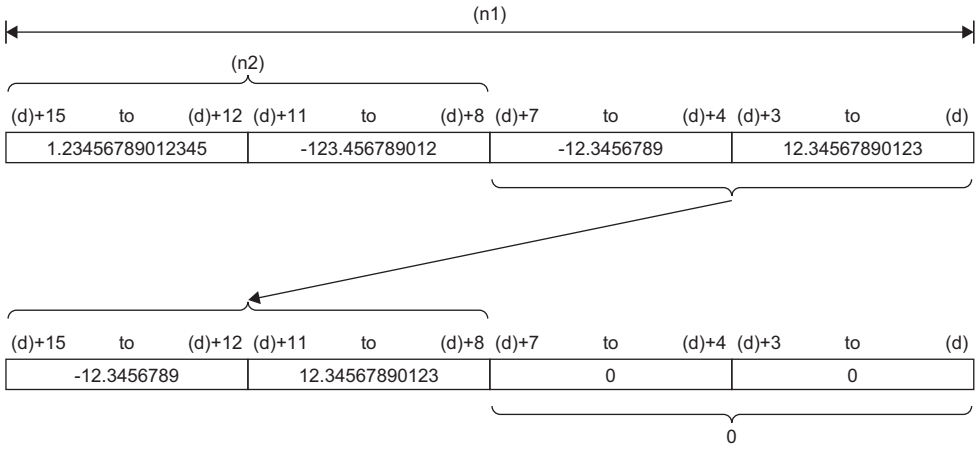
*1 FX and FY cannot be used.

Processing details

- These instructions shift point(s) of double-precision real number data to the left by the (n2) point(s) within the double-precision real number (n1) point(s) of data area starting from the device specified by (d).
- The (n2) point(s) from the least significant bit is/are filled with 0(s).
- If (n1) or (n2) is 0, no processing is performed.
- In case of (n1)≤(n2), (n1) points of data starting from the device specified by (d) become all 0.

Ex.

When (n1)=4 and (n2)=2



Operation error

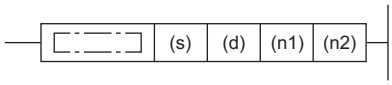
There is no operation error.

EDSFTL(P)

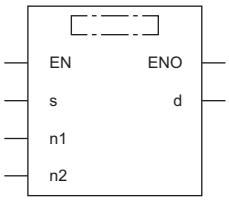
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.

These instructions shift point(s) of double-precision real number data to the left by the (n2) point(s) within the double-precision real number (n1) point(s) of data area starting from the specified device. In the empty area after the shift, specified data is stored.

Ladder	ST
	ENO:=EDSFTL(EN,s,n1,n2,d); ENO:=EDSFTLP(EN,s,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EDSFTL	
EDSFTLP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device stored in the empty area after the shift	—	Double-precision real number	ANYREAL_64
(d)	Shift target start device	—	Double-precision real number	ANYREAL_64
(n1)*1	Amount of data to be shifted	0 to 65535	16-bit unsigned binary	ANY16
(n2)*1	Number of shifts	0 to 65535	16-bit unsigned binary	ANY16

*1 Set values so that (n2)≤(n1).

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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	$	
(s)	—	—	○	—	—	—	—	○	—	○	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—

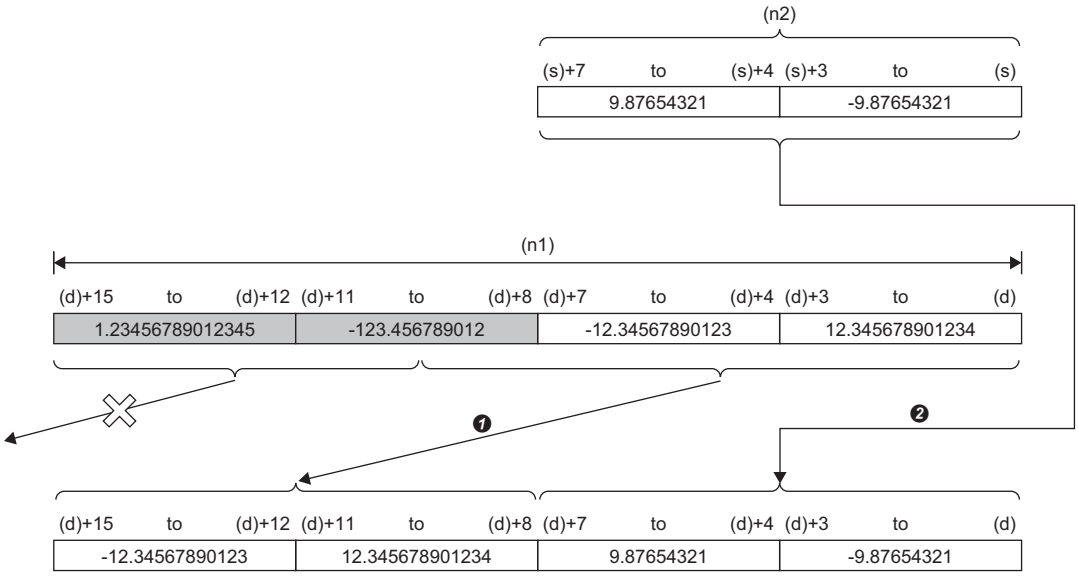
*1 FX and FY cannot be used.

Processing details

- These instructions shift point(s) of double-precision real number data to the left by the (n2) point(s) within the double-precision real number (n1) point(s) of data area starting from the device specified by (d). In the empty area after the shift, (n2) point(s) of data starting from (s) is/are stored.
- When constant is specified for (s), the specified values are stored in (n2) points from the least significant bit after the shift.
- When (n2) is 0, the processing is not performed.

Ex.

When (n1)=4 and (n2)=2



❶ Shift to the left by (n2)-point

❷ Copy

Operation error

Error code (SD0)	Description
2821H	The range for (n2) points from (s) and that for (n1) points from (d) are overlapping.
3405H	The values specified in (n1) and (n2) are such that (n1)<(n2).

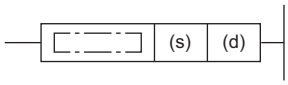
6.6 Data Conversion Instructions

Converting binary data to BCD 4-digit data

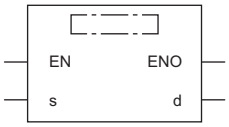
BCD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

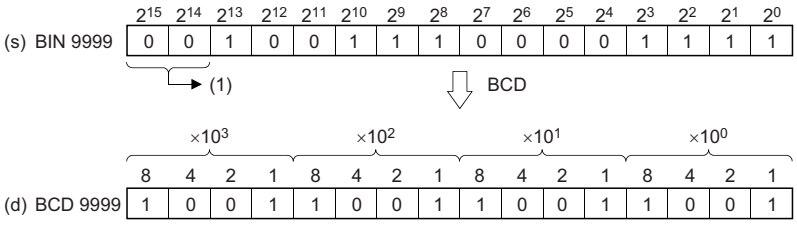
Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the device where the binary data is stored	0 to 9999	16-bit signed binary	ANY16
(d)	Device for storing the converted BCD data	—	BCD 4-digit	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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

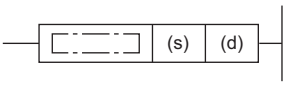
Error code (SD0)	Description
3401H	Data in the device specified by (s) is out of the range, 0 to 9999.

Converting binary data to BCD 8-digit data

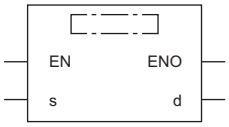
DBCD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

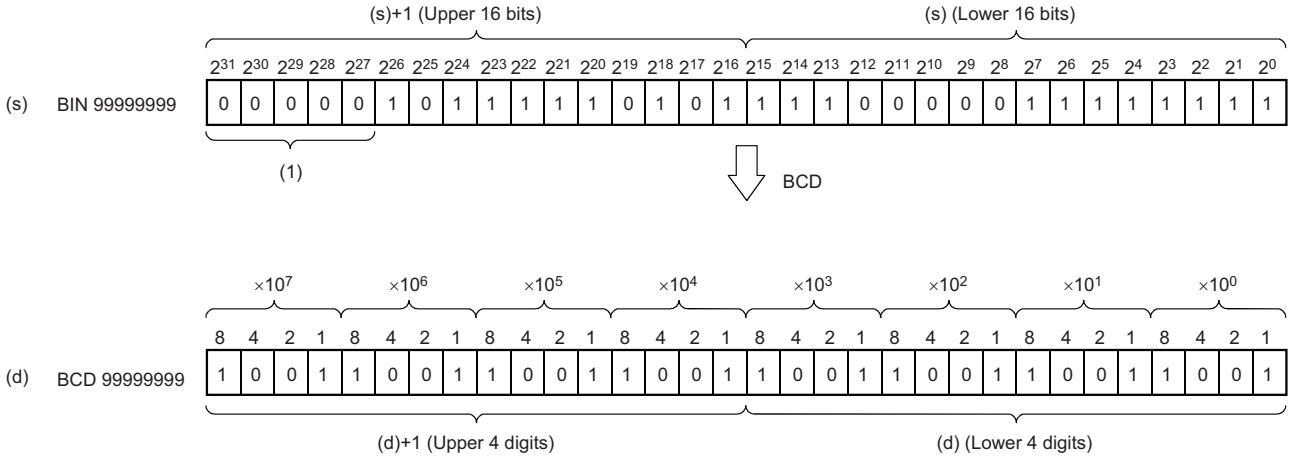
Operand	Description	Range	Data type	Data type (label)
(s)	Binary data or the start device where the binary data is stored	0 to 99999999	32-bit signed binary	ANY32
(d)	Start device for storing the converted BCD data	—	BCD 8-digit	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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 to the upper 5 bits.

Operation error

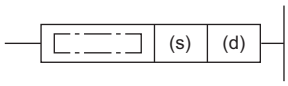
Error code (SD0)	Description
3401H	Data in the device specified by (s) is out of the range, 0 to 99999999.

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

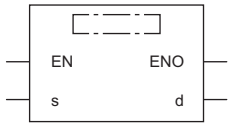
BIN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

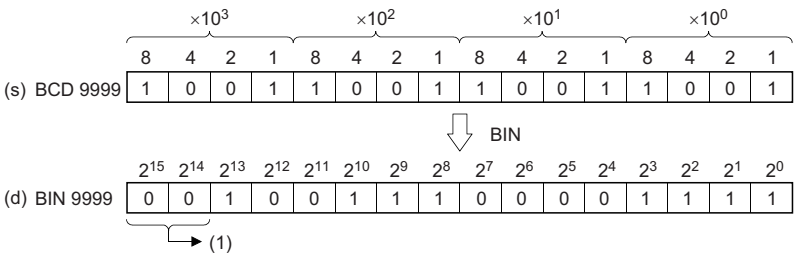
Operand	Description	Range	Data type	Data type (label)
(s)	BCD data or the device where the BCD data is stored	0 to 9999	BCD 4-digit	ANY16
(d)	Device for storing the converted binary data	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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
3401H	A value other than 0 to 9 exists at any digit of the value in the device specified by (s).*1

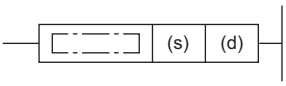
*1 Turning on SM754 can prevent this error from being detected.
If the specified value is out of the valid range, the BIN(P) instruction is not executed regardless of the status (on/off) of SM754.
The BIN(P) instruction does not execute the next operation until the command (execution condition) is turned off and on regardless of the presence of an error.

Converting BCD 8-digit data to 32-bit binary data

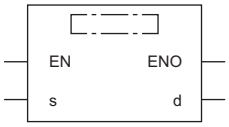
DBIN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

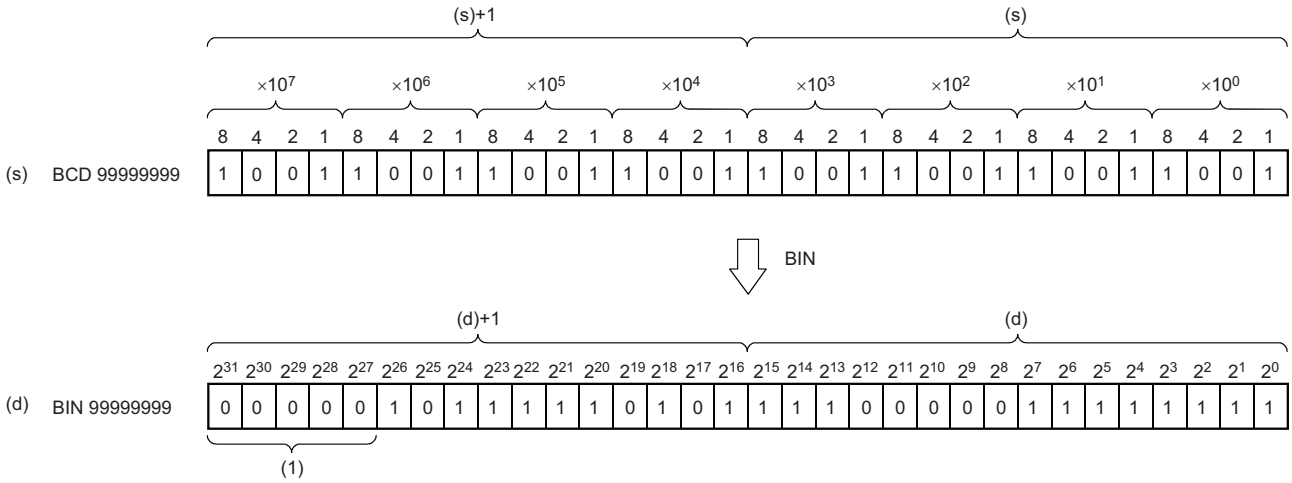
Operand	Description	Range	Data type	Data type (label)
(s)	BCD data or the start device where the BCD data is stored	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the converted binary data	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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
3401H	A value other than 0 to 9 exists at any digit of the value in the device specified by (s).*1

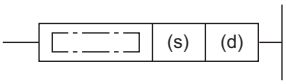
*1 Turning on SM754 can prevent this error from being detected.
If the specified value is out of the valid range, the DBIN(P) instruction is not executed regardless of the status (on/off) of SM754.
The DBIN(P) instruction does not execute the next operation until the command (execution condition) is turned off and on regardless of the presence of an error.

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

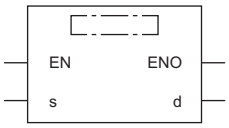
FLT2INT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified single-precision real number to 16-bit signed binary data.

Ladder	ST*1
	ENO:=FLT2INT(EN,s,d); ENO:=FLT2INTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

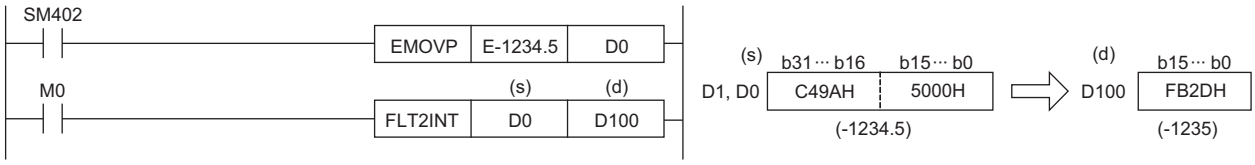
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

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 48 Precautions

The following program example converts, when M0 turns on, the single-precision real number stored in D0 and D1 to 16-bit signed binary data, and stores the converted data in D100.



Operation error

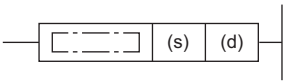
Error code (SD0)	Description
3401H	The single-precision real number in the device specified by (s) is out of the range, -32768 to 32767.
3402H	An unusual number is set to (s). • The single-precision real number set to (s) is not within the following range: $0, 2^{-126} \leq \lvert(s)\rvert < 2^{128}$ • 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

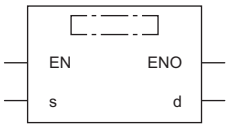
FLT2UINT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified single-precision real number to 16-bit unsigned binary data.

Ladder	ST*1
	ENO:=FLT2UINT(EN,s,d); ENO:=FLT2UINTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

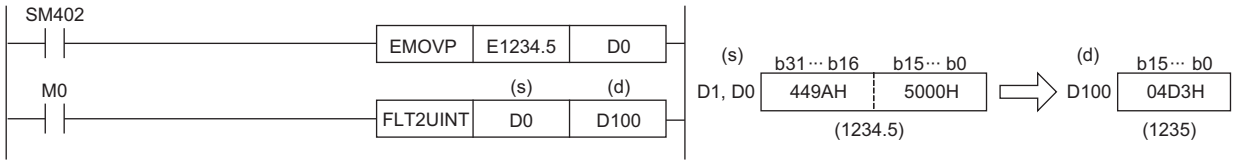
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

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 48 Precautions

The following program example converts, when M0 turns on, the single-precision real number stored in D0 and D1 to 16-bit unsigned binary data, and stores the converted data in D100.



Operation error

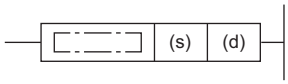
Error code (SD0)	Description
3401H	The single-precision real number in the device specified by (s) is out of the range, 0 to 65535.
3402H	An unusual number is set to (s). • The single-precision real number set to (s) is not within the following range: $0, 2^{-126} \leq \lvert(s)\rvert < 2^{128}$ • The 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

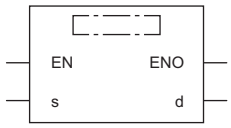
FLT2DINT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified single-precision real number to 32-bit signed binary data.

Ladder	ST*1
	ENO:=FLT2DINT(EN,s,d); ENO:=FLT2DINTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

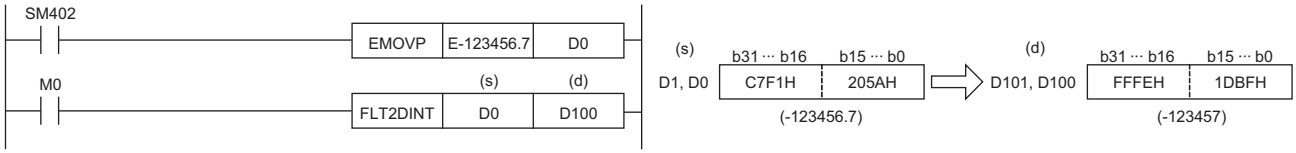
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

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 48 Precautions

The following program example converts, when M0 turns on, the single-precision real number stored in D0 and D1 to 32-bit signed binary data, and stores the converted data in D100 and D101.



Operation error

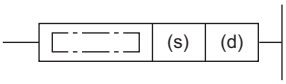
Error code (SD0)	Description
3401H	The single-precision real number in the device specified by (s) is out of the range, -2147483648 to 2147483647.
3402H	An unusual number is set to (s). • The single-precision real number set to (s) is not within the following range: $0, 2^{-126} \leq \lvert(s)\rvert < 2^{128}$ • 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

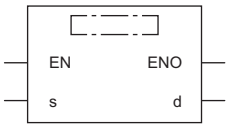
FLT2UDINT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified single-precision real number to 32-bit unsigned binary data.

Ladder	ST*1
	ENO:=FLT2UDINT(EN,s,d); ENO:=FLT2UDINTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

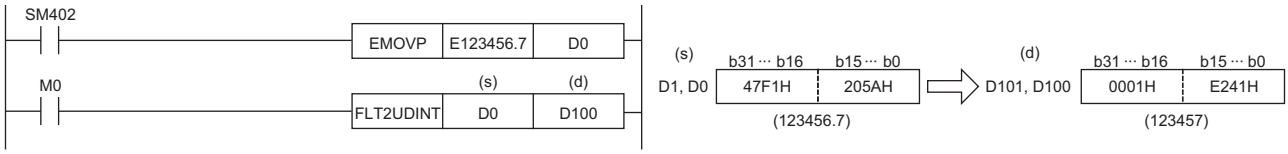
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	○	○	○	○	○	—	○	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

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 48 Precautions

The following program example converts, when M0 turns on, the single-precision real number stored in D0 and D1 to 32-bit unsigned binary data, and stores the converted data in D100 and D101.



Operation error

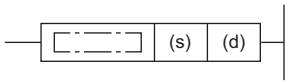
Error code (SD0)	Description
3401H	The single-precision real number in the device specified by (s) is out of the range, 0 to 4294967295.
3402H	An unusual number is set to (s). • The single-precision real number set to (s) is not within the following range: $0, 2^{-126} \leq \lvert(s)\rvert < 2^{128}$ • 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

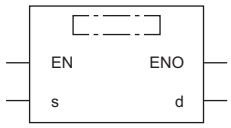
DBL2INT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified double-precision real number to 16-bit signed binary data.

Ladder	ST*1
	ENO:=DBL2INT(EN,s,d); ENO:=DBL2INTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

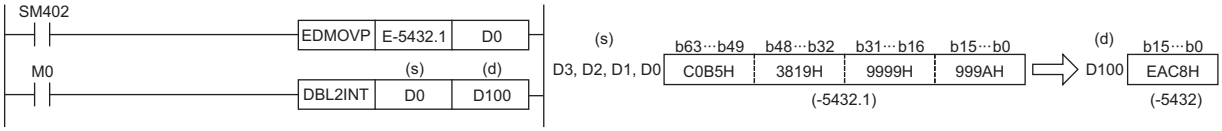
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○	—	○	—	—	○	—	—	—	—

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 48 Precautions

The following program example converts, when M0 turns on, the double-precision real number stored in D0 to D3 to 16-bit signed binary data, and stores the converted data in D100.



Operation error

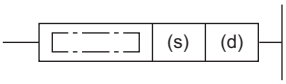
Error code (SD0)	Description
3401H	The double-precision real number in the device specified by (s) is out of the range, -32768 to 32767.
3402H	An unusual number is set to (s). • The double-precision real number set to (s) is not within the following range: $0, 2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$ • 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

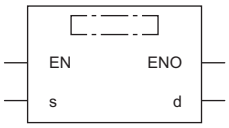
DBL2UINT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified double-precision real number to 16-bit unsigned binary data.

Ladder	ST*1
	ENO:=DBL2UINT(EN,s,d); ENO:=DBL2UINTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

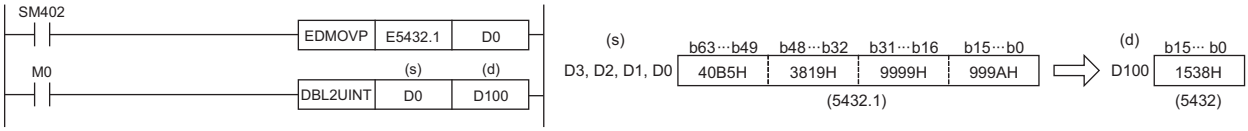
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○	—	○	—	—	○	—	—	—	—

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 48 Precautions

The following program example converts, when M0 turns on, the double-precision real number stored in D0 to D3 to 16-bit unsigned binary data, and stores the converted data in D100.



Operation error

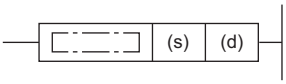
Error code (SD0)	Description
3401H	The double-precision real number in the device specified by (s) is out of the range, 0 to 65535.
3402H	An unusual number is set to (s). • The double-precision real number set to (s) is not within the following range: 0, $2^{-1022} \leq |(s)| < 2^{1024}$ • The value 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

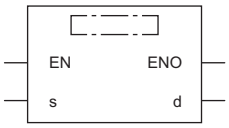
DBL2DINT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified double-precision real number to 32-bit signed binary data.

Ladder	ST*1
	ENO:=DBL2DINT(EN,s,d); ENO:=DBL2DINTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

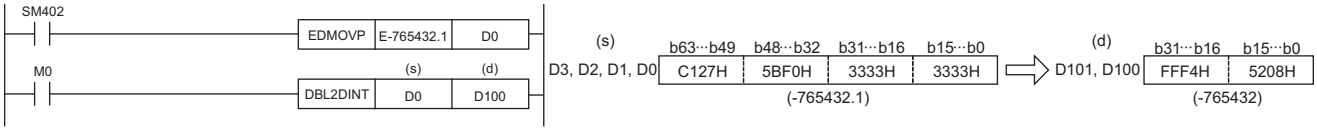
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○	—	○	○	○	○	—	—	—	—

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 48 Precautions

The following program example converts, when M0 turns on, the double-precision real number stored in D0 to D3 to 32-bit signed binary data, and stores the converted data in D100 and D101.



Operation error

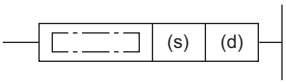
Error code (SD0)	Description
3401H	The double-precision real number in the device specified by (s) is out of the range, -2147483648 to 2147483647.
3402H	An unusual number is set to (s). • The double-precision real number set to (s) is not within the following range: $0, 2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

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

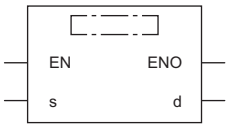
DBL2UDINT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified double-precision real number to 32-bit unsigned binary data.

Ladder	ST*1
	ENO:=DBL2UDINT(EN,s,d); ENO:=DBL2UDINTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

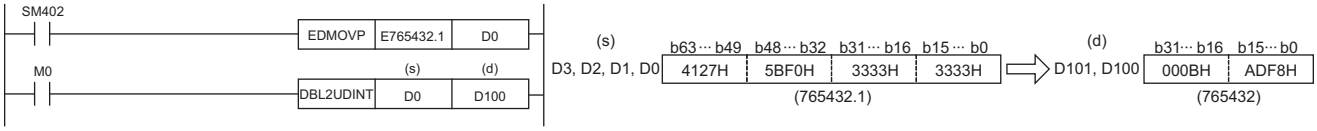
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○	—	○	○	○	○	—	—	—	—

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 48 Precautions

The following program example converts, when M0 turns on, the double-precision real number stored in D0 to D3 to 32-bit unsigned binary data, and stores the converted data in D100 and D101.



Operation error

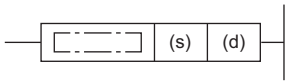
Error code (SD0)	Description
3401H	The double-precision real number in the device specified by (s) is out of the range, 0 to 4294967295.
3402H	An unusual number is set to (s). • The double-precision real number set to (s) is not within the following range: $0, 2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$ • The value set to a device or label is -0, a subnormal number, NaN (not a number), or ±∞.

Converting 16-bit signed binary data to 16-bit unsigned binary data

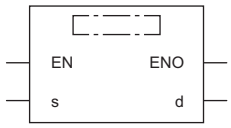
INT2UINT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified 16-bit signed binary data to 16-bit unsigned binary data.

Ladder	ST*1
	ENO:=INT2UINT(EN,s,d); ENO:=INT2UINTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Point

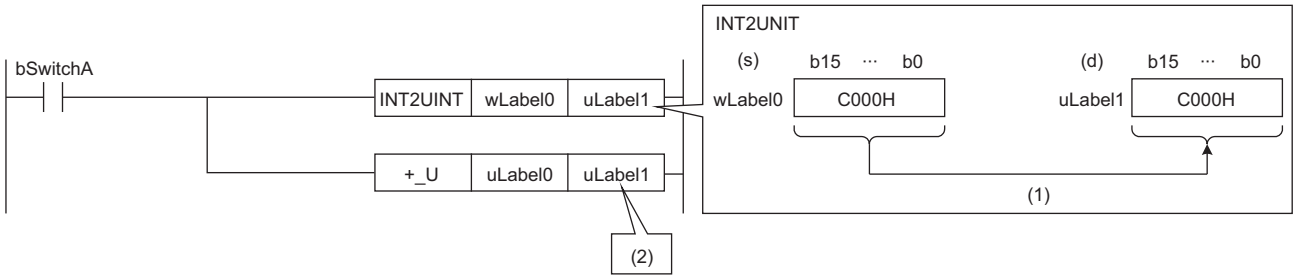
The INT2UINT(P) instruction is used in programming using labels. The purpose of using this instruction is to match the data type of the specified label with the data type that can be specified by the instruction operand. In programming using devices, use of the INT2UINT(P) instruction is not required.

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

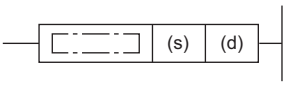
There is no operation error.

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

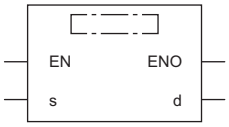
INT2DINT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified 16-bit signed binary data to 32-bit signed binary data.

Ladder	ST*1
	ENO:=INT2DINT(EN,s,d); ENO:=INT2DINTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

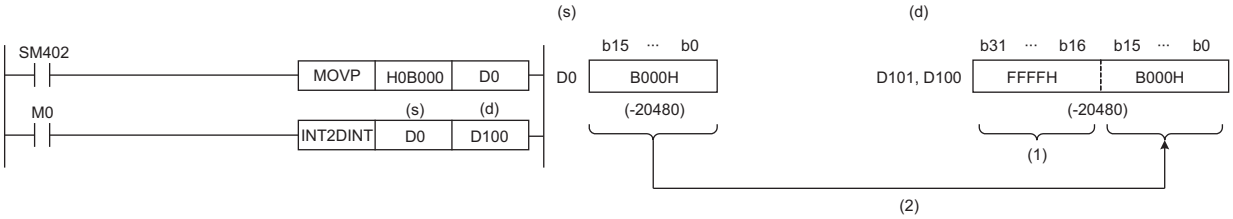
■Applicable devices

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□\□, U3E□\(H)G□	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).

The following program example converts, when M0 turns on, the 16-bit signed binary data stored in D0 to 32-bit signed binary data, and stores the converted data 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

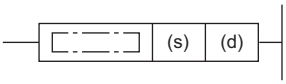
There is no operation error.

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

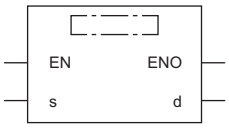
INT2UDINT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified 16-bit signed binary data to 32-bit unsigned binary data.

Ladder	ST*1
	ENO:=INT2UDINT(EN,s,d); ENO:=INT2UDINTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

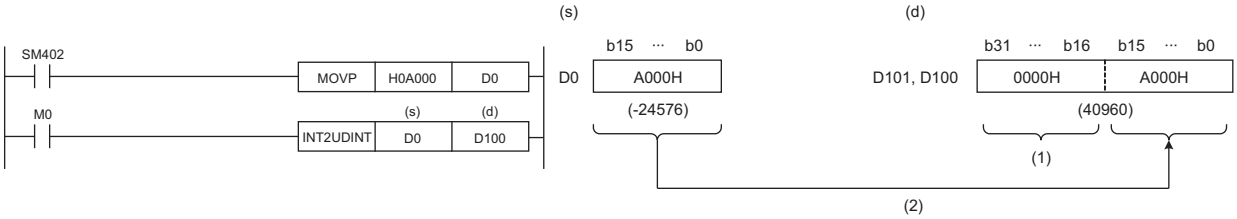
■Applicable devices

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□\□, U3E□\(H)G□	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).

The following program example converts, when M0 turns on, the 16-bit signed binary data stored in D0 to 32-bit unsigned binary data, and stores the converted data in D100 and D101.



(1) The value, 0, is stored.

(2) Data before conversion is stored in the lower 16 bits.

Operation error

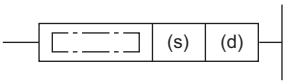
There is no operation error.

Converting 16-bit unsigned binary data to 16-bit signed binary data

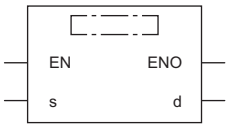
UINT2INT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified 16-bit unsigned binary data to 16-bit signed binary data.

Ladder	ST*1
	ENO:=UINT2INT(EN,s,d); ENO:=UINT2INTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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□\□, U3E□\(H)G□	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.

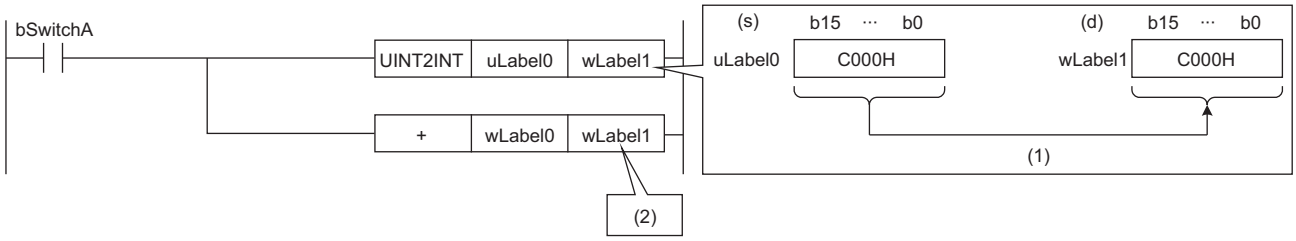
Processing details

- 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

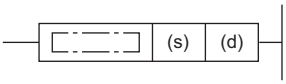
There is no operation error.

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

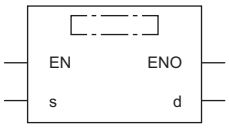
UINT2DINT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified 16-bit unsigned binary data to 32-bit signed binary data.

Ladder	ST*1
	ENO:=UINT2DINT(EN,s,d); ENO:=UINT2DINTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

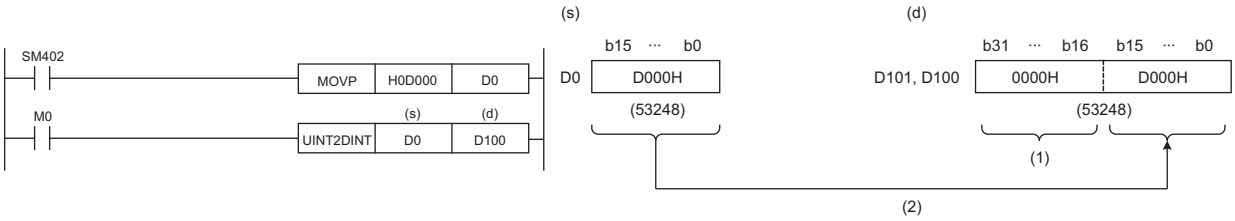
■Applicable devices

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□\□, U3E□\(H)G□	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).

The following program example converts, when M0 turns on, the 16-bit unsigned binary data stored in D0 to 32-bit signed binary data, and stores the converted data in D100 and D101.



(1) The value, 0, is stored.

(2) Data before conversion is stored in the lower 16 bits.

Operation error

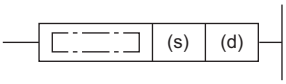
There is no operation error.

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

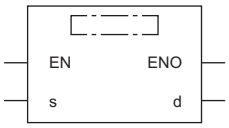
UINT2UDINT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified 16-bit unsigned binary data to 32-bit unsigned binary data.

Ladder	ST*1
	ENO:=UINT2UDINT(EN,s,d); ENO:=UINT2UDINTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

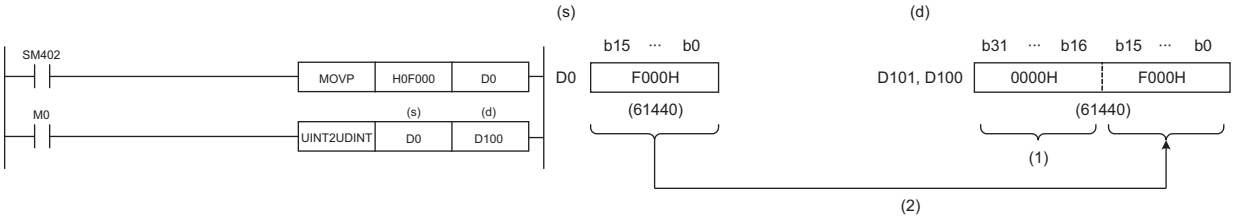
■Applicable devices

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□\□, U3E□\(H)G□	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).

The following program example converts, when M0 turns on, the 16-bit unsigned binary data stored in D0 to 32-bit unsigned binary data, and stores the converted data in D100 and D101.



(1) The value, 0, is stored.

(2) Data before conversion is stored in the lower 16 bits.

Operation error

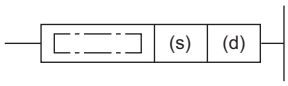
There is no operation error.

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

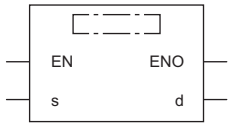
DINT2INT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified 32-bit signed binary data to 16-bit signed binary data.

Ladder	ST*1
	ENO:=DINT2INT(EN,s,d); ENO:=DINT2INTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

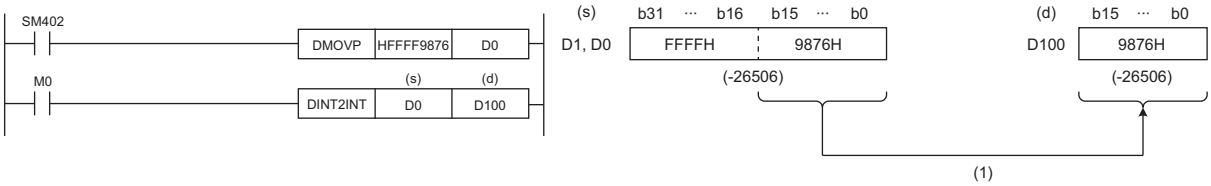
■Applicable devices

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□\□, U3E□\(H)G□	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).

The following program example converts, when M0 turns on, the 32-bit signed binary data stored in D0 and D1 to 16-bit signed binary data, and stores the converted data in D100.



(1) Data before conversion is stored in the lower 16 bits.

Operation error

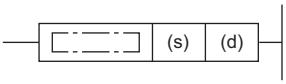
Error code (SD0)	Description
3401H	The 32-bit signed binary data in the device specified by (s) is out of the range, -32768 to 32767.

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

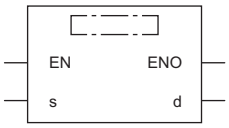
DINT2UINT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified 32-bit signed binary data to 16-bit unsigned binary data.

Ladder	ST*1
	ENO:=DINT2UINT(EN,s,d); ENO:=DINT2UINTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

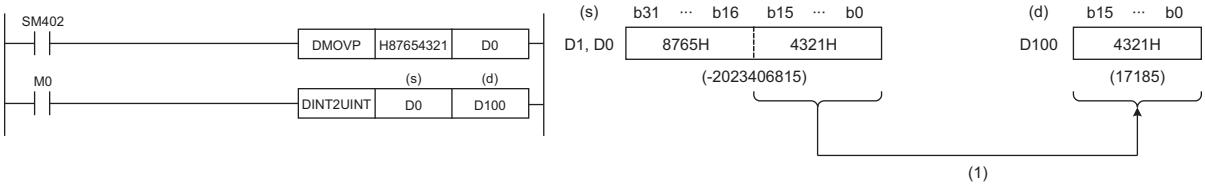
■Applicable devices

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□\□, U3E□\(H)G□	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).

The following program example converts, when M0 turns on, the 32-bit signed binary data stored in D0 and D1 to 16-bit unsigned binary data, and stores the converted data in D100.



(1) Data before conversion is stored in the lower 16 bits.

Operation error

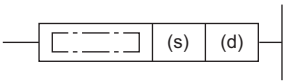
There is no operation error.

Converting 32-bit signed binary data to 32-bit unsigned binary data

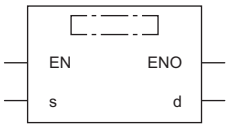
DINT2UDINT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified 32-bit signed binary data to 32-bit unsigned binary data.

Ladder	ST*1
	ENO:=DINT2UDINT(EN,s,d); ENO:=DINT2UDINTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Point

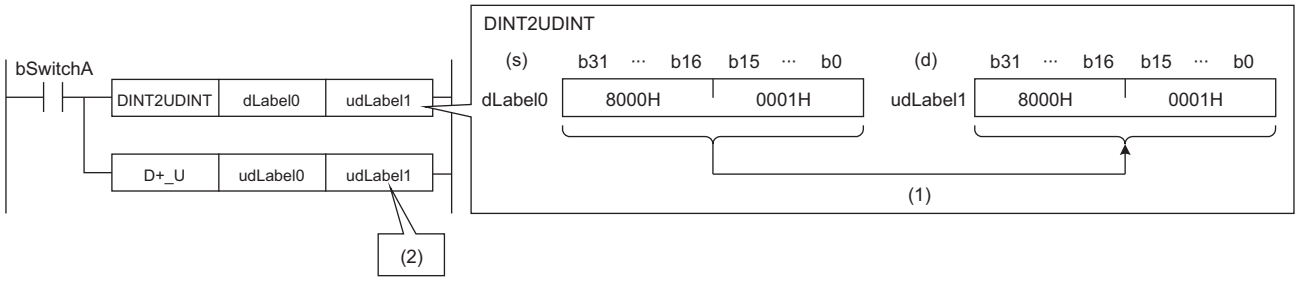
The DINT2UDINT(P) instruction is used in programming using labels. The purpose of using this instruction is to match the data type of the specified label with the data type that can be specified by the instruction operand.
In programming using devices, use of the DINT2UDINT(P) instruction is not required.

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

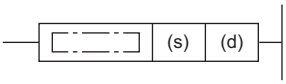
There is no operation error.

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

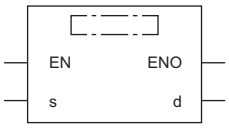
UDINT2INT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified 32-bit unsigned binary data to 16-bit signed binary data.

Ladder	ST*1
	ENO:=UDINT2INT(EN,s,d); ENO:=UDINT2INTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

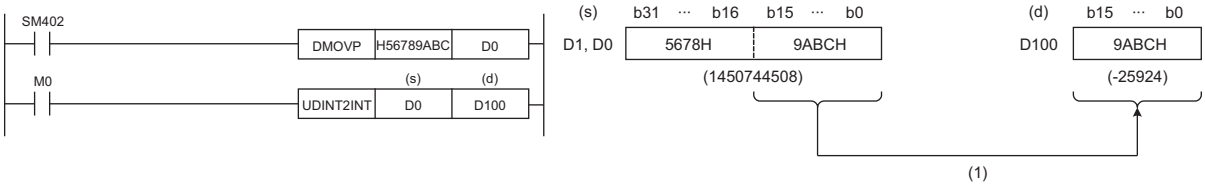
■Applicable devices

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□\□, U3E□\(H)G□	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).

The following program example converts, when M0 turns on, the 32-bit unsigned binary data stored in D0 and D1 to 16-bit signed binary data, and stores the converted data in D100.



(1) Data before conversion is stored in the lower 16 bits.

Operation error

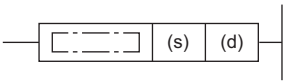
There is no operation error.

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

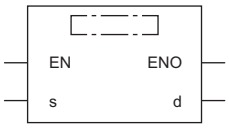
UDINT2UINT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified 32-bit unsigned binary data to 16-bit unsigned binary data.

Ladder	ST*1
	ENO:=UDINT2UINT(EN,s,d); ENO:=UDINT2UINTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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

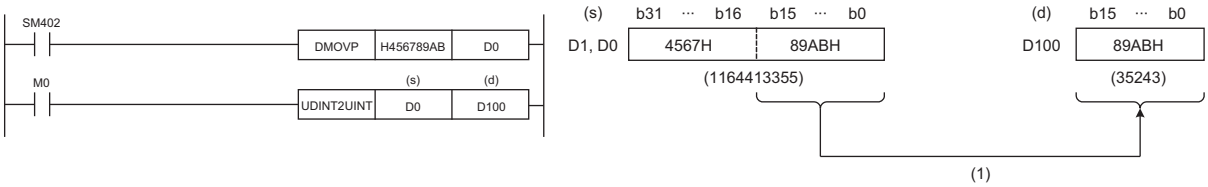
■Applicable devices

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□\□, U3E□\(H)G□	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).

The following program example converts, when M0 turns on, the 32-bit unsigned binary data stored in D0 and D1 to 16-bit unsigned binary data, and stores the converted data in D100.



(1) Data before conversion is stored in the lower 16 bits.

Operation error

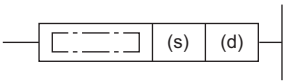
There is no operation error.

Converting 32-bit unsigned binary data to 32-bit signed binary data

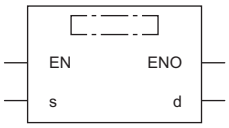
UDINT2DINT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert the specified 32-bit unsigned binary data to 32-bit signed binary data.

Ladder	ST*1
	ENO:=UDINT2DINT(EN,s,d); ENO:=UDINT2DINTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■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□\□, U3E□\(H)G□	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.

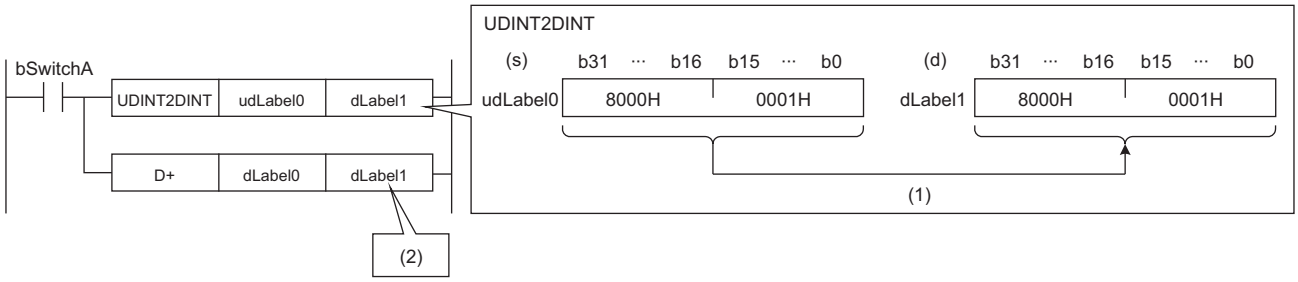
Processing details

- 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

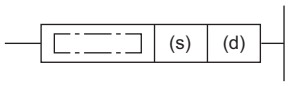
There is no operation error.

Converting 16-bit binary data to Gray code data

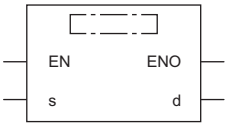
GRY(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

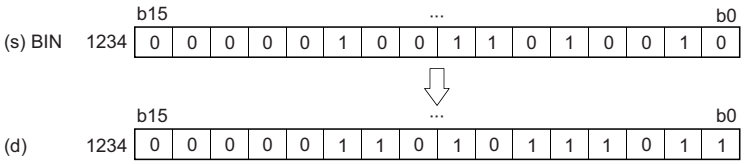
Operand		Description	Range	Data type	Data type (label)
(s)	GRY(P)	Binary data or the device where the binary data is stored	0 to 32767	16-bit signed binary	ANY16_S
	GRY(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	GRY(P)	Device for storing the converted Gray code data	—	16-bit signed binary	ANY16_S
	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□\□, U3E□\(H)G□	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

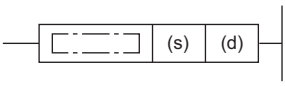
Error code (SD0)	Description
3401H	When the GRY(P) instruction is used, the value in the device specified by (s) is out of the range, 0 to 32767.

Converting 32-bit binary data to Gray code data

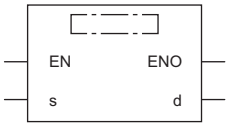
DGRY(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

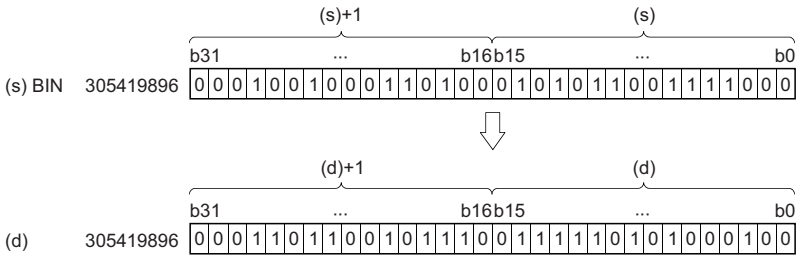
Operand		Description	Range	Data type	Data type (label)
(s)	DGRY(P)	Binary data or the start device where the binary data is stored	0 to 2147483647	32-bit signed binary	ANY32_S
	DGRY(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	DGRY(P)	Start device for storing the converted Gray code data	—	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□\□, U3E□\(H)G□	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

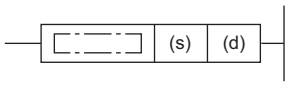
Error code (SD0)	Description
3401H	When the DGRY(P) instruction is used, the value in the device specified by (s) is out of the range, 0 to 2147483647.

Converting 16-bit binary Gray code data to 16-bit binary data

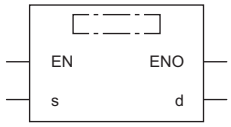
GBIN(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

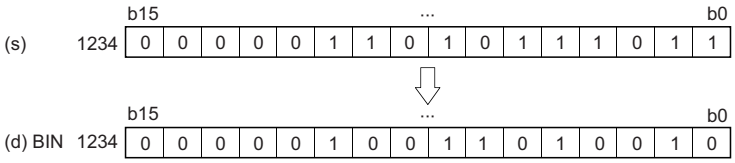
Operand		Description	Range	Data type	Data type (label)
(s)	GBIN(P)	Gray code data or the device where the Gray code data is stored	0 to 32767	16-bit signed binary	ANY16_S
	GBIN(P)_U		0 to 65535	16-bit unsigned binary	ANY16_U
(d)	GBIN(P)	Device for storing the converted binary data	—	16-bit signed binary	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□\□, U3E□\(H)G□	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

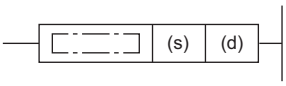
Error code (SD0)	Description
3401H	When the GBIN(P) instruction is used, the value in the device specified by (s) is out of the range, 0 to 32767.

Converting 32-bit binary Gray code data to 32-bit binary data

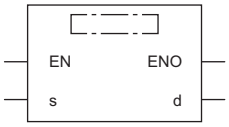
DGBIN(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

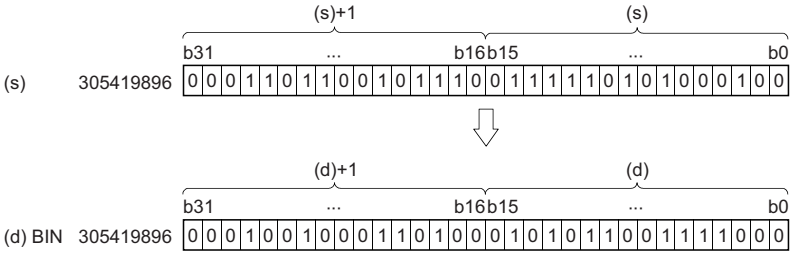
Operand		Description	Range	Data type	Data type (label)
(s)	DGBIN(P)	Gray code data or the start device where the Gray code data is stored	0 to 2147483647	32-bit signed binary	ANY32_S
	DGBIN(P)_U		0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	DGBIN(P)	Start device for storing the converted binary 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□\□, U3E□\(H)G□	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: Upper 16 bits

(s): Lower 16 bits

Operation error

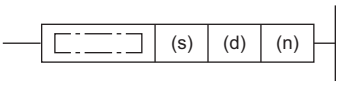
Error code (SD0)	Description
3401H	When the DGBIN(P) instruction is used, the value in the device specified by (s) is out of the range, 0 to 2147483647.

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

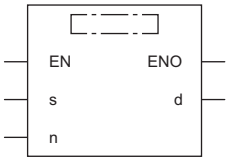
BKBCD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

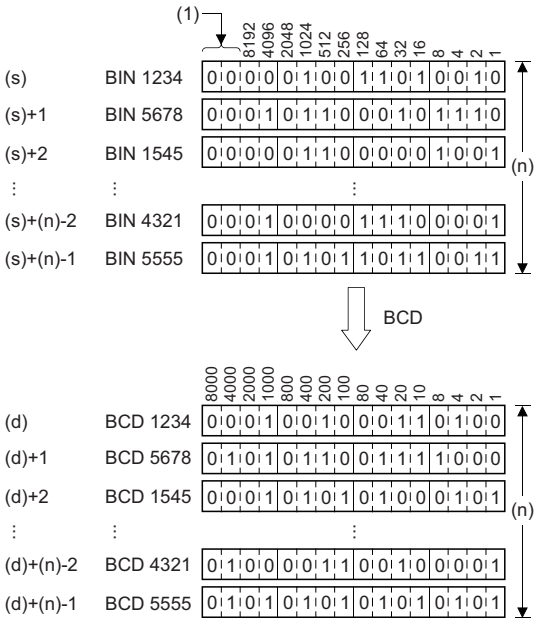
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

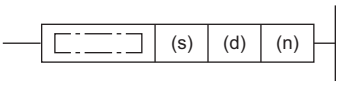
Error code (SD0)	Description
2821H	The device ranges specified by (s) and (d) are overlapping.
3401H	The (n) points of data starting from the device specified by (s) is out of the range, 0 to 9999.

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

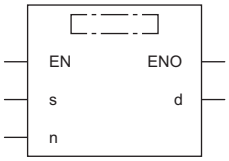
BKBIN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

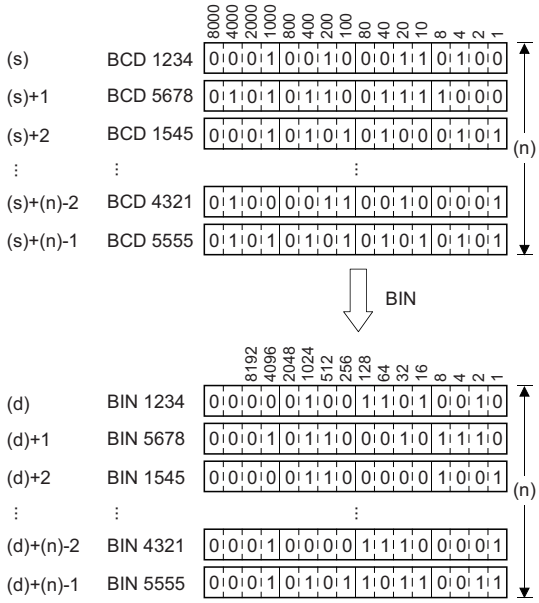
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

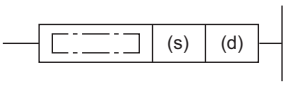
Error code (SD0)	Description
2821H	The device ranges specified by (s) and (d) are overlapping.
3401H	The (n) points of data starting from the device specified by (s) is out of the range, 0 to 9999.

Converting decimal ASCII data to 16-bit binary data

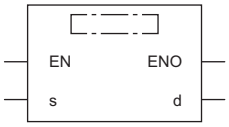
DABIN(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

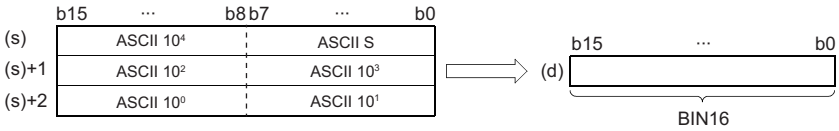
- These instructions convert the decimal ASCII data in the device specified by (s) and later to 16-bit binary data, and store the converted data in the device specified by (d).
- The setting method of the decimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705*1	Setting method of (s)	Reference
OFF	Set (s) with a fixed number of digits (a sign + 5 digits in the numeric part).	Page 470 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 471 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

*1 For the firmware version of the CPU module supporting SM705, refer to the following.
MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned 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.

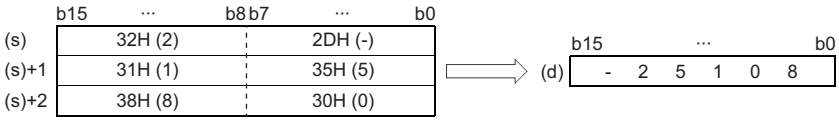


ASCII S: Sign data of ASCII code
ASCII 10^4: Ten-thousands place of ASCII code
ASCII 10^3: Thousands place of ASCII code
ASCII 10^2: Hundreds place of ASCII code
ASCII 10^1: Tens place of ASCII code
ASCII 10^0: Ones place of ASCII code

- 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).

Value to be set in (s)	Data of (s) to (s)+2	Value to be set in (s)	Data of (s) to (s)+2
• 0 • Positive value (1 digit in numeric part)	• Set 00H in the upper byte of (s). • The data of (s)+1 or later is ignored. b15 ··· b8 b7 ··· b0 (s): 00H / ASCII 10^0 (s)+1: (s)+2:	• Positive value (2 digits in numeric part) • Negative value (1 digit in numeric part)	• Set 00H in the lower byte of (s)+1. • The data of the upper byte of (s)+1 or later is ignored. b15 ··· b8 b7 ··· b0 (s): ASCII 10^0 / ASCII 10^1 / 2DH (-) (s)+1: 00H (s)+2:
⋮			
• Negative value (4 digits in numeric part)	• Set 00H in the upper byte of (s)+2. b15 ··· b8 b7 ··· b0 (s): ASCII 10^3 / 2DH (-) (s)+1: ASCII 10^1 / ASCII 10^2 (s)+2: 00H / ASCII 10^0	• Positive value (5 digits in numeric part)	• 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. b15 ··· b8 b7 ··· b0 (s): ASCII 10^3 / ASCII 10^4 (s)+1: ASCII 10^1 / ASCII 10^2 (s)+2: / ASCII 10^0
• Negative value (5 digits in numeric part)	• The data of (s)+3 or later is ignored. Since the number of digits is the maximum, 00H is not required to be set. b15 ··· b8 b7 ··· b0 (s): ASCII 10^4 / 2DH (-) (s)+1: ASCII 10^2 / ASCII 10^3 (s)+2: ASCII 10^0 / ASCII 10^1	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	

- 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
3401H	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.

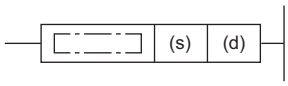
*1 When SM705 (Number of conversion digits selection) is off, no error is detected no matter what value is set for the ASCII code of the first character.

Converting decimal ASCII data to 32-bit binary data

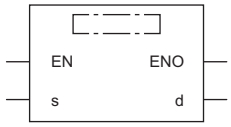
DDABIN(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

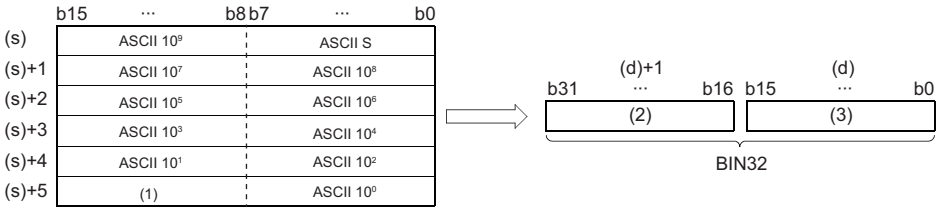
- These instructions convert the decimal ASCII data in the device areas specified by (s) and later to 32-bit binary data, and store the converted data in the device specified by (d).
- The setting method of the decimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705*1	Setting method of (s)	Reference
OFF	Set (s) with a fixed number of digits (a sign + 10 digits in the numeric part).	Page 473 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 474 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

*1 For the firmware version of the CPU module supporting SM705, refer to the following.
MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned 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.

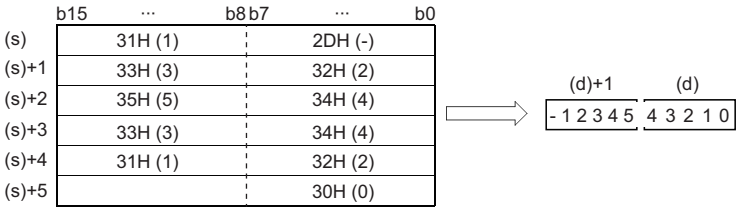


ASCII S: Sign data of ASCII code
ASCII 10^0: Ones place of ASCII code
ASCII 10^1: Tens place of ASCII code
ASCII 10^2: Hundreds place of ASCII code
ASCII 10^3: Thousands place of ASCII code
ASCII 10^4: Ten-thousands place of ASCII code
ASCII 10^5: Hundred-thousands place of ASCII code
ASCII 10^6: Millions 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).

Value to be set in (s)	Data of (s) to (s)+5	Value to be set in (s)	Data of (s) to (s)+5
• 0 • Positive value (1 digit in numeric part)	• Set 00H in the upper byte of (s). • The data of (s)+1 or later is ignored. b15 ··· b8 b7 ··· b0 (s): 00H / ASCII 10^0 (s)+1 (s)+2 (s)+3 (s)+4 (s)+5	• Positive value (2 digits in numeric part) • Negative value (1 digit in numeric part)	• Set 00H in the lower byte of (s)+1. • The data of the upper byte of (s)+1 or later is ignored. b15 ··· b8 b7 ··· b0 (s): ASCII 10^0 / ASCII 10^1 / 2DH (-) (s)+1: 00H (s)+2 (s)+3 (s)+4 (s)+5
⋮			
• Positive value (9 digits in numeric part) • Negative value (8 digits in numeric part)	• Set 00H in the upper byte of (s)+4. • (s)+5 is ignored. b15 ··· b8 b7 ··· b0 (s): ASCII 10^7 / ASCII 10^8 / 2DH (-) (s)+1: ASCII 10^5 / ASCII 10^6 (s)+2: ASCII 10^3 / ASCII 10^4 (s)+3: ASCII 10^1 / ASCII 10^2 (s)+4: 00H / ASCII 10^0 (s)+5	• Negative value (9 digits in numeric part)	• Set 00H in the lower byte of (s)+5. • The data of the upper byte of (s)+5 is ignored. b15 ··· b8 b7 ··· b0 (s): ASCII 10^8 / 2DH (-) (s)+1: ASCII 10^6 / ASCII 10^7 (s)+2: ASCII 10^4 / ASCII 10^5 (s)+3: ASCII 10^2 / ASCII 10^3 (s)+4: ASCII 10^0 / ASCII 10^1 (s)+5: 00H
• Positive value (10 digits in numeric part)	• (s)+5 is ignored. Since the number of digits is the maximum, 00H is not required to be set. b15 ··· b8 b7 ··· b0 (s): ASCII 10^8 / ASCII 10^9 (s)+1: ASCII 10^6 / ASCII 10^7 (s)+2: ASCII 10^4 / ASCII 10^5 (s)+3: ASCII 10^2 / ASCII 10^3 (s)+4: ASCII 10^0 / ASCII 10^1 (s)+5	• Negative value (10 digits in numeric part)	• 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. b15 ··· b8 b7 ··· b0 (s): ASCII 10^9 / 2DH (-) (s)+1: ASCII 10^7 / ASCII 10^8 (s)+2: ASCII 10^5 / ASCII 10^6 (s)+3: ASCII 10^3 / ASCII 10^4 (s)+4: ASCII 10^1 / ASCII 10^2 (s)+5: ASCII 10^0

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
3401H	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.

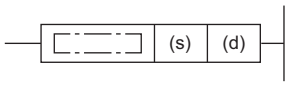
*1 When SM705 (Number of conversion digits selection) is off, no error is detected no matter what value is set for the ASCII code of the first character.

Converting hexadecimal ASCII data to 16-bit binary data

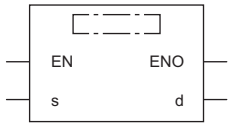
HABIN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

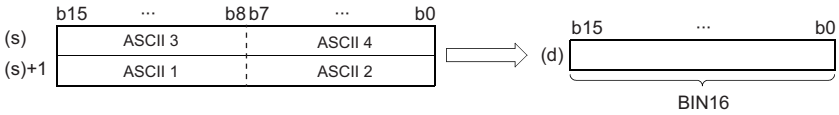
- These instructions convert the hexadecimal ASCII data stored in the device areas specified by (s) and later to 16-bit binary data, and store the converted data in the device specified by (d).
- The setting method of the hexadecimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705*1	Setting method of (s)	Reference
OFF	Set (s) with a fixed number of digits (4 digits).	Page 477 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 477 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

*1 For the firmware version of the CPU module supporting SM705, refer to the following.
MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned 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.

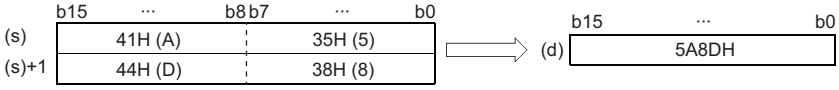


ASCII □: ASCII code (□th digit)

- 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.

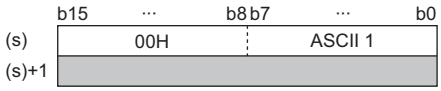
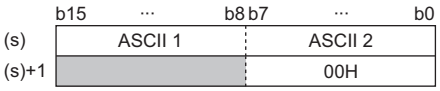
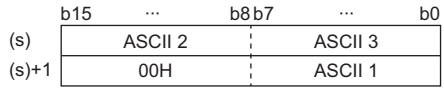
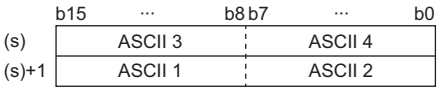
When 5A8DH is specified in (s)



■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

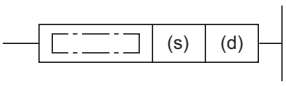
Error code (SD0)	Description
3401H	Invalid data that cannot be converted are input in (s) to (s)+1. • A value in each place of ASCII code is other than "30H" to "39H" and "41H" to "46H".

Converting hexadecimal ASCII data to 32-bit binary data

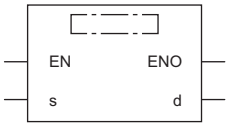
DHABIN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

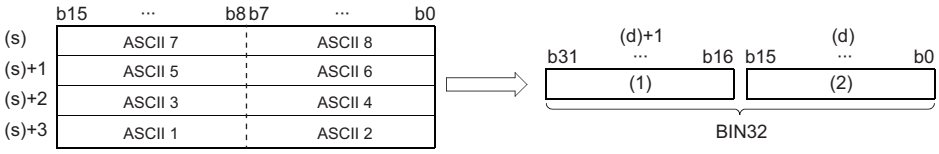
- These instructions convert the hexadecimal ASCII data stored in the device specified by (s) and later to 32-bit binary data, and store the converted data in the device specified by (d).
- The setting method of the hexadecimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705*1	Setting method of (s)	Reference
OFF	Set (s) with a fixed number of digits (8 digits).	Page 480 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 481 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

*1 For the firmware version of the CPU module supporting SM705, refer to the following.
MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Setting method of (s) for when SM705 (Number of conversion digits selection) is off

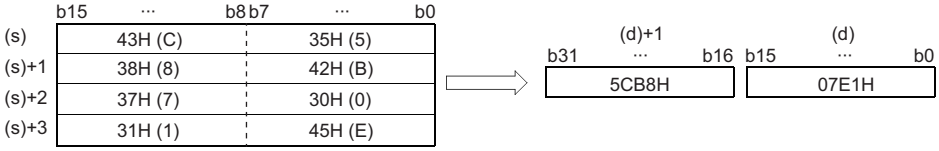
- Set hexadecimal ASCII data with 8 digits (fixed) in (s) to (s)+3.



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).

Value to be set in (s)	Data of (s) to (s)+3	Value to be set in (s)	Data of (s) to (s)+3
• 0H to FH	• Set 00H in the upper byte of (s). • The data of (s)+1 or later is ignored. b15 ··· b8 b7 ··· b0 (s): 00H | ASCII 1 (s)+1: (s)+2: (s)+3:	• 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. b15 ··· b8 b7 ··· b0 (s): ASCII 1 | ASCII 2 (s)+1: | 00H (s)+2: (s)+3:
⋮			
• 1000000H to FFFFFFFH	• Set 00H in the upper byte of (s)+3. • The data of (s)+4 or later is ignored. b15 ··· b8 b7 ··· b0 (s): ASCII 6 | ASCII 7 (s)+1: ASCII 4 | ASCII 5 (s)+2: ASCII 2 | ASCII 3 (s)+3: 00H | ASCII 1	• 10000000H to FFFFFFFFH	• The data of (s)+4 or later is ignored. b15 ··· b8 b7 ··· b0 (s): ASCII 7 | ASCII 8 (s)+1: ASCII 5 | ASCII 6 (s)+2: ASCII 3 | ASCII 4 (s)+3: ASCII 1 | ASCII 2

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

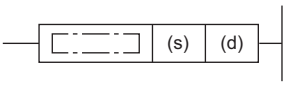
Error code (SD0)	Description
3401H	Invalid data that cannot be converted are input in (s) to (s)+3. • A value in each place of ASCII code is other than "30H" to "39H" and "41H" to "46H".

Converting decimal ASCII data to BCD 4-digit data

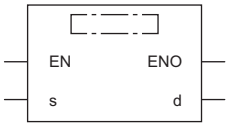
DABCD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

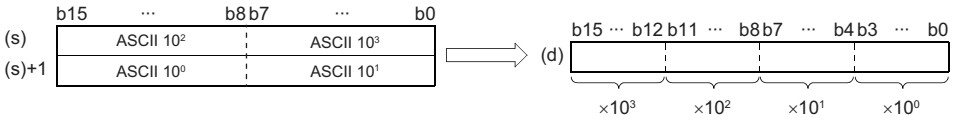
- These instructions convert the decimal ASCII data stored in the device areas specified by (s) and later to BCD 4-digit data, and store the converted data in the device specified by (d).
- The setting method of the decimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705*1	Setting method of (s)	Reference
OFF	Set (s) with a fixed number of digits (4 digits).	Page 483 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 484 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

*1 For the firmware version of the CPU module supporting SM705, refer to the following.
MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned 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.

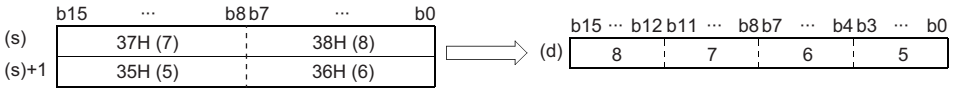


ASCII 10^3: Thousands place of ASCII code
ASCII 10^2: Hundreds place of ASCII code
ASCII 10^1: Tens place of ASCII code
ASCII 10^0: Ones place of ASCII code

- 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).

Value to be set in (s)	Data of (s) to (s)+1	Value to be set in (s)	Data of (s) to (s)+1
• 0 to 9	• Set 00H in the upper byte of (s)+0. • The data of (s)+1 or later is ignored. b15 ··· b8 b7 ··· b0 (s): 00H | ASCII 10^0 (s)+1:	• 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. b15 ··· b8 b7 ··· b0 (s): ASCII 10^0 | ASCII 10^1 (s)+1: | 00H
• 100 to 999	• Set 00H in the upper byte of (s)+1. • The data of (s)+2 or later is ignored. b15 ··· b8 b7 ··· b0 (s): ASCII 10^1 | ASCII 10^2 (s)+1: 00H | ASCII 10^0	• 1000 to 9999	• The data of (s)+2 or later is ignored. b15 ··· b8 b7 ··· b0 (s): ASCII 10^2 | ASCII 10^3 (s)+1: ASCII 10^0 | ASCII 10^1

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 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

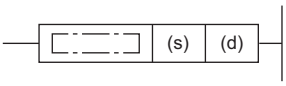
Error code (SD0)	Description
3401H	Invalid data that cannot be converted are input in (s). • A character other than 0 to 9 exists in the data.

Converting decimal ASCII data to BCD 8-digit data

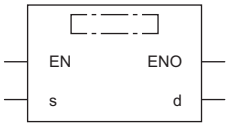
DDABCD(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

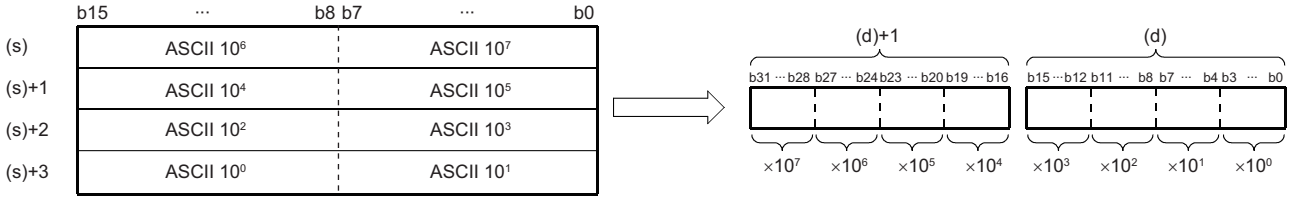
- These instructions convert the decimal ASCII data stored in the device areas specified by (s) and later to BCD 8-digit data, and store the converted data in the device number specified by (d).
- The setting method of the decimal ASCII data to be set in (s) depends on the status of SM705 (Number of conversion digits selection).

Status of SM705*1	Setting method of (s)	Reference
OFF	Set (s) with a fixed number of digits (8 digits).	Page 486 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 487 Setting method of (s) for when SM705 (Number of conversion digits selection) is on

*1 For the firmware version of the CPU module supporting SM705, refer to the following.
MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Setting method of (s) for when SM705 (Number of conversion digits selection) is off

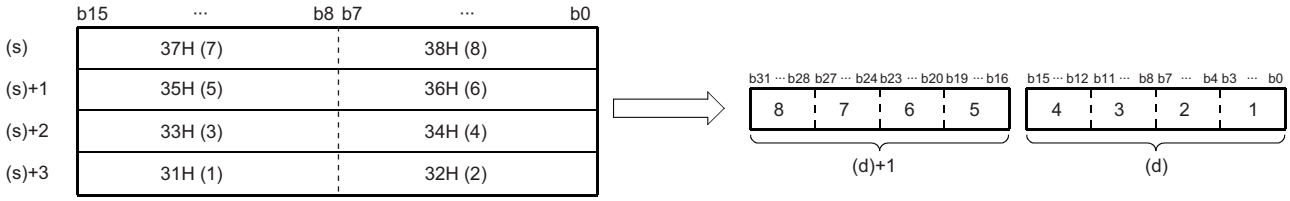
- Set decimal ASCII data with 8 digits (fixed) in (s) to (s)+3.



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. b15 ··· b8 b7 ··· b0 (s): 00H | ASCII 10^0 (s)+1: (s)+2: (s)+3:	• 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. b15 ··· b8 b7 ··· b0 (s): ASCII 10^0 | ASCII 10^1 (s)+1: | 00H (s)+2: (s)+3:
⋮			
• 1000000 to 9999999	• Set 00H in the upper byte of (s)+3. • The data of (s)+4 or later is ignored. b15 ··· b8 b7 ··· b0 (s): ASCII 10^5 | ASCII 10^6 (s)+1: ASCII 10^3 | ASCII 10^4 (s)+2: ASCII 10^1 | ASCII 10^2 (s)+3: 00H | ASCII 10^0	• 10000000 to 99999999	• The data of (s)+4 or later is ignored. b15 ··· b8 b7 ··· b0 (s): ASCII 10^6 | ASCII 10^7 (s)+1: ASCII 10^4 | ASCII 10^5 (s)+2: ASCII 10^2 | ASCII 10^3 (s)+3: ASCII 10^0 | ASCII 10^1

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

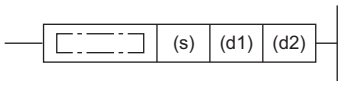
Error code (SD0)	Description
3401H	Invalid data that cannot be converted are input in (s). • A character other than 0 to 9 exists in the data.

Converting decimal string data to 16-bit binary data

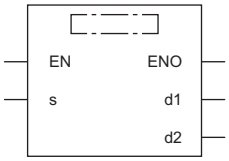
VAL(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert character 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

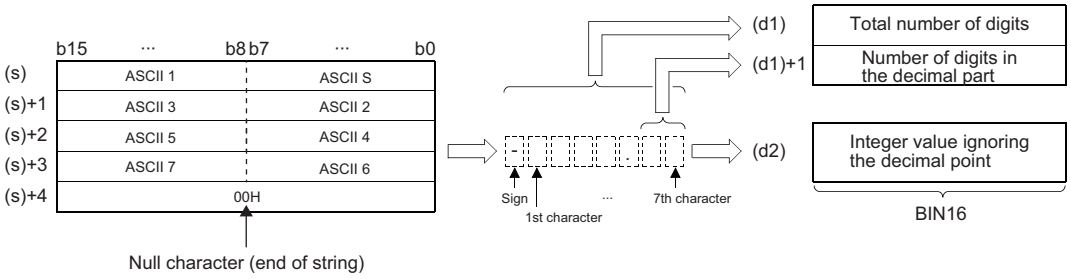
Operand		Description	Range	Data type	Data type (label)
(s)		String data to be converted to binary data or the start device where the string data is stored	—	String	ANYSTRING_SINGLE
(d1)	VAL(P)	Start device for storing the number of binary digits after conversion	—	16-bit signed binary	ANY16_S_ARRAY (Number 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

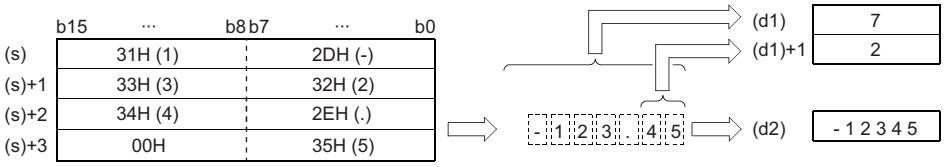
- These instructions convert the character strings stored in the device numbers specified by (s) and later to 16-bit binary data, and store the number of digits in (d1) and converted binary data in the device specified by (d2). For conversion of character strings to binary data, the data from the device number specified by (s) to the device number containing "00H" is processed as character strings.



ASCII S: ASCII code (sign data)
ASCII □: ASCII code (□th character)

Ex.

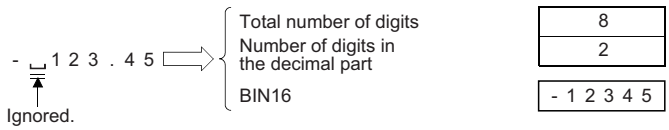
When a string "-123.45" (signed) is stored in the device specified by (s) or later



- The total number of characters of the character string stored in the device specified by (s) is 2 to 8.
- Of the character 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 character 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 character 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 character 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 character string in the device specified by (s), the instruction performs conversion to binary data by ignoring "20H" and "30H".

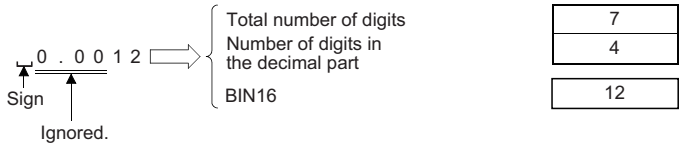
Ex.

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

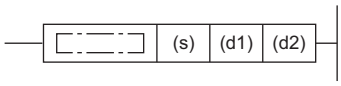
Error code (SD0)	Description
2820H	00H is not set between the device number specified by (s) and the last device number of the relevant device.
3401H	Invalid data that cannot be converted are input in (s). • The number of characters is not between 2 and 8. • The number of characters in the decimal part 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.
3405H	The number of characters of the character string in the device specified by (s) exceeds 16383.

Converting decimal string data to 32-bit binary data

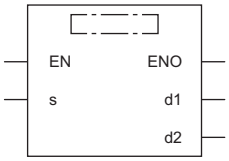
DVAL(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert character 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

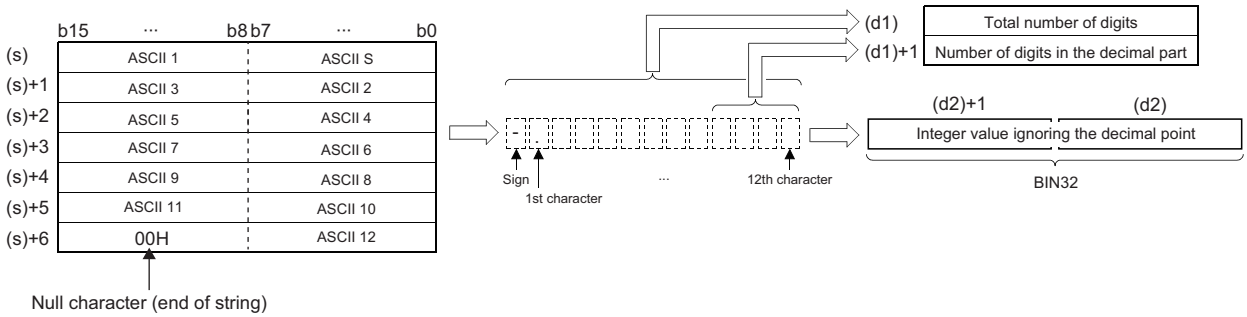
Operand		Description	Range	Data type	Data type (label)
(s)		String data to be converted to binary data or the start device where the string data is stored	—	String	ANYSTRING_SINGLE
(d1)	DVAL(P)	Start device for storing the number of binary digits after conversion	—	16-bit signed binary	ANY16_S_ARRAY (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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	○	○	○	○	○	○	○	—	—	—	—

Processing details

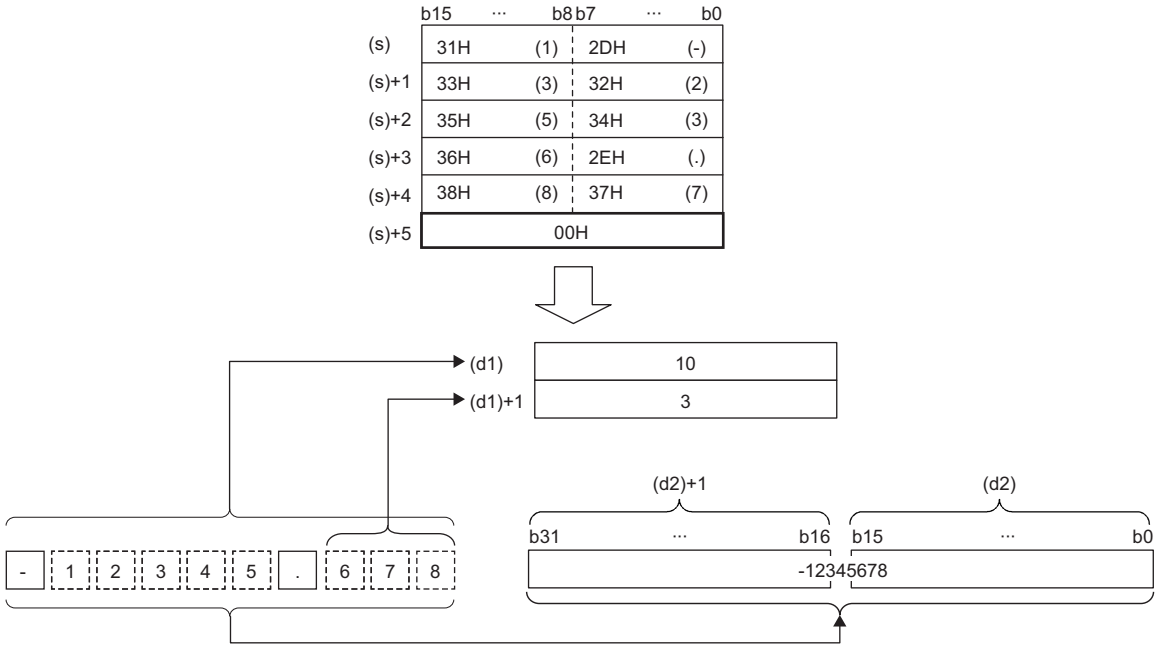
- These instructions convert the character strings stored in the device numbers specified by (s) and later to 32-bit binary data, and store the number of digits in (d1) and converted binary data in the device specified by (d2). For conversion of character strings to binary data, the data from the device number specified by (s) to the device number containing "00H" is processed as character 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

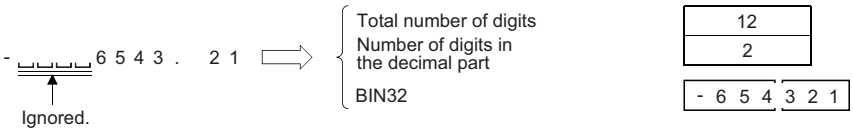


- The total number of characters of the character string stored in the device specified by (s) is 2 to 13.
- Of the character string stored in the device specified by (s), the number of characters in the decimal part is 0 to 10. Note, however, that the number must not exceed the total number of digits minus 3.
- The range of numerical character strings that can be converted to binary is as follows. Numerical character 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 character 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 character string in the device specified by (s), the instruction performs conversion to binary data by ignoring "20H" and "30H".

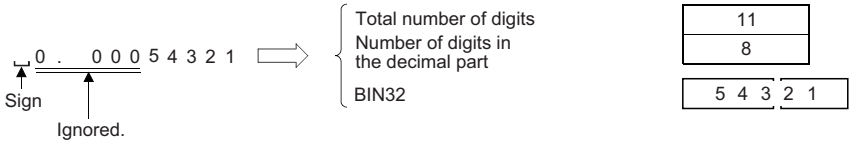
Ex.

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

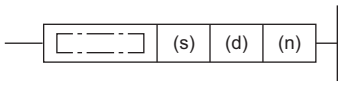
Error code (SD0)	Description
2820H	00H is not set between the device number specified by (s) and the last device number of the relevant device.
3401H	Invalid data that cannot be converted are input in (s). • The number of characters of the character string is not between 2 and 13. • The number of characters in the decimal part of the character 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.
3405H	The number of characters of the character string in the device specified by (s) exceeds 16383.

Converting hexadecimal ASCII to hexadecimal binary data

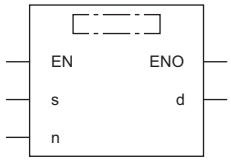
ASC2INT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

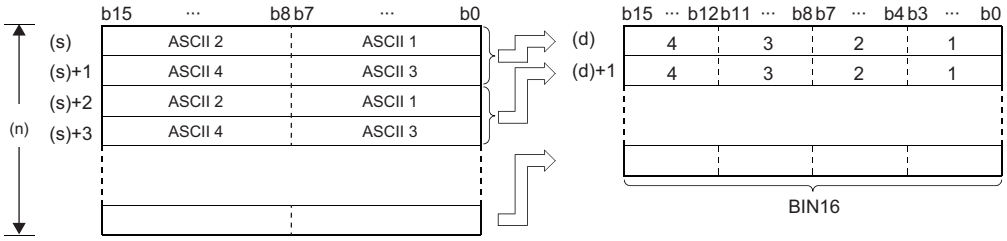
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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.

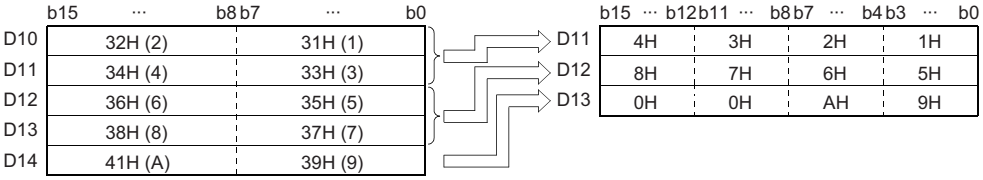


ASCII □: ASCII code (□th digit)

1 to 4: 1st to 4th digit

(n): Number of characters specified by (n)

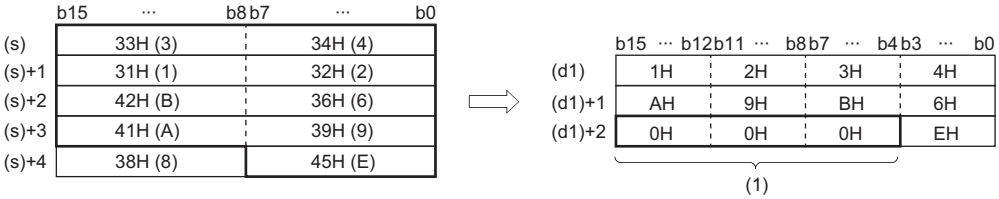
- Setting the number of characters for (n) automatically determines the range of the character string in the device specified by (s) and the device range in which the binary data in the device specified by (d) is stored.
- Processing is performed normally even if the device range in which the ASCII data 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

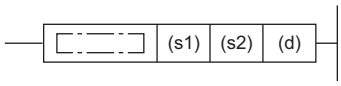
Error code (SD0)	Description
3401H	A character other than hexadecimal numerical character string (an ASCII code other than "30H" to "39H" and "41H" to "46H") is set in the device specified by (s).
3405H	Out-of-range data is set in the device specified by (n). • The specified number of characters is not between 0 and 16383.

Converting single-precision real number to BCD format data

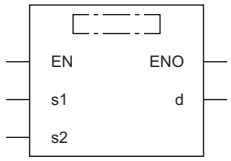
EMOD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

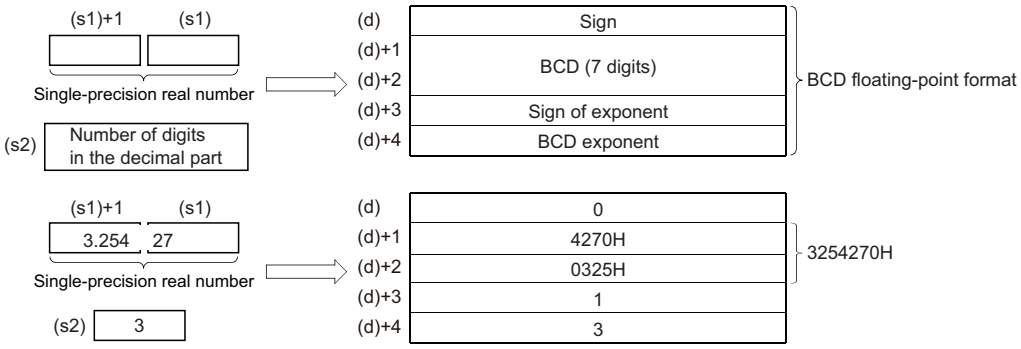
Operand	Description	Range	Data type	Data type (label)
(s1)	Single-precision real number data or the start device where the single-precision real number data is stored	0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Decimal part digit data	0 to 7	16-bit signed 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□\□, U3E□\(H)G□	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.

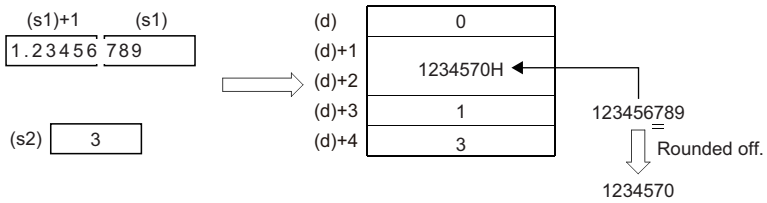


- For the sign in (d) and the exponent sign in (d)+3, 0 is set for positive and 1 is set for negative.
- For the BCD exponent in (d)+4, a value between 0 and 38 is stored.
- The number of decimal part digits of the single-precision real number data in the device specified by (s1) is stored in the device specified by (s2). The example in the above figure shows the following.

3.25427

(s2)=3

- 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 48 Precautions

Operation error

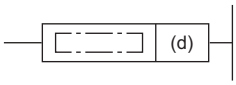
Error code (SD0)	Description
3401H	The number of decimal part digits in the device specified by (s2) is out of the range from 0 to 7.
3402H	The value set to a device or label in (s1) is -0, a subnormal number, NaN (not a number), or ±∞.

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

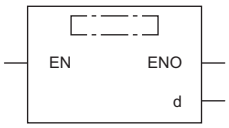
NEG(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

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

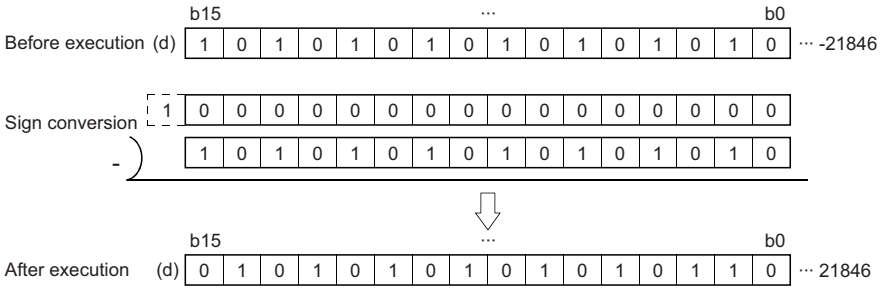
Operand	Description	Range	Data type	Data type (label)
(d)	Device where the data subjected to two's complement is stored	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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

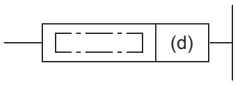
There is no operation error.

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

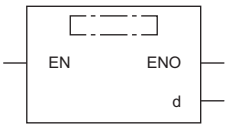
DNEG(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

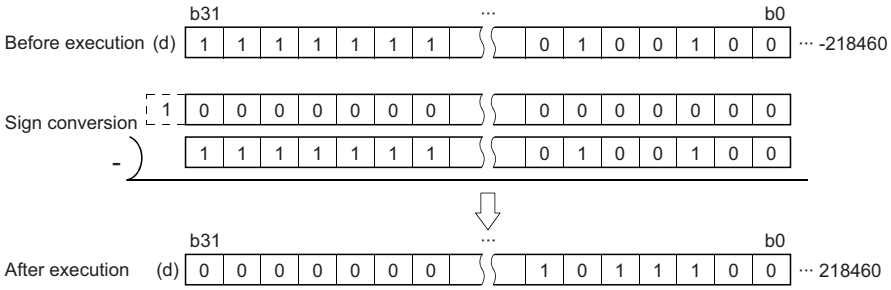
Operand	Description	Range	Data type	Data type (label)
(d)	Start device where the data subjected to two's complement is stored	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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

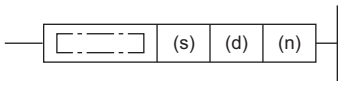
There is no operation error.

Decoding 8-bit data to 256-bit data

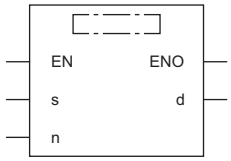
DECO(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

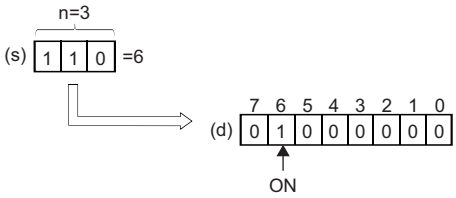
Operand	Description	Range	Data type	Data type (label)
(s)	Decode data or the start number of the device where the decode data is stored	—	Bit/16-bit signed binary	ANY_ELEMENTARY
(d)	Device for storing the decoded data	—	Bit/Word	ANY_ELEMENTARY
(n)	Effective bit length	1 to 8	16-bit unsigned binary	ANY16
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□\□, U3E□\(H)G□	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

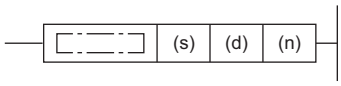
Error code (SD0)	Description
3401H	The value specified by (n) is out of the range, 0 to 8.

Encoding 256-bit data to 8-bit data

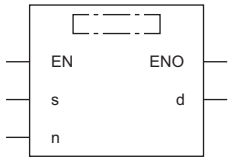
ENCO(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

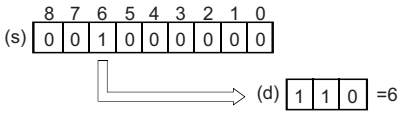
Operand	Description	Range	Data type	Data type (label)
(s)	Device where the encode data is stored	—	Bit/Word	ANY_ELEMENTARY
(d)	Start number of the device for storing the encoded data	—	Bit/16-bit signed binary	ANY_ELEMENTARY
(n)	Effective bit length	1 to 8	16-bit unsigned binary	ANY16
EN	Execution condition	—	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□\□, U3E□\(H)G□	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

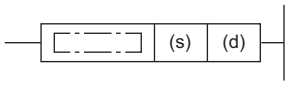
Error code (SD0)	Description
3401H	The value specified by (n) is out of the range, 0 to 8.
	The bits in the $2^{(n)}$ bit data in the device specified by (s) are all 0s.

Decoding data to seven-segment display data

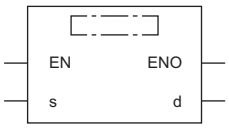
SEG(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

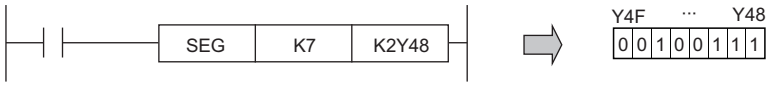
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

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.

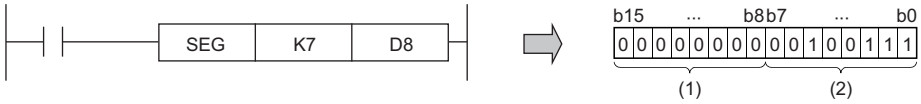
Bit device



The data in Y48 to Y4F does not change until the next data is output.

Ex.

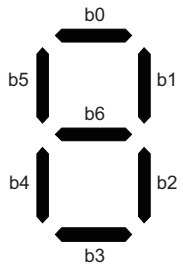
Word device



(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	
1	0001		0	0	0	0	0	1	1	0	
2	0010		0	1	0	1	1	0	1	1	
3	0011		0	1	0	0	1	1	1	1	
4	0100		0	1	1	0	0	1	1	0	
5	0101		0	1	1	0	1	1	0	1	
6	0110		0	1	1	1	1	1	0	1	
7	0111		0	0	1	0	0	1	1	1	
8	1000		0	1	1	1	1	1	1	1	
9	1001		0	1	1	0	1	1	1	1	
A	1010		0	1	1	1	0	1	1	1	
B	1011		0	1	1	1	1	1	0	0	
C	1100		0	0	1	1	1	0	0	1	
D	1101		0	1	0	1	1	1	1	0	
E	1110		0	1	1	1	1	0	0	1	
F	1111		0	1	1	1	0	0	0	1	

Operation error

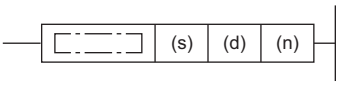
There is no operation error.

Separating data in units of 4 bits

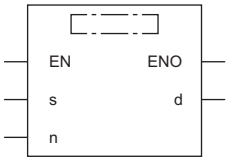
DIS(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

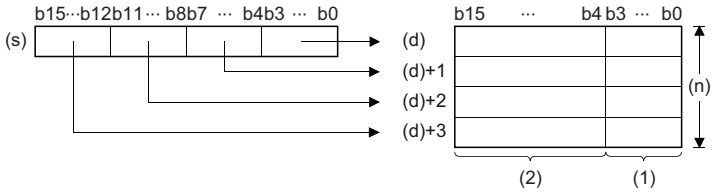
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

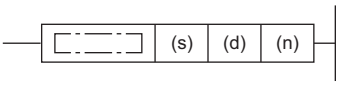
Error code (SD0)	Description
3401H	The value specified by (n) is out of the range, 0 to 4.

Combining data in units of 4 bits

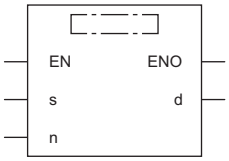
UNI(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

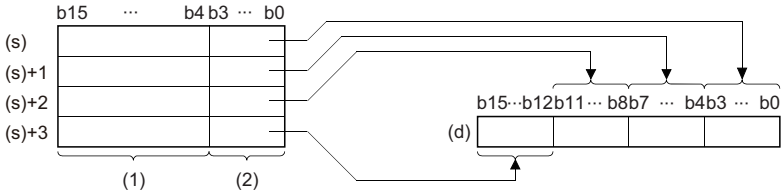
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

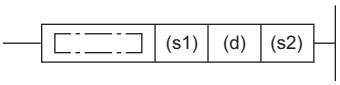
Error code (SD0)	Description
3401H	The value specified by (n) is out of the range, 0 to 4.

Separating data in units of bits

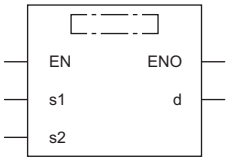
NDIS(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

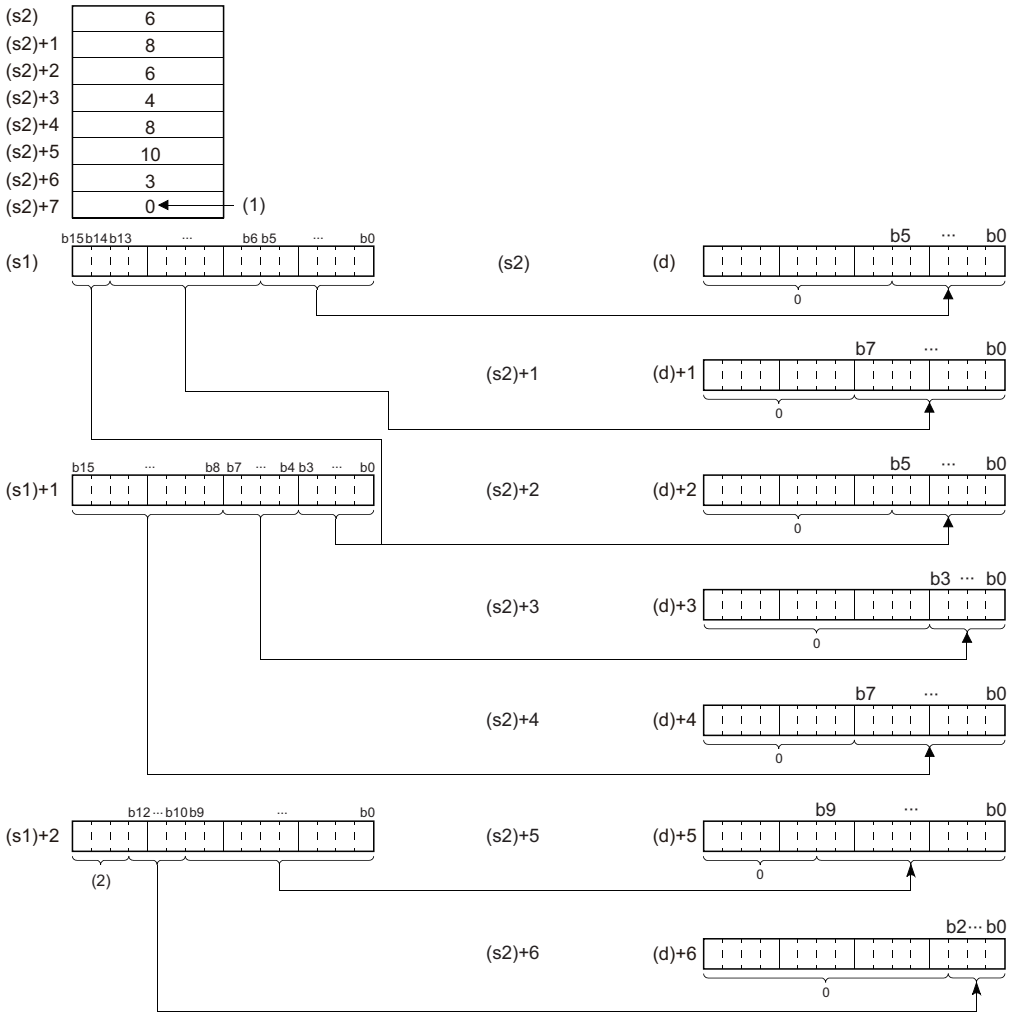
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

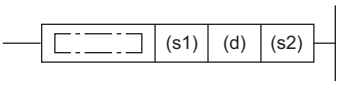
Error code (SD0)	Description
2821H	The device ranges specified by (s1) and (s2) are overlapping.
	The device ranges specified by (s1) and (d) are overlapping.
	The device ranges specified by (s2) and (d) are overlapping.
3401H	Invalid data that cannot be converted are input in (s2). • The value specified is out of the range, 1 to 16. • There is no 0 in the label or device area (between the specified device number and the last device number).

Combining data in units of bits

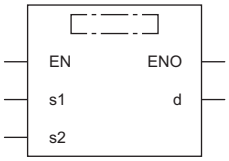
NUNI(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

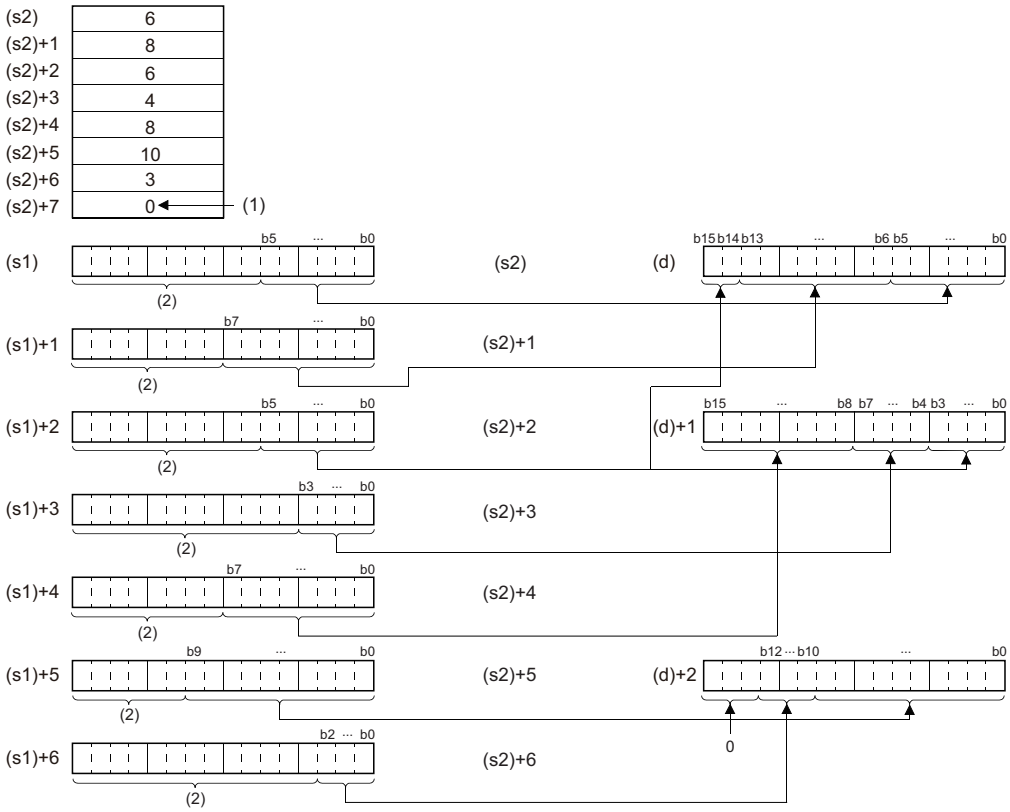
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

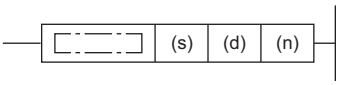
Error code (SD0)	Description
2821H	The device ranges specified by (s1) and (s2) are overlapping.
	The device ranges specified by (s1) and (d) are overlapping.
	The device ranges specified by (s2) and (d) are overlapping.
3401H	Invalid data that cannot be converted are input in (s2). • The value specified is out of the range, 1 to 16. • There is no 0 in the label or device area (between the specified device number and the last device number).

Separating data in units of bytes

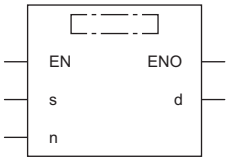
WTOB(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

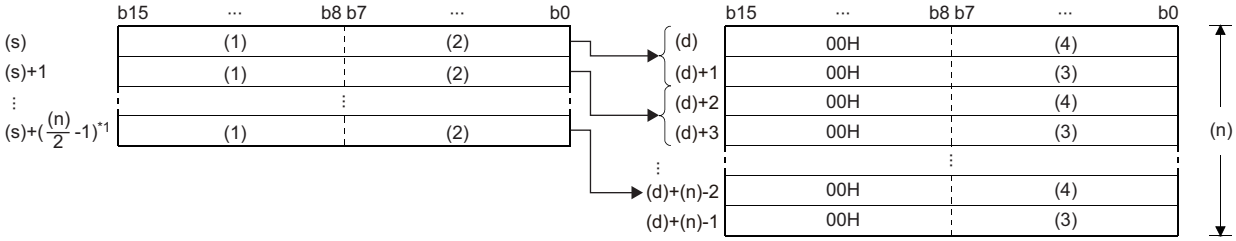
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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.

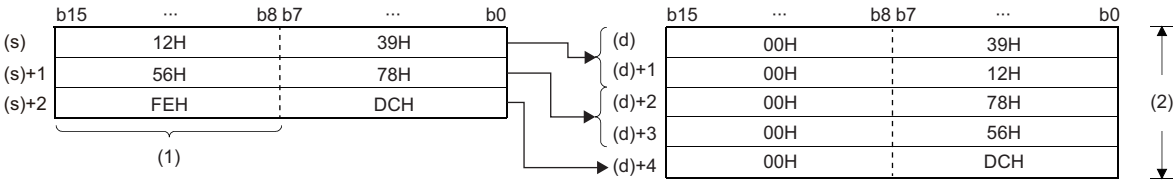


(1) Upper byte
(2) Lower byte
(3) Upper byte data
(4) Lower byte data

*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.

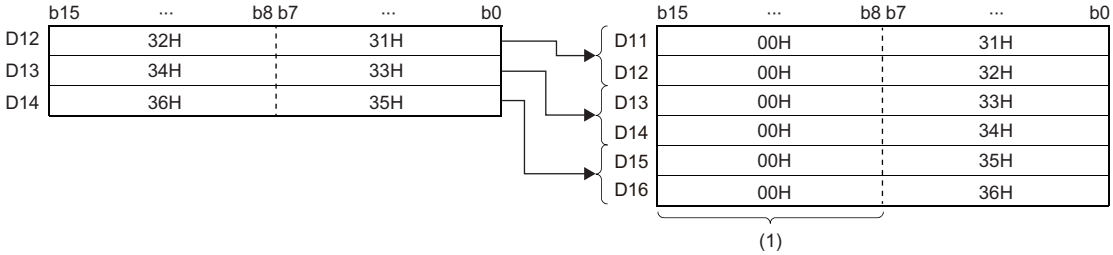


(1) The data, FEH, is ignored when (n) is 5.
(2) (n)=5

- 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

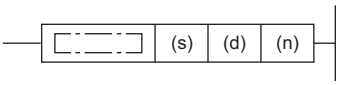
There is no operation error.

Combining data in units of bytes

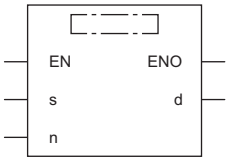
BTOW(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

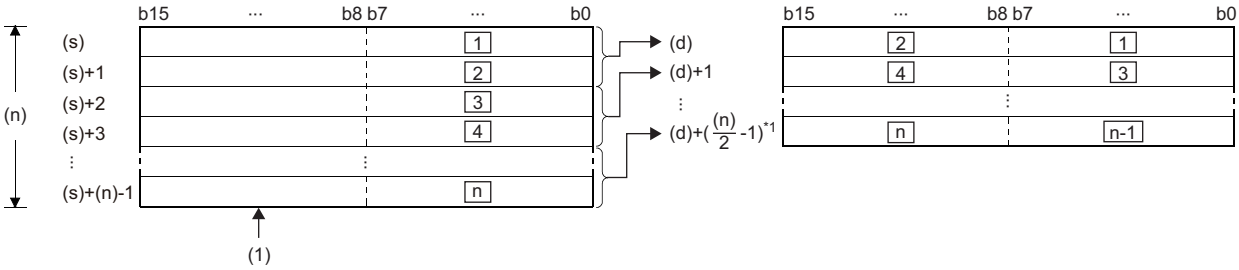
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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.



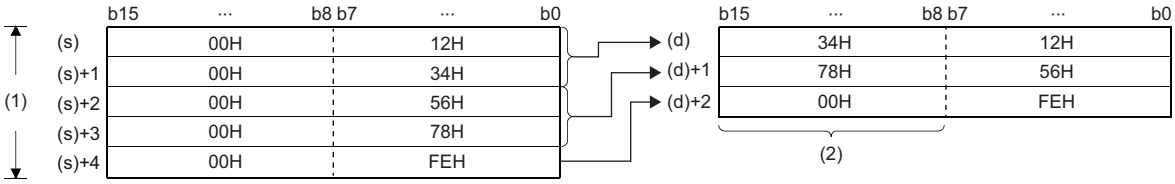
□: □th byte data

(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

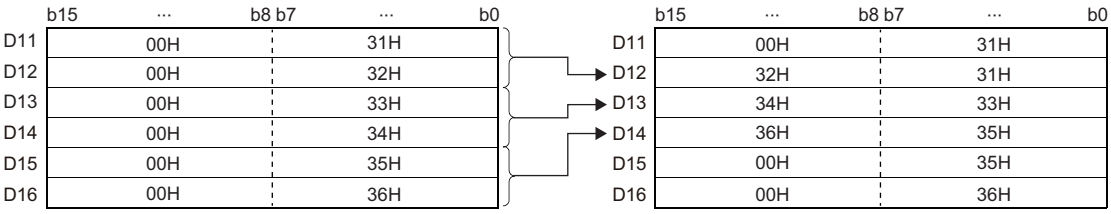
(2) Filled with 00H.

- 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.

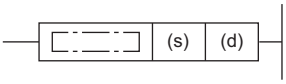
6.7 Data Transfer Instructions

Transferring 16-bit binary data

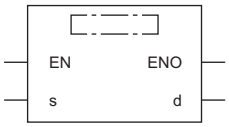
MOV(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

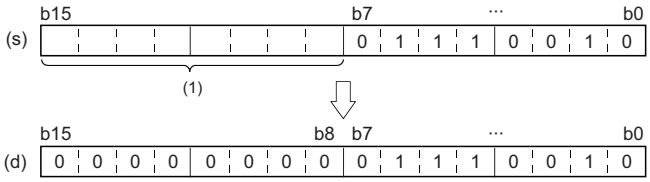
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

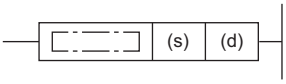
There is no operation error.

Transferring 32-bit binary data

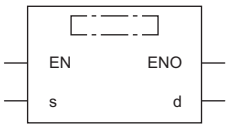
DMOV(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

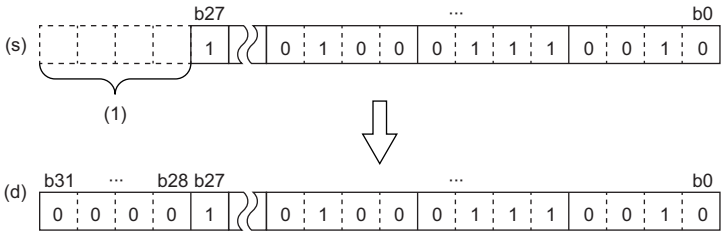
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

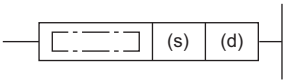
There is no operation error.

Inverting and transferring 16-bit binary data

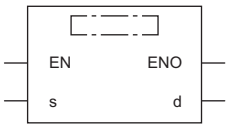
CML(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

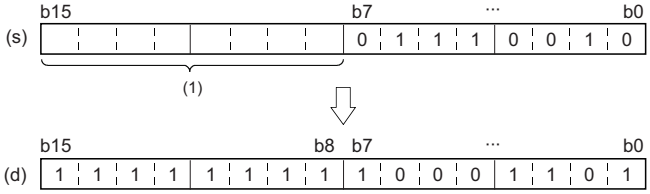
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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

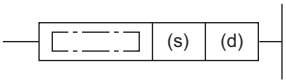
There is no operation error.

Inverting and transferring 32-bit binary data

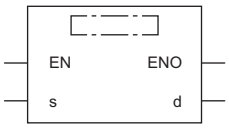
DCML(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

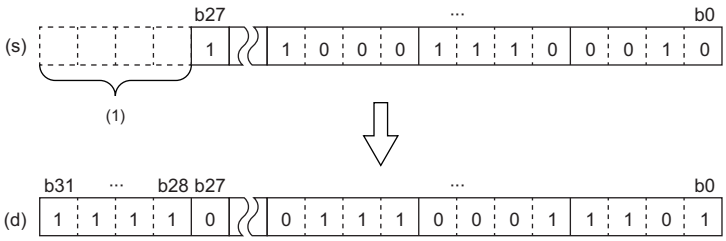
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	○	○	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(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.

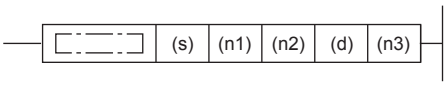
Shifting data in units of 4 bits

SMOV(P)

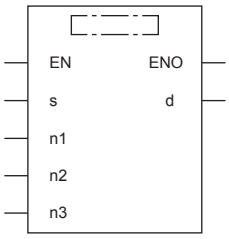
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions distribute and combine data in units of 4 bits.

Ladder	ST
	ENO:=SMOV(EN,s,n1,n2,n3,d); ENO:=SMOVP(EN,s,n1,n2,n3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SMOV	
SMOVP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device where the specified units of data is stored	—	16-bit signed binary	ANY16
(n1)*1	Start position where the data to be shifted is stored	1 to 4	16-bit unsigned binary	ANY16_U
(n2)*1	Number of units to be shifted	1 to 4	16-bit unsigned binary	ANY16_U
(d)	Device where the shifted data is stored	—	16-bit signed binary	ANY16
(n3)*1	Start unit position of shift destination	1 to 4	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Set values so that (n2)≤(n1) and (n2)≤(n3).

■Applicable devices

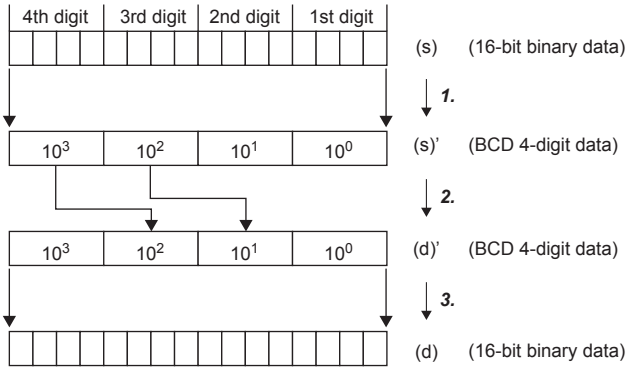
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	○	○	○	○	—	—	○	—	—	—	—
(n1)	○*1	○	○	○	○	—	—	○	○	—	—	—
(n2)	○*1	○	○	○	○	—	—	○	○	—	—	—
(d)	○*1	○	○	○	○	—	—	○	—	—	—	—
(n3)	○*1	○	○	○	○	—	—	○	○	—	—	—

*1 FX and FY cannot be used.

Processing details

- When SM773 is OFF, the contents of (s) and (d) are converted to 4-digit BCD (0000 to 9999). After the data in lower (n2) digits from the (n1)th digit is transferred (combined) to (d) starting from the (n3)th digit, the data is converted into BIN and then stored in (d).

In the case of (n1) = 4, (n2)= 2 and (n3) = 3



1. Conversion

(s) is converted from binary to BCD data.

2. Shifting

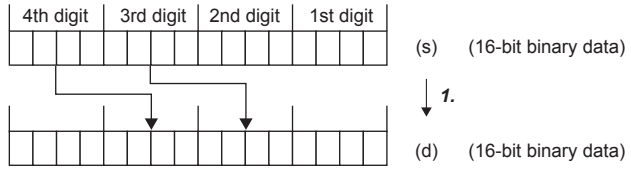
Lower (n2) digits starting from the (n1)th digit are transferred (combined) to (d)' starting from the (n3)th digit. In the case of this example, the first and fourth digits of (d)' are not affected even if data is transferred from (s)'.

3. Conversion

The combined data (BCD) is converted into binary, and stored to (d).

- When SM773 is on, conversion from BIN to BCD is not performed and data are shifted in units of 4 bits.

In the case of (n1) = 4, (n2)= 2 and (n3) = 3



1. Shifting data in units of 4 bits

Lower (n2)×4-bit data from (n1)×4th bit is transferred (combined) by lower (n2)×4-bit data from (n3)×4th bit of (d). In the case of this example, the first and fourth digits of (d) are not affected even if data is transferred from (s).

- When digit specification of bit is used for (n1), the number of digits becomes the specified number of digits × 4 bits from the start bit device.

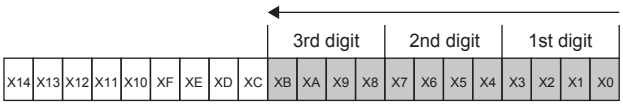
Ex.

To specify "3 digits × 4 bits" from X0:

[SMOV K3X0 K3 K2 K2X100 K2]
(1) (2)

(1) Specified constant

(2) Start bit device



Digit specification denotes the "specified number of digits 3×4 bits" = 12 bits from the start bit device X0.

Operation error

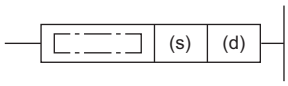
Error code (SD0)	Description
3405H	Any one of (n1), (n2), or (n3) is other than 1 to 4.
	Either (s) or (d) is other than 0 to 9999 when SM773 is OFF.
	(n2) is greater than (n1) or (n3).

Inverting and transferring 1-bit data

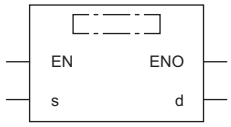
CMLB(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

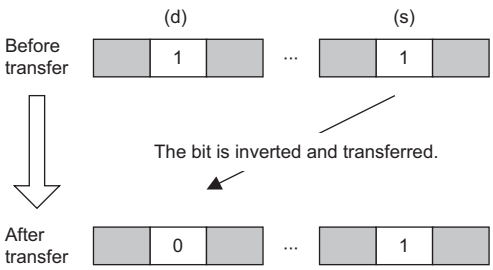
Operand	Description	Range	Data type	Data type (label)
(s)	Inversion target data or the number of the device where the inversion target data is stored	—	Bit	ANY_BOOL
(d)	Transfer destination device number	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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

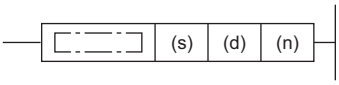
There is no operation error.

Transferring 16-bit binary data block (16 bits)

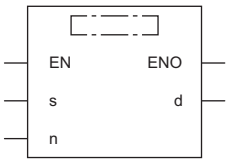
BMOV(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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 available for the engineering tool with version "1.030G" or later (except for BOOL, POINTER, and STRUCT data types (structures)). (Available data types are the same as the ones for GX Works2.) 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

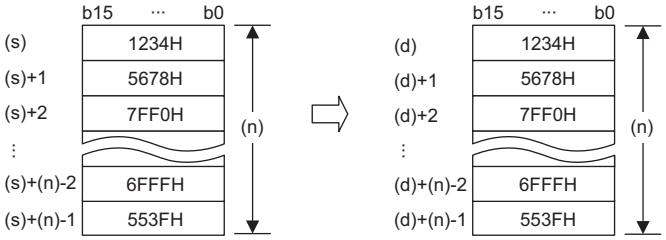
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	—	○	—	○	—	—	—	—
(d)	○	○	○	○	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	—
(d)	○	○	—
(n)	○	○	○

Processing details

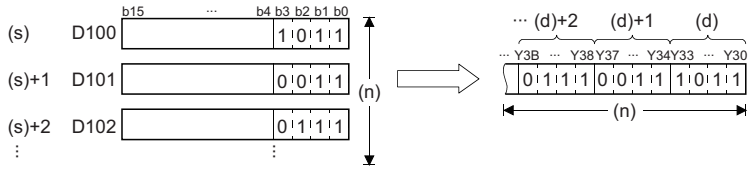
- These instructions batch-transfer the (n) points of 16-bit binary data starting from the device specified by (s) to the device specified by (d).



- 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.

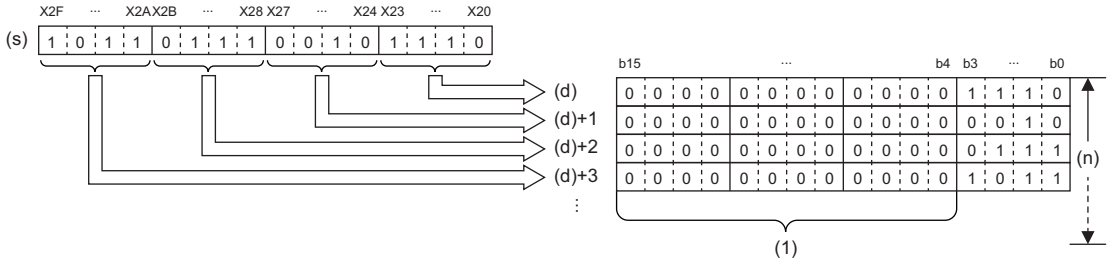
When K1Y30 is specified in (d), the lower 4 bits of the word device specified by (s) are transferred.



- 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 K1X20 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). However, the CPU buffer memory access device (U3E0\G□) of the host CPU module in which index modification is not specified in the I/O No. specification can be specified by both (s) and (d).

Operation error

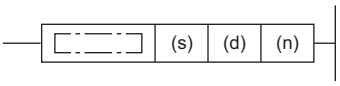
Error code (SD0)	Description
3420H	The link direct device, module access device, or CPU buffer memory access device is specified by both (s) and (d). However, this is not true when the CPU buffer memory access device (U3E0\G□) of the host CPU module in which index modification is not specified in the I/O No. specification is specified.

Transferring 16-bit binary data block (32 bits)

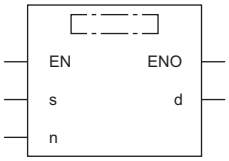
BMOVL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

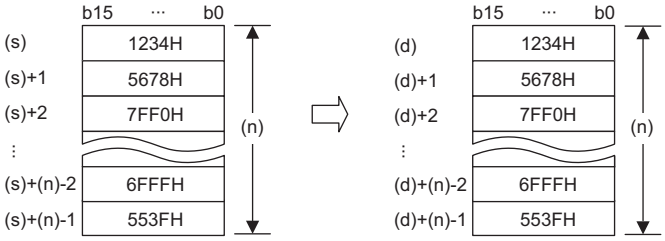
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	—	○	—	○	—	—	—	—
(d)	○	○	○	○	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	○	○	○	○	—	—	—

Processing details

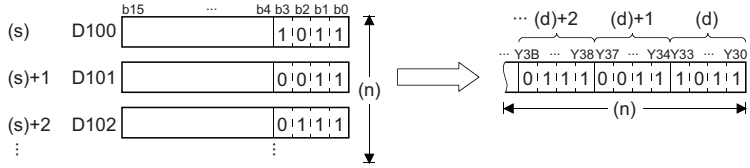
- These instructions batch-transfer the (n) points of 16-bit binary data starting from the device specified by (s) to the device specified by (d).



- 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 K1Y30 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. Note that the CPU buffer memory access device (U3E0\G□) of the host CPU module in which index modification is not specified in the I/O No. specification can be specified by both (s) and (d).

Operation error

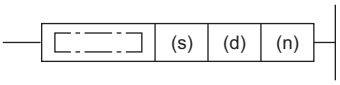
Error code (SD0)	Description
3420H	The link direct device, module access device, or CPU buffer memory access device is specified by both (s) and (d). However, this is not true when the CPU buffer memory access device (U3E0\G□) of the host CPU module in which index modification is not specified in the I/O No. specification is specified.

Transferring the same 16-bit binary data block (16 bits)

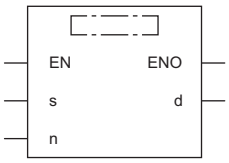
FMOV(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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*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 available for the engineering tool with version "1.030G" or later (except for BOOL, POINTER, and STRUCT data types (structures)). (Available data types are the same as the ones for GX Works2.) 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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

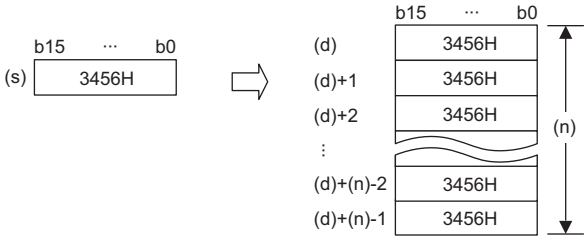
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	—
(d)	○	○	—
(n)	○	○	○

Processing details

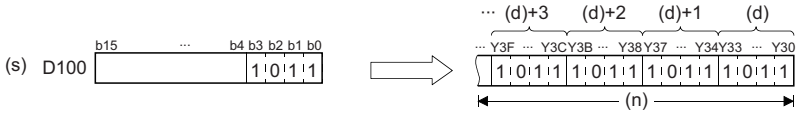
- These instructions transfer the 16-bit binary data in the device specified by (s) to the (n) points of the device specified by (d).



- 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.

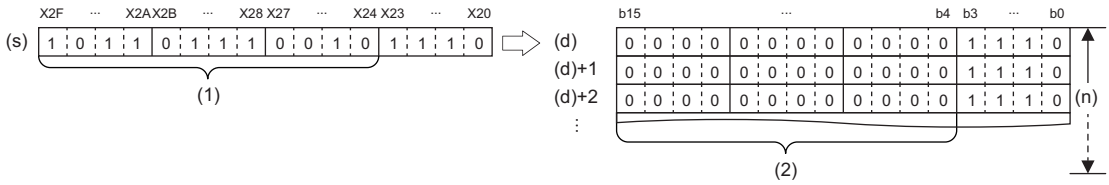
When K1Y30 is specified in (d), the lower 4 bits of the word device specified by (s) are transferred.



- 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 K1X20 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

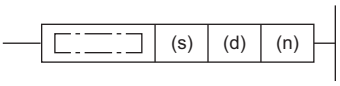
There is no operation error.

Transferring the same 16-bit binary data block (32 bits)

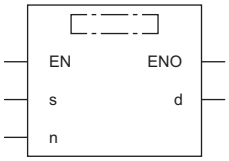
FMOVL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

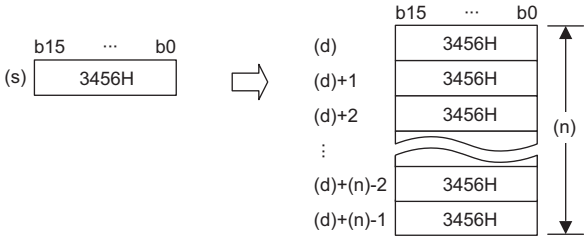
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○	○	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	○	○	○	○	—	—	—

Processing details

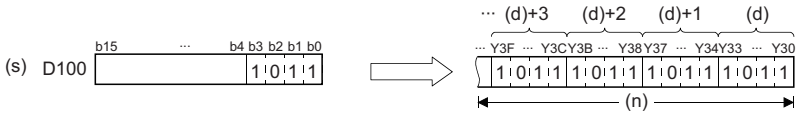
- These instructions transfer the 16-bit binary data in the device specified by (s) to the (n) points of the device specified by (d).



- 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 K1Y30 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

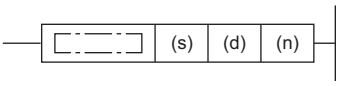
There is no operation error.

Transferring the same 32-bit binary data block (16 bits)

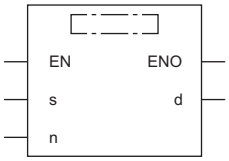
DFMOV(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

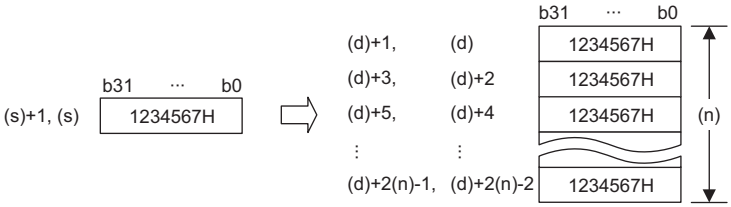
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

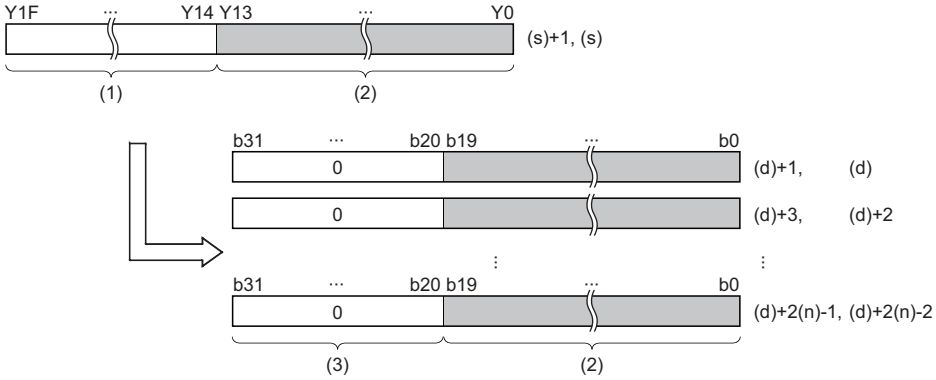
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

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).

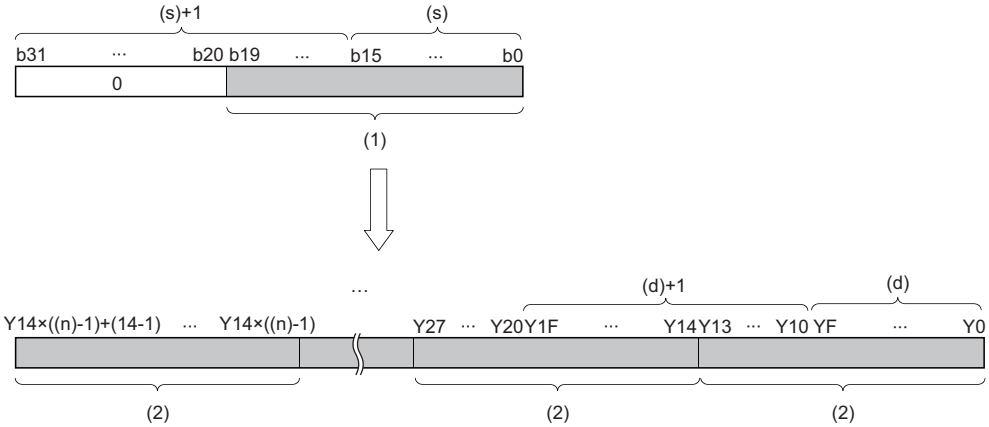


- When the number of digits is specified for the data in the device specified by (s), only the data corresponding to the specified digits are transferred. When K5Y0 is specified in (s), the lower 20 bits (five digits) of the word device specified by (s) are transferred.



(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 K5Y0 is specified in (d), the lower 20 bits (five digits) of the word 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

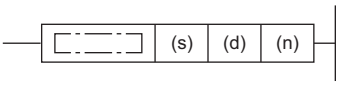
There is no operation error.

Transferring the same 32-bit binary data block (32 bits)

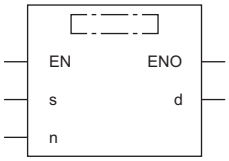
DFMOVL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

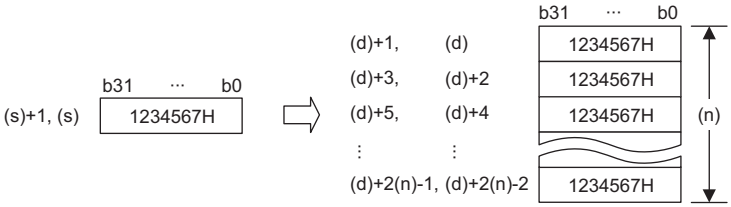
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

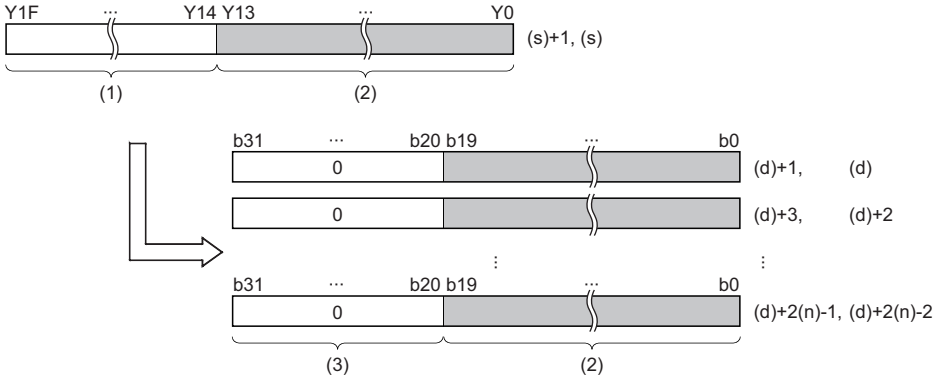
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	○	○	○	○	—	—	—

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).

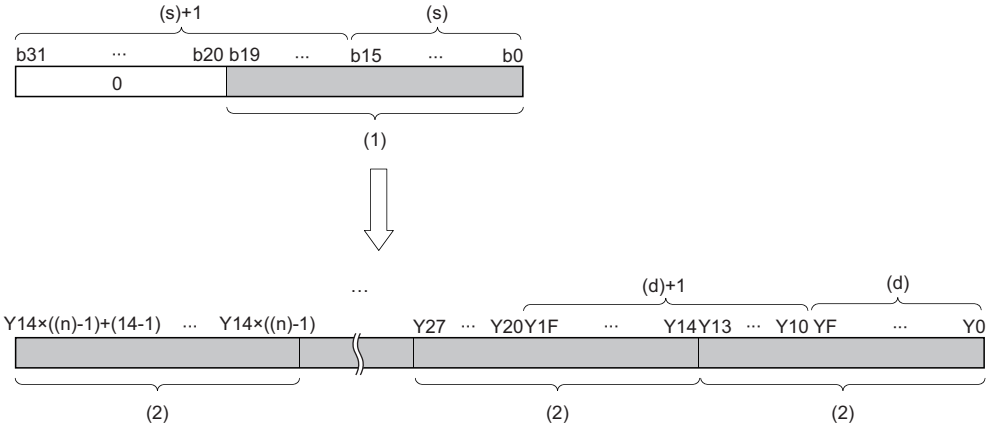


- When the number of digits is specified for the data in the device specified by (s), only the data corresponding to the specified digits are transferred. When K5Y0 is specified in (s), the lower 20 bits (five digits) of the word device specified by (s) are transferred.



(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 K5Y0 is specified in (d), the lower 20 bits (five digits) of the word 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

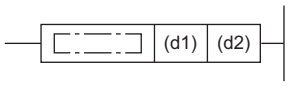
There is no operation error.

Exchanging 16-bit binary data

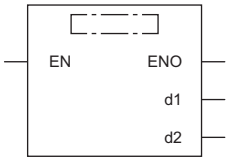
XCH(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

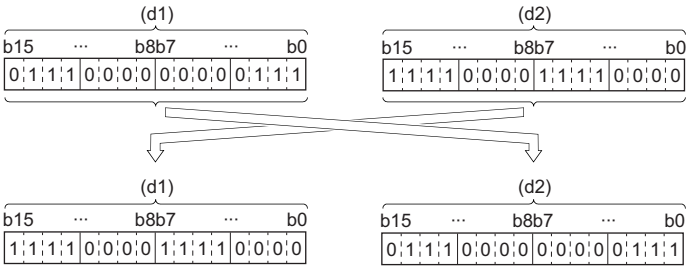
Operand	Description	Range	Data type	Data type (label)
(d1)	Device where the exchange target data is stored	—	16-bit signed binary	ANY16
(d2)	Device where the exchange target data is stored	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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

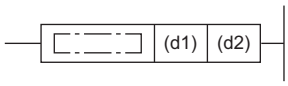
There is no operation error.

Exchanging 32-bit binary data

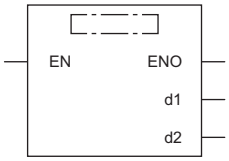
DXCH(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

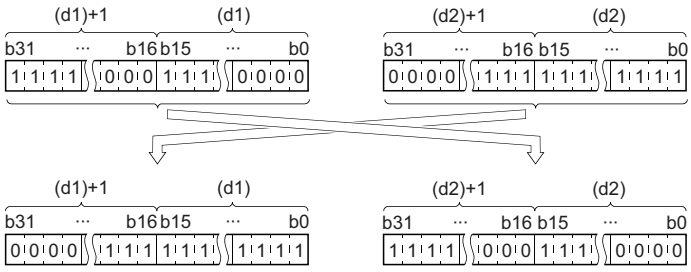
Operand	Description	Range	Data type	Data type (label)
(d1)	Start device where the exchange target data is stored	—	32-bit signed binary	ANY32
(d2)	Start device where the exchange target data is stored	—	32-bit signed binary	ANY32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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

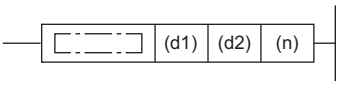
There is no operation error.

Exchanging 16-bit binary block data

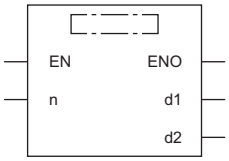
BXCH(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

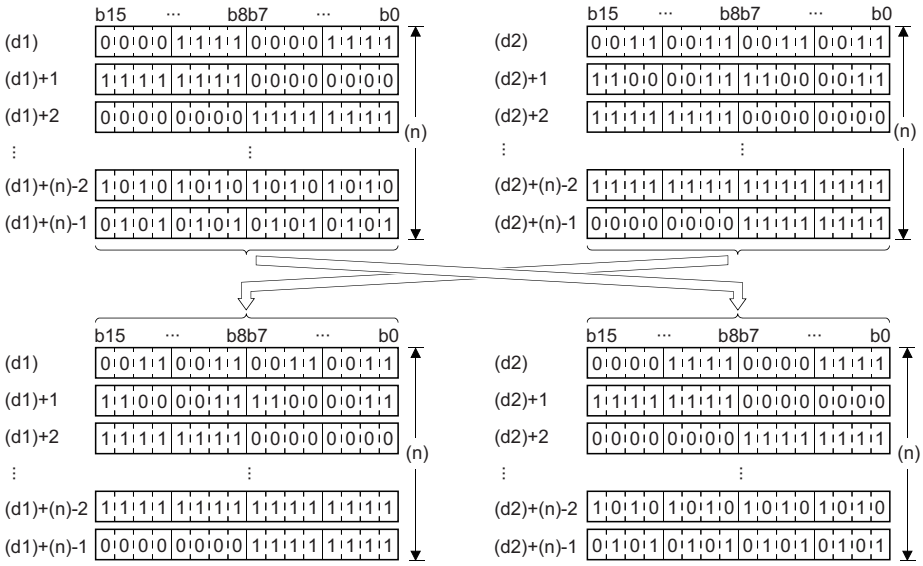
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

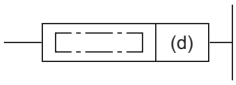
Error code (SD0)	Description
2821H	The device ranges specified by (d1) and (d2) are overlapping.

Exchanging the upper and lower bytes of 16-bit binary data

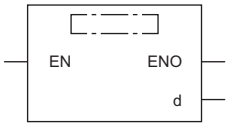
SWAP(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

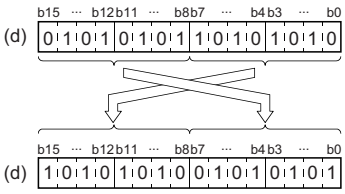
Operand	Description	Range	Data type	Data type (label)
(d)	Start device for storing the data whose upper and lower 8-bit data is exchanged	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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.

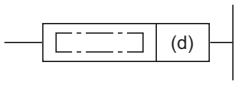
Exchanging the upper and lower bytes of 32-bit binary data

DSWAP(P)

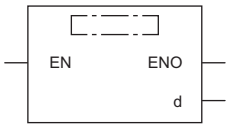
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

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

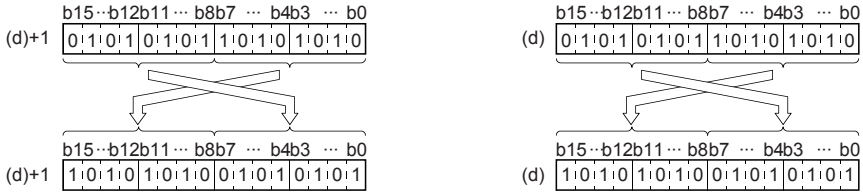
■Applicable devices

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□\□, U3E□\(H)G□	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

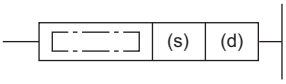
There is no operation error.

Transferring 1-bit data

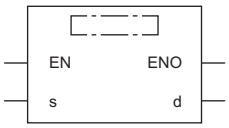
MOVB(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

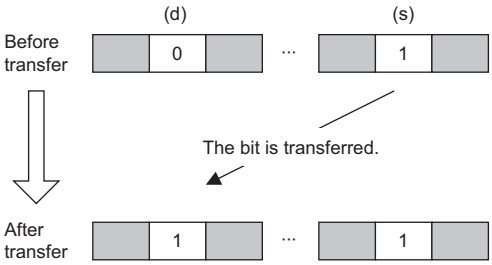
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	—	—	○	—	○	—	—	—	—
(d)	○	○	○	—	—	○	—	○	—	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	—
(d)	○	○	—

Processing details

- These instructions transfer the bit data in the device specified by (s) to the device specified by (d).



Operation error

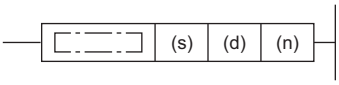
There is no operation error.

Transferring n-bit data

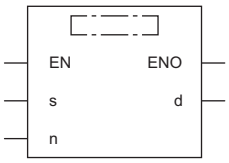
BLKMOVB(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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.

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only safety devices and safety labels of data types described in the table can be used.

■Applicable devices

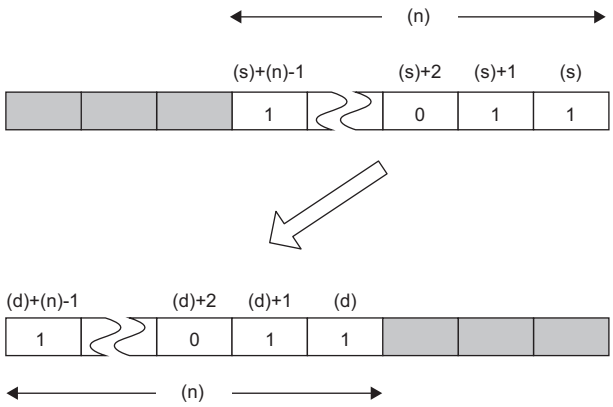
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

- In safety programs executed by the SIL2 Process CPU and Safety CPU, only the following safety devices and constants can be used.

Operand	Bit	Word	Constant
	SA\X, SA\Y, SA\M, SA\SM, SA\B	SA\T, SA\ST, SA\C, SA\D, SA\W, SA\SD	K, H
(s)	○	○	—
(d)	○	○	—

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.

MEMO

PART 5 APPLICATION INSTRUCTIONS

This part consists of the following chapters.

7 PROGRAM CONTROL

8 DATA PROCESSING

9 DEBUGGING AND FAILURE DIAGNOSTIC

10 STRING PROCESSING

11 REAL VALUE PROCESSING

12 RANDOM NUMBER

13 DEVICE OPERATION

14 TIMER, COUNTER

15 SHORTCUT CONTROL

16 RAMP SIGNAL

17 MATRIX INPUT

18 CPU MODULE DATABASE ACCESS FUNCTION

19 CLOCK

20 MODULE ACCESS

21 PARAMETER SETTING OPERATION

22 CPU MODULE DATA LOGGING FUNCTION

23 RECORDING FUNCTION

24 BUILT-IN ETHERNET FUNCTION INSTRUCTIONS

25 PID OPERATION INSTRUCTION

26 PID CONTROL INSTRUCTIONS

27 MULTIPLE CPU DEDICATED INSTRUCTIONS

28 SFC PROGRAM INSTRUCTIONS

29 REDUNDANT SYSTEM INSTRUCTIONS

30 SAFETY SYSTEM INSTRUCTIONS

7 PROGRAM CONTROL

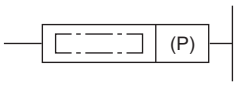
7.1 Program Branch Instructions

Pointer branch

CJ, SCJ, JMP

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- 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

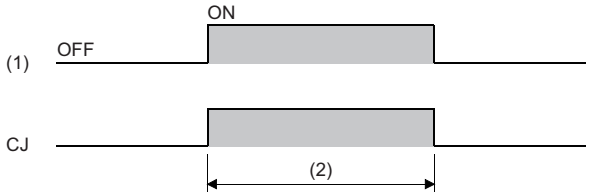
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(P)	—	—	—	—	—	—	—	—	—	—	—	○

Processing details

■CJ

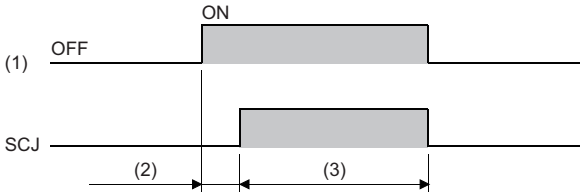
- This instruction executes the program specified by the pointer number within the same program file when the execution command is on.
- When the execution command is off, the program in the next step is executed.



(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.



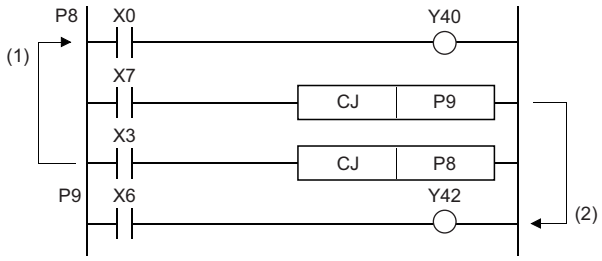
(1) Execution command
(2) One scan
(3) Executed in each scan

■JMP

- This instruction unconditionally executes the program specified by the pointer number within the same program file.

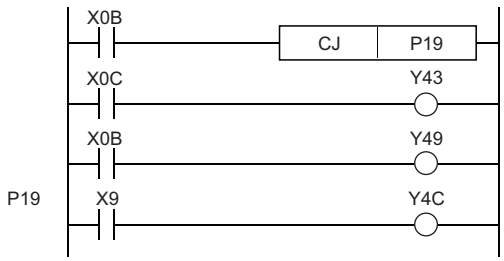
Precautions

- 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.



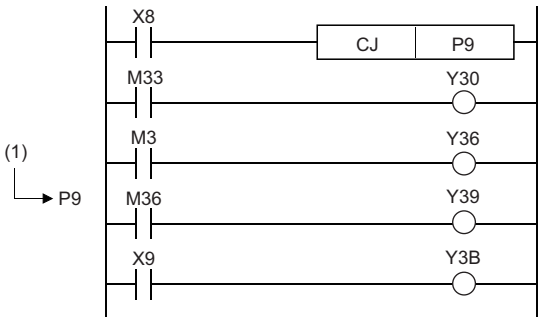
(1) While X3 is on, the loop is repeated.
(2) To exit the loop, turn on X7.

- The value in the device skipped with these instructions remains unchanged.



When XB turns on, the program jumps to the label, P19.
Y43 and Y49 remain unchanged even if XB and XC 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

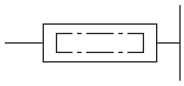
Error code (SD0)	Description
3380H	The pointer number specified by (P) is not set before the END instruction.
	The pointer number specified by (P) is not used as a label in the same program.
	The pointer number specified by (P) is a global pointer in another program.

Jumping to END

GOEND

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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
3340H	After execution of the FOR instruction, the GOEND instruction is executed before the NEXT instruction.
3381H	After execution of the CALL(P) or ECALL(P) instruction, the GOEND instruction is executed before the RET instruction.
33A1H	The GOEND instruction is executed before the IRET instruction in the interrupt program specified by the interrupt pointer (I).

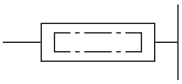
7.2 Program Execution Control Instructions

Disabling/enabling interrupt programs

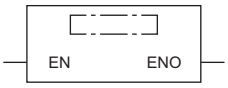
DI, EI

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- 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	Every scan

Processing details

■DI

- This instruction disables execution of interrupt programs until the EI instruction is executed even though they are requested.
- When the system is powered on or the CPU module is reset, the system is in the state where the DI instruction has been executed.
- 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.

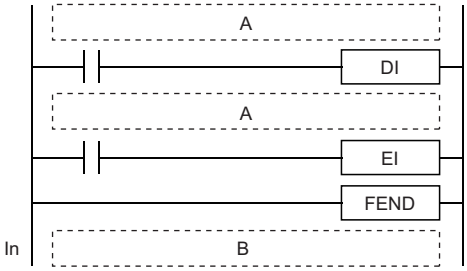
☞ Page 570 DI

- 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.
- For the operation of the EI instruction used with the DI (Disabling interrupt programs with specified priority or lower) instruction, refer to the following.

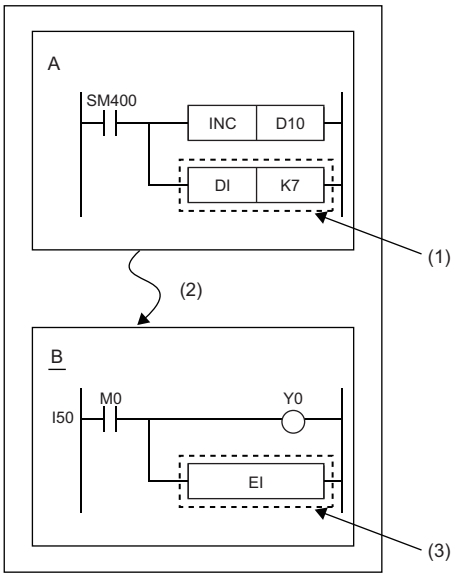
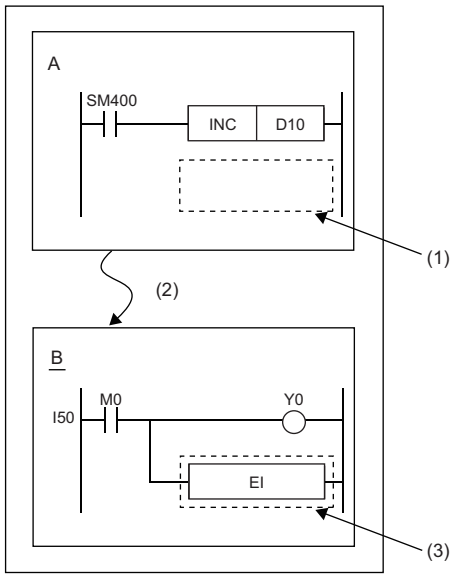
☞ Page 570 DI



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.

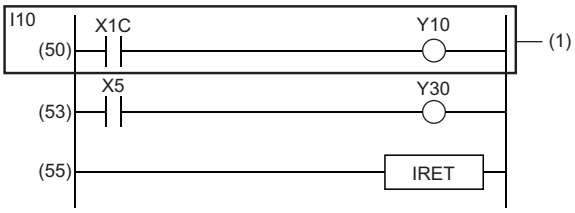
A: Sequence program
B: Interrupt program

- 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.

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.)

Point

- An interrupt pointer occupies one step. (At the program (1) below, for example, I10 is step 50, X1C is step 51, and Y10 is step 52.)



- If the EI and DI instructions are provided in the master control, these instructions are executed regardless of the execution status of the MC instruction.

Operation error

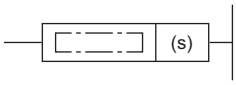
Error code (SD0)	Description
3362H	More than 16 DI (Disabling interrupt programs) instructions and DI (Disabling interrupt programs with specified priority or lower) instructions are nested.

Disabling interrupt programs with specified priority or lower

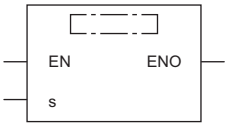
DI

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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	Every scan

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Interrupt disable priority	1 to 8*1	16-bit unsigned binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 For the R00CPU, R01CPU, and R02CPU, the range is 3 to 8.

■Applicable devices

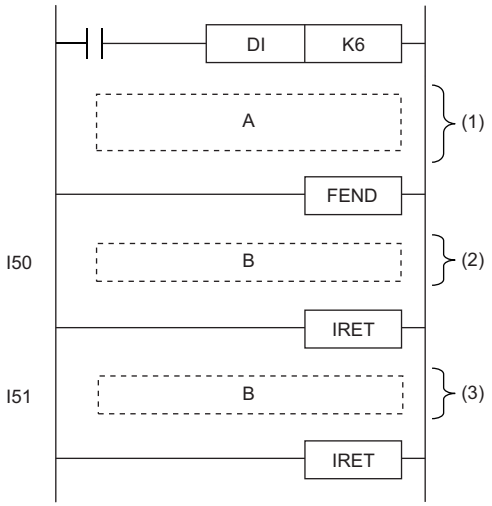
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	○	—	—	○	○	—	—	—

Processing details

- This instruction disables execution of interrupt programs with the interrupt pointer numbers specified by (s) or lower.

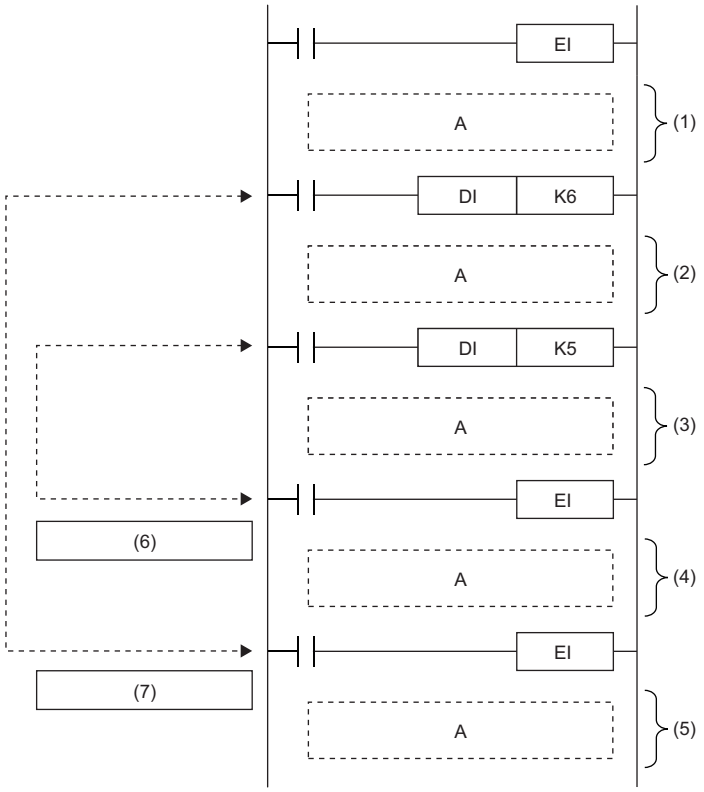
Interrupt priority setting

I No.	Priority
I50	5
I51	6



A: Sequence program
B: Interrupt program
(1) Interrupt disabled section for priority 6 to 8 (Interrupt enabled section for priority 1 to 5)
(2) The interrupt program can be executed because its priority is 5.
(3) The interrupt program cannot be executed because its priority is 6.

- 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.



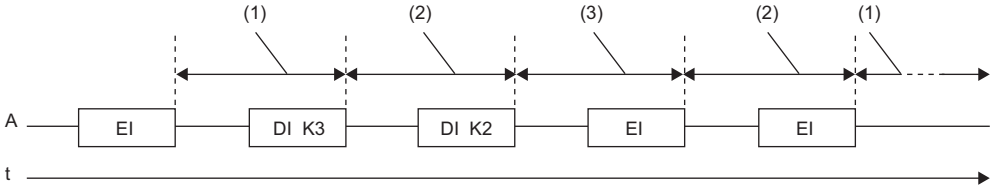
A: Sequence program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 6 to 8 (Interrupt enabled section for priority 1 to 5)
(3) Interrupt disabled section for priority 5 to 8 (Interrupt enabled section for priority 1 to 4)
(4) Interrupt disabled section for priority 6 to 8 (Interrupt enabled section for priority 1 to 5)
(5) Interrupt enabled section for all priorities
(6) EI instruction for [DI K5]
(7) EI instruction for [DI K6]

- 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[*1] can be set in parameter. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

*1 I0 to I15, I50 to I1023

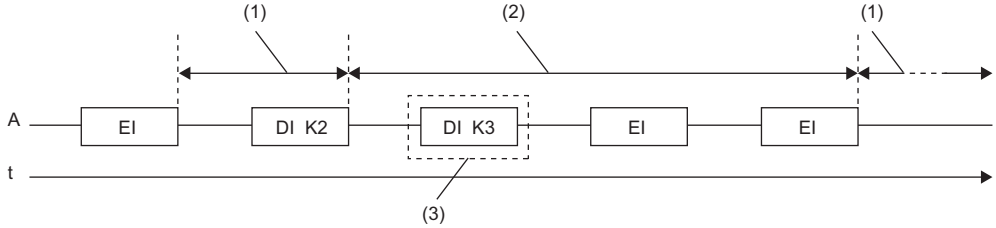
- The interrupt disabled priority currently set can be checked in SD758.
- If the DI (Disabling interrupt programs with specified priority or lower) instruction is executed in the interrupt program and the interrupt disabled priority is changed, the value in SD758 also changes.

- 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



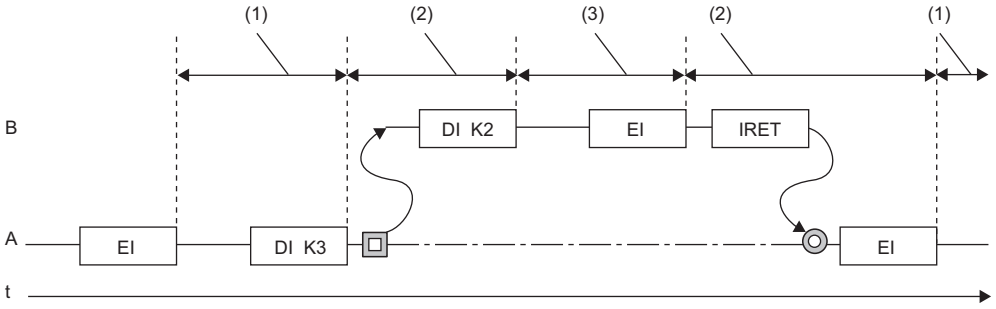
A: Scan execution type program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 3 to 8 (Interrupt enabled section for priority 1 and 2)
(3) Interrupt disabled section for priority 2 to 8 (Interrupt enabled section for priority 1)

  - 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



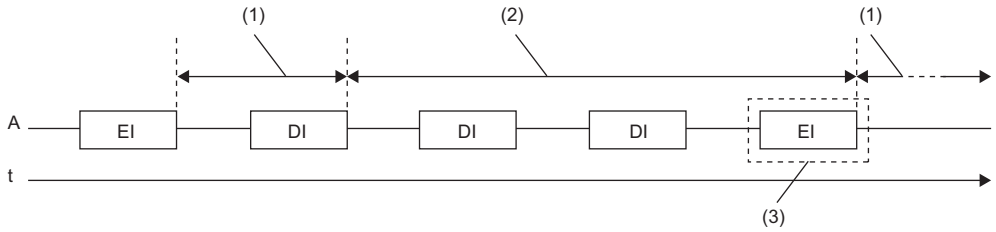
A: Scan execution type program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 2 to 8 (Interrupt enabled section for priority 1)
(3) The priority of the disabled interrupt remains unchanged because the interrupt with priority 2 or lower is already disabled.

  - When the DI (Disabling interrupt programs with specified priority or lower) instruction is executed in the interrupt program



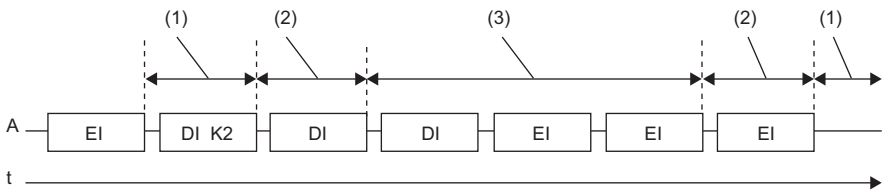
A: Scan execution type program
B: Interrupt program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 3 to 8 (Interrupt enabled section for priority 1 and 2)
(3) Interrupt disabled section for priority 2 to 8 (Interrupt enabled section for priority 1)

  - When only the DI (Disabling interrupt programs) instruction is executed



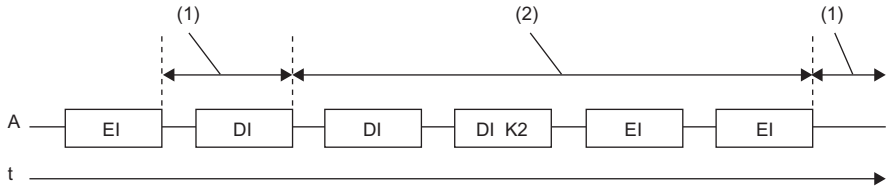
A: Scan execution type program
(1) Interrupt enabled section for all priorities
(2) Interrupt disabled section for priority 1 to 8 (Interrupt disabled section for all priorities)
(3) Executing the EI instruction only once enables interrupts with all priorities.

- 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 to 8 (Interrupt enabled section for priority 1)
(3) Interrupt disabled section for priority 1 to 8 (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 to 8 (Interrupt disabled section for all priorities)

Operation error

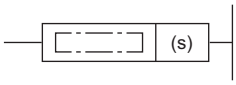
Error code (SD0)	Description
3362H	More than 16 DI (Disabling interrupt programs) instructions and DI (Disabling interrupt programs with specified priority or lower) instructions are nested.
3405H	The priority specified by (s) is out of range.

Interrupt program mask

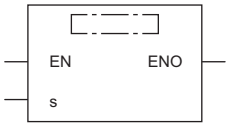
IMASK

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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 CPU module 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.

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 577 SIMASK

Operation error

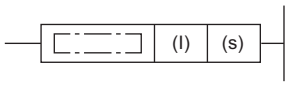
There is no operation error.

Disabling/enabling the specified interrupt pointer

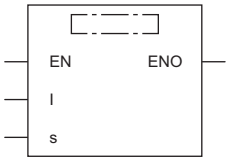
SIMASK

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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	Every scan

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 (I)
	X, 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	$	
(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 CPU module 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 are stored in SD1400 to SD1463.

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

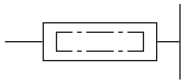
Error code (SD0)	Description
3405H	The interrupt pointer number specified by (I) is out of the valid range.
	The value in the device specified by (s) is neither 0 (disabled) nor 1 (enabled).

Returning from the interrupt program

IRET

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

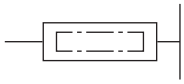
Error code (SD0)	Description
33A0H	There is no pointer corresponding to the interrupt number.
33A1H	After an interrupt occurs, the END, FEND, GOEND, or STOP instruction is executed before the IRET instruction.
33A2H	The IRET instruction is executed before the interrupt program specified by the interrupt pointer (I) is executed.
33A3H	The IRET instruction is executed in a fixed scan execution type program.
33A4H	The IRET instruction is executed in an event execution type program.

Resetting the watchdog timer

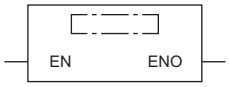
WDT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions reset the watchdog timer.

Ladder	ST
	ENO:=WDT(EN); ENO:=WDTP(EN);

FBD/LD

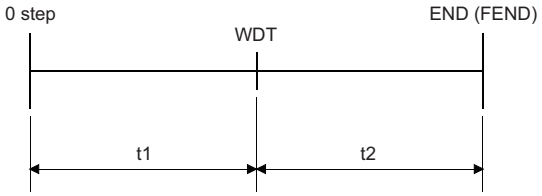


■Execution condition

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.

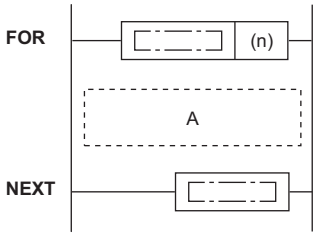
7.3 Structure Creation Instructions

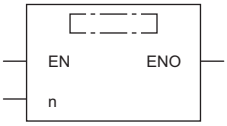
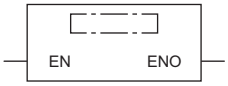
Performing the FOR to NEXT instruction loop

FOR, NEXT

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

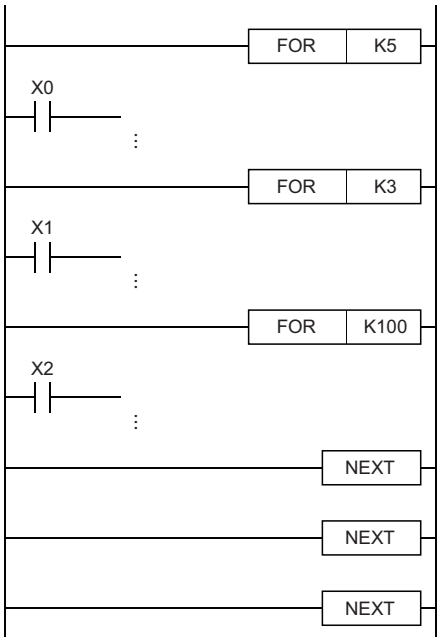
Operand	Description	Range	Data type	Data type (label)
(n)	Number of FOR to NEXT instruction loops	1 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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.



Operation error

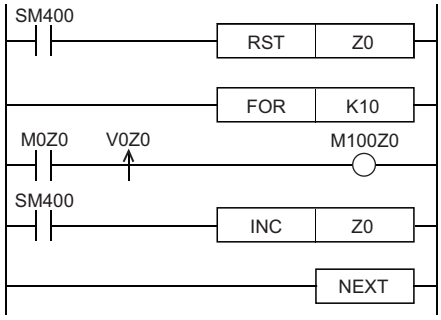
Error code (SD0)	Description
3340H	After execution of the FOR instruction, the END, FEND, or GOEND instruction is executed before the NEXT instruction.
	The STOP instruction is provided within the FOR to NEXT instruction loop.
3341H	The NEXT instruction is executed before the FOR instruction.
3361H	More than 16 FOR instructions are nested. (The 17th instruction is executed.)

Point

- To terminate the FOR to NEXT instruction loop being executed, use the BREAK instruction.

☞ Page 583 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.



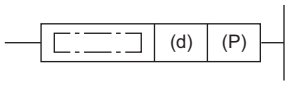
- The JMP instruction cannot be used to branch into the FOR to NEXT instruction loop from the outside.
- To create an easy-to-understand program, use a pair of instructions, the FOR and NEXT instructions, within a single program block.

Forcibly terminating the FOR to NEXT instruction loop

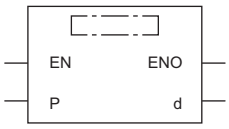
BREAK(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions forcibly terminate the loop processing between the FOR and NEXT instructions, and pass the control to the specified pointer.

Ladder	ST
	Not supported

FBD/LD



■Execution condition

Instruction	Execution condition
BREAK	
BREAKP	

Setting data

■Description, range, data type

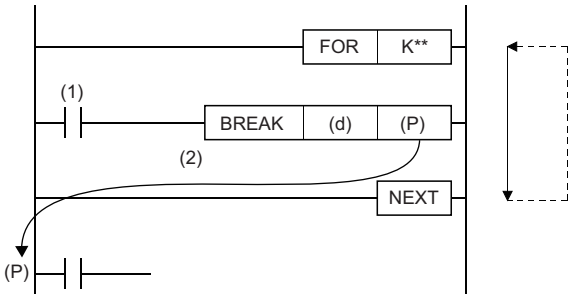
Operand	Description	Range	Data type	Data type (label)
(d)	Number of the device for storing the remaining number of loops	—	16-bit signed binary	ANY16
(P)	Branch destination pointer number at the time of forced termination of loop processing	—	Device name	POINTER
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(P)	—	—	—	—	—	—	—	—	—	—	—	○

Processing details

- These instructions forcibly terminate the loop processing between the FOR and NEXT instructions, and pass the control to the pointer specified by (P). Only a pointer in the same program file can be specified for (P). An operation error occurs if a pointer in another program file is specified.



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.

Operation error

Error code (SD0)	Description
3342H	The instruction is used outside the FOR to NEXT instruction loop.
3380H	The jump destination corresponding to the pointer specified by (P) does not exist.
	A pointer in another program is specified in (P).

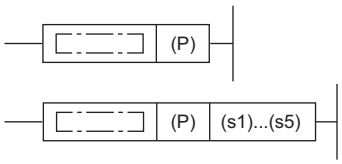
Calling a subroutine program

CALL(P)

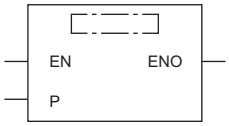
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

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.

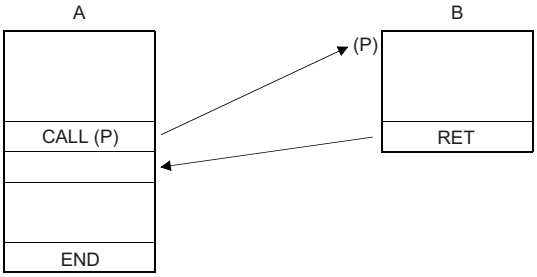
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(P)	—	—	—	—	—	—	—	—	—	—	—	○
(s1) to (s5)	○*1	○	○	○	○	○	○	○	○	—	—	—

*1 Devices other than F can be used.

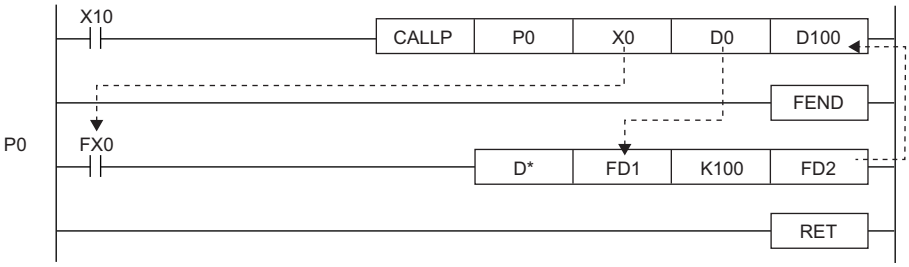
Processing details

- 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).



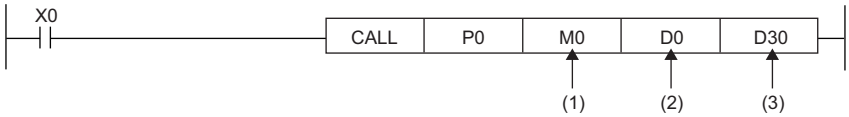
- Before execution of the subroutine program, bit data is transferred to FX and word data is transferred to FD.
- After execution of the subroutine program, the data in FY and FD are transferred to the corresponding devices.
- Function devices FX and FY are processed in units of bits. Function device FD is processed 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	Data size
FX, FY	Bit device	1 point
	Bit-specified word device	1 bit
FD	Digit-specified bit device*2	4 words*3
	Word device	4 words*3

*2 An error does not occur even if the device number specified by (s1) to (s5) is not a multiple of 16.
*3 The data size varies depending on the instruction used.

Ex.
Data in the specified devices



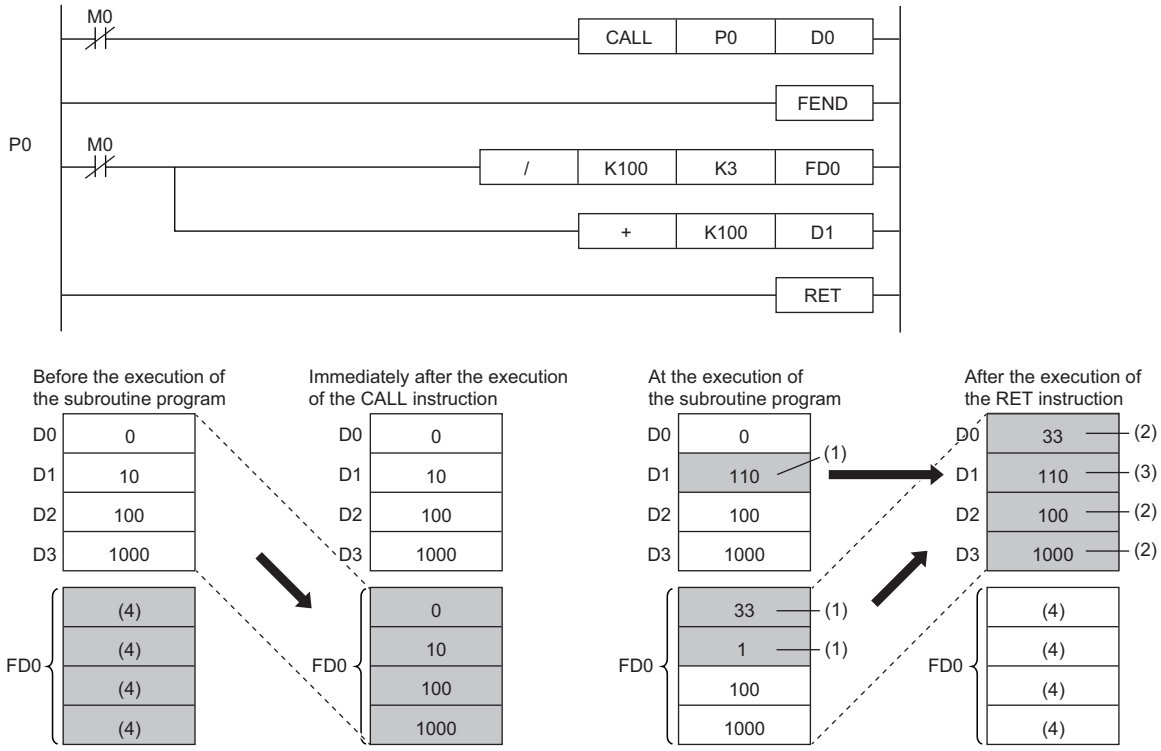
(1) M0 occupied (The data is transferred to FX0.)
(2) D0 to D3 occupied (The data is transferred to FD1.)
(3) D30 to D33 occupied (The data is transferred to FD2.)

- 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.
- 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.
- 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.

- 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.

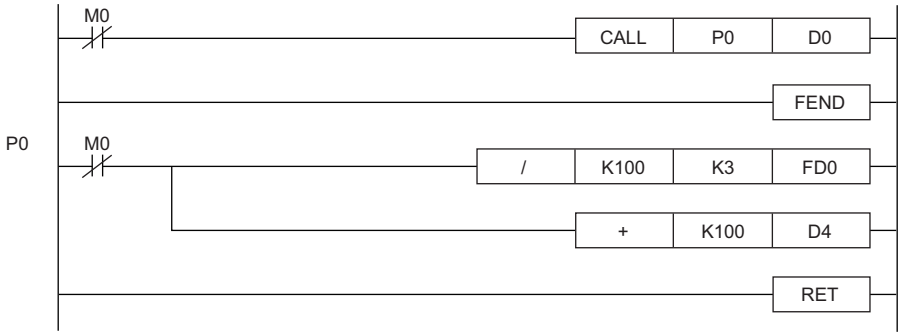
Wrong operation: While D0 is specified for FD0 in the subroutine program, D1 is used in the subroutine program.

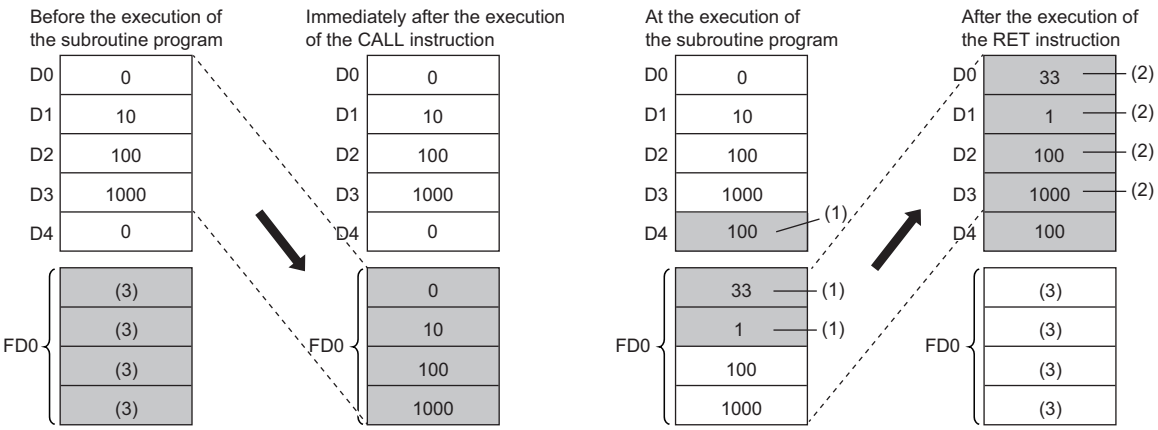


(1) The operation result of the subroutine program is stored.
(2) These values are replaced with the function device values.
(3) The value of D1 is not replaced with the function device value.
(4) Undefined values are stored.

Ex.

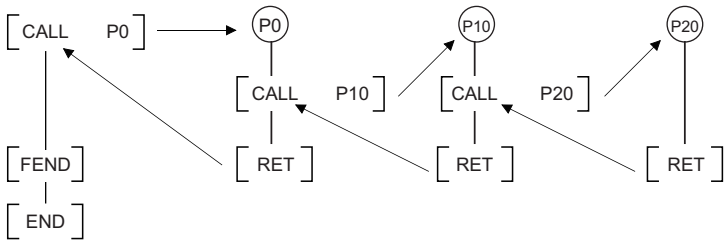
Correct operation: While D0 is specified for FD0 in the subroutine program, D4 is used in the subroutine program.





(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.

Operation error

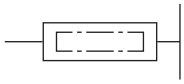
Error code (SD0)	Description
2820H	The device specified in an argument from (s1) to (s5) cannot be secured for the data size.
3360H	More than 16 CALL(P) instructions are nested. (The 17th instruction is executed.)
3380H	The subroutine program corresponding to the pointer specified by (P) does not exist.
3381H	After execution of the CALL(P) instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3382H	The RET instruction is executed before the CALL(P) instruction.

Returning from the subroutine program called

RET

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This instruction indicates the end of a subroutine program.

Ladder	ST
	Not supported

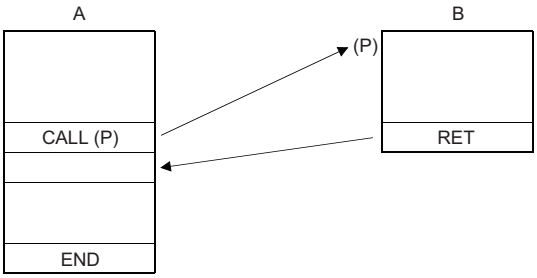
FBD/LD
Not supported

■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
3381H	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.
3382H	The RET instruction is executed before the CALL(P), FCALL(P), ECALL(P), EFCALL(P), or XCALL instruction.

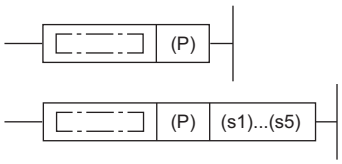
Calling a subroutine program and turning the output off

FCALL(P)

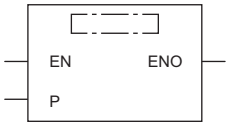
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

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

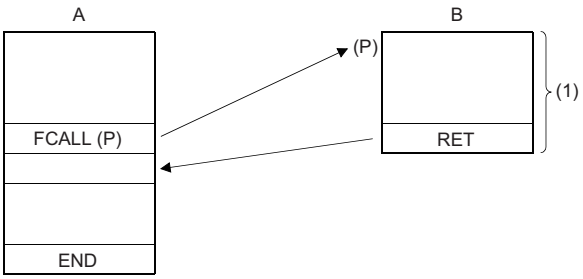
Operand	Bit		Word			Double word		Indirect specification	Constant			Others (P)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(P)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)···(s5)	○*1	○	○	○	○	○	○	○	○	—	—	—

*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.

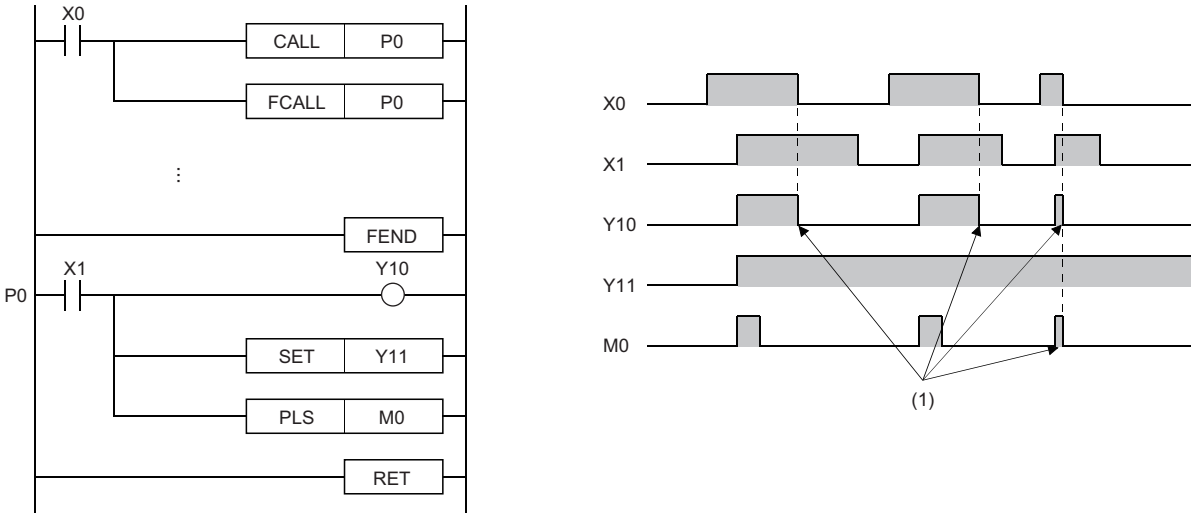
Device used for operation	Operation result (device status)
High-speed timer, low-speed timer	Set to 0
High-speed retentive timer, low-speed retentive timer, counter	Maintains the current status.
Device in OUT instruction	Forcibly turned off.
Device in the SET, RST, SFT(P), or basic/application instruction	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.

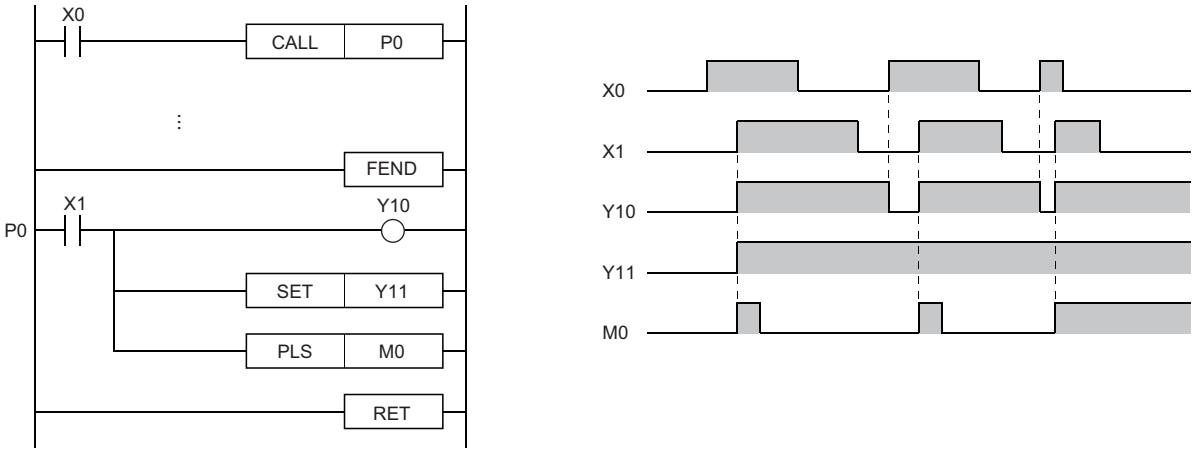
Ex.

When the FCALL(P) instruction is used

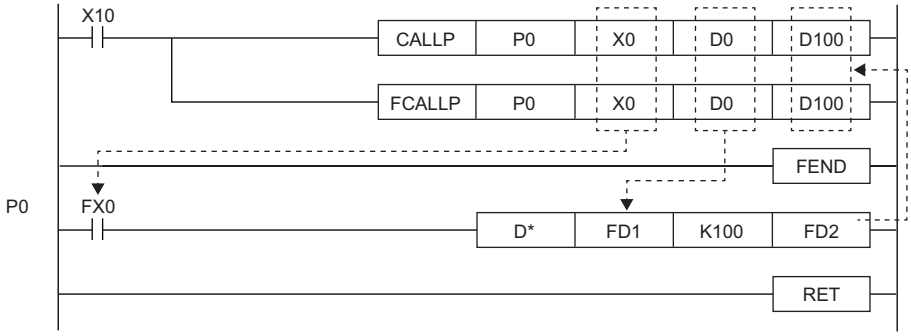


(1) Forced off by the FCALL instruction

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).



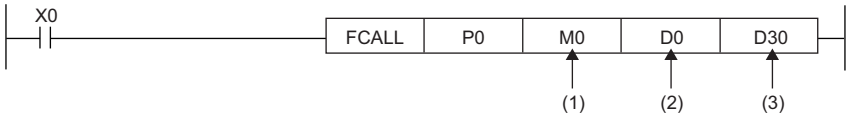
- Before execution of the subroutine program, bit data is transferred to FX and word data is transferred to FD.
- After execution of the subroutine program, the contents of FY and FD are transferred to the corresponding devices.
- Function devices FX and FY are processed in units of bits. Function device FD is processed 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 it cannot be secured for the data size.

- The following table lists the data sizes of individual types of function devices.

Function device	Device	Data size
FX, FY	Bit device	1 point
	When a bit-specified word device is used	1 bit
FD	When the bit device digit is specified*3	4 words*4
	Word device	4 words

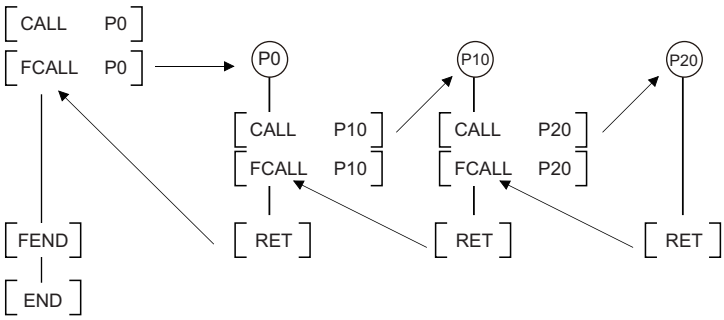
*3 An error does not occur even if the device number specified by (s1) to (s5) is not a multiple of 16 in bit device digit specification mode.
*4 The upper two words of FD are 0.

Ex.
Content of specified device



(1) M0 occupied (The data is transferred to FX0.)
(2) D0 to D3 occupied (The data is transferred to FD1.)
(3) D30 to D33 occupied (The data is transferred to FD2.)

- 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.
3360H	More than 16 FCALL(P) instructions are nested. (The 17th instruction is executed.)
3380H	The subroutine program corresponding to the pointer specified by the FCALL(P) instruction does not exist.
3381H	After execution of the FCALL(P) instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3382H	The RET instruction is executed before the FCALL(P) instruction.

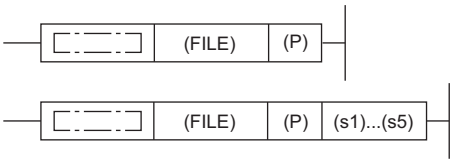
Calling a subroutine program in the specified program file

ECALL(P)

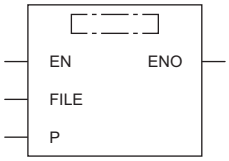
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

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.

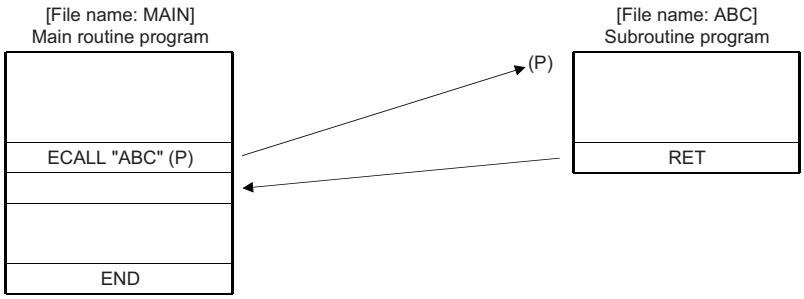
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(File name)	—	—	○	—	—	—	—	○	—	—	○	—
(P)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)···(s5)	○*1	○	○	○	○	○	○	○	○	—	—	—

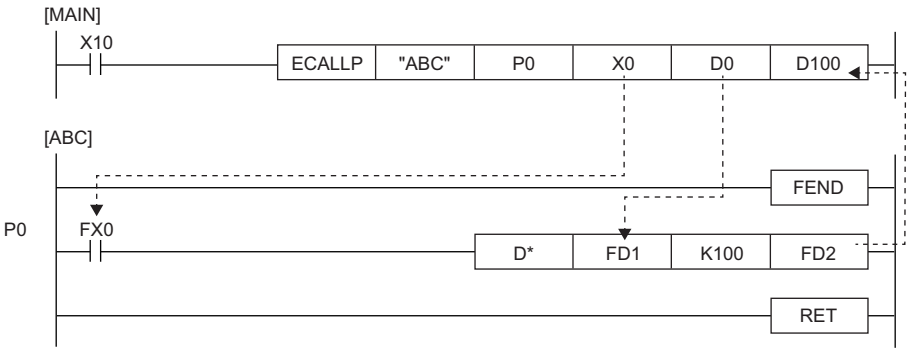
*1 Any value other than F can be used.

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).



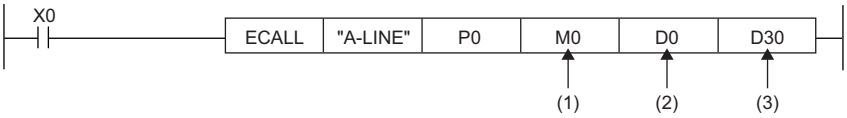
- Before execution of the subroutine program, bit data is transferred to FX and word data is transferred to FD.
- After execution of the subroutine program, the contents of FY and FD are transferred to the corresponding devices.
- Function devices FX and FY are processed in units of bits. Function device FD is processed 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 it cannot be secured for the data size.

- The following table lists the data sizes of individual types of function devices.

Function device	Device	Data size
FX, FY	Bit device	1 point
	When a bit-specified word device is used	1 bit
FD	When the bit device digit is specified*2	4 words*3
	Word device	4 words*3

*2 An error does not occur even if the device number specified by (s1) to (s5) is not a multiple of 16 in bit device digit specification mode.
*3 The data size varies depending on the instruction used.

Ex.
Content of specified device



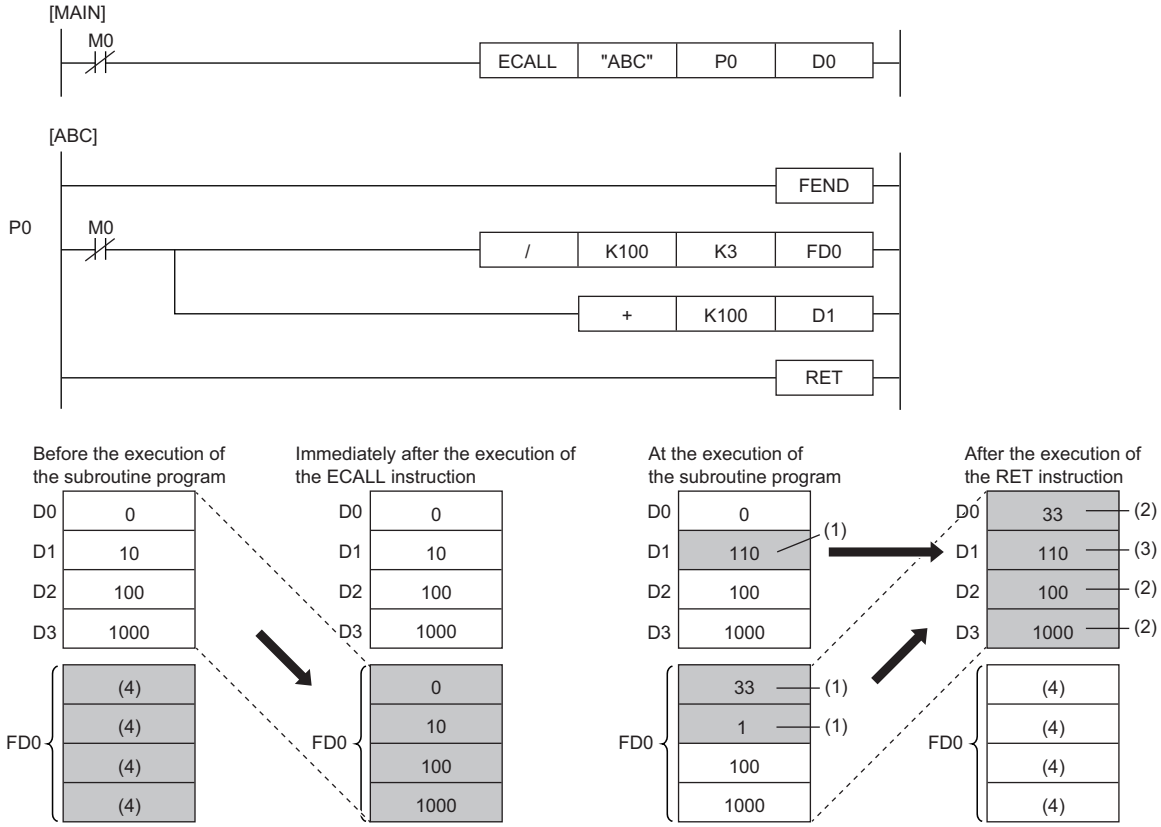
(1) M0 occupied (The data is transferred to FX0.)
(2) D0 to D3 occupied (The data is transferred to FD1.)
(3) D30 to D33 occupied (The data is transferred to FD2.)

- 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.

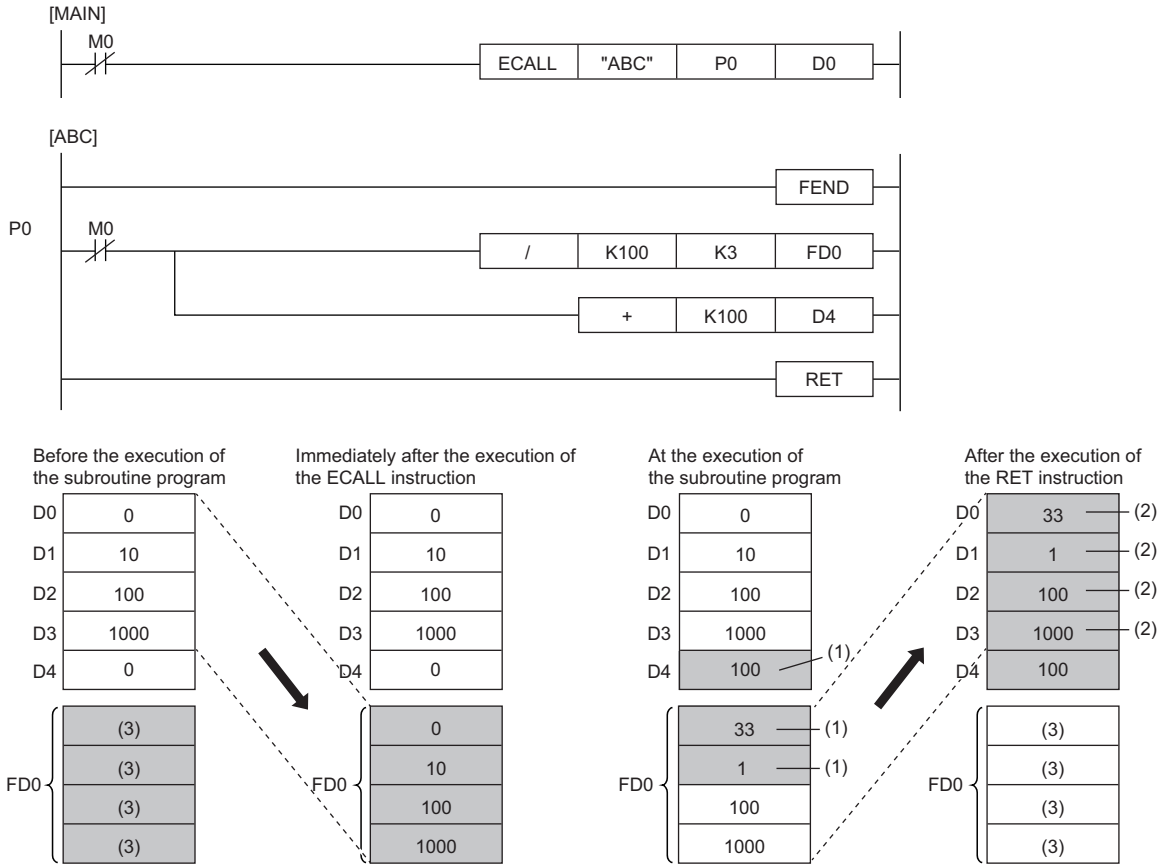
Wrong operation: While D0 is specified for FD0 in the subroutine program, D1 is used in the subroutine program.



(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.

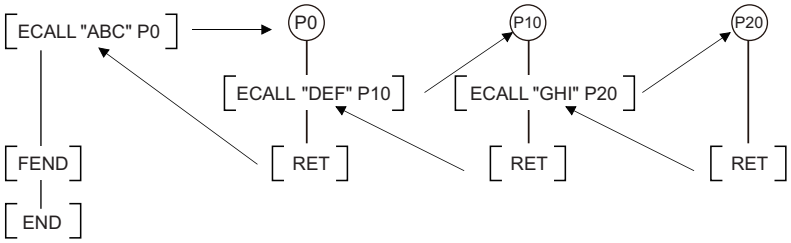
Ex.

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.

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.
3360H	More than 16 ECALL(P) instructions are nested. (The 17th instruction is executed.)
3380H	The subroutine program corresponding to the pointer specified by (P) does not exist.
3381H	After execution of the ECALL(P) instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3382H	The RET instruction is executed before the ECALL(P) instruction.

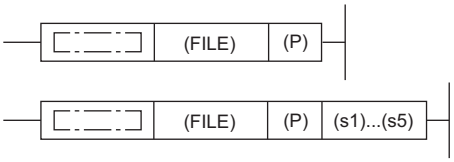
Calling a subroutine program in the specified program file and turning the output off

EFCALL(P)

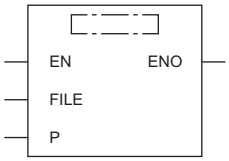
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

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

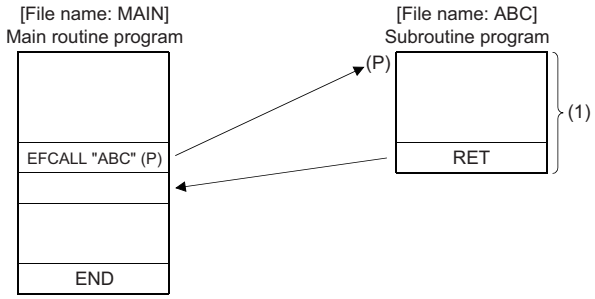
Operand	Bit		Word			Double word		Indirect specification	Constant			Others (P)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(File name)	—	—	○	—	—	—	—	○	—	—	○	—
(P)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)···(s5)	○*1	○	○	○	○	○	○	○	○	—	—	—

*1 Any value other than F can be used.

Processing details

- When the EFCALL(P) instruction is executed, these instructions perform 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.

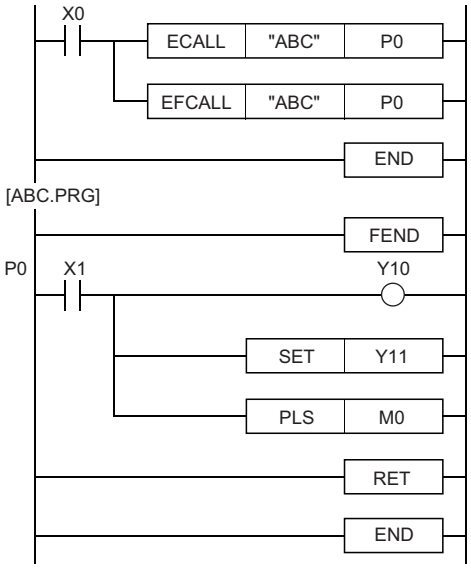
Device used for operation	Operation result (device status)
High-speed timer, low-speed timer	Set to 0
High-speed retentive timer, low-speed retentive timer, counter	Maintains the current status.
Device in OUT instruction	Forcibly turned off.
Device in the SET, RST, SFT(P), or basic/application instruction	Maintains the current status.
PLS or pulse instruction (□P)	Performs the same processing as when the condition contact is off.

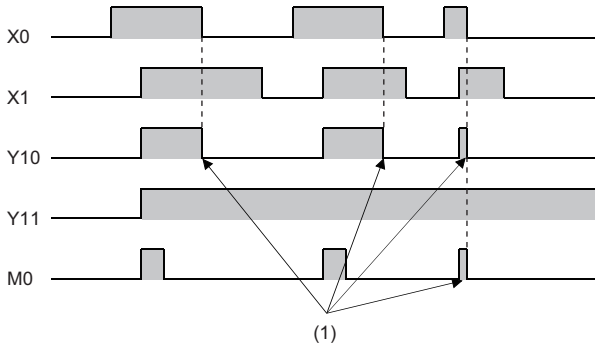
- The EFCALL(P) instruction is used in combination with the ECALL(P) instruction. If the EFCALL(P) instruction is not used in combination with the ECALL(P) instruction, non-execution processing of the subroutine program is not performed even if the execution command is turned off, and therefore the output 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

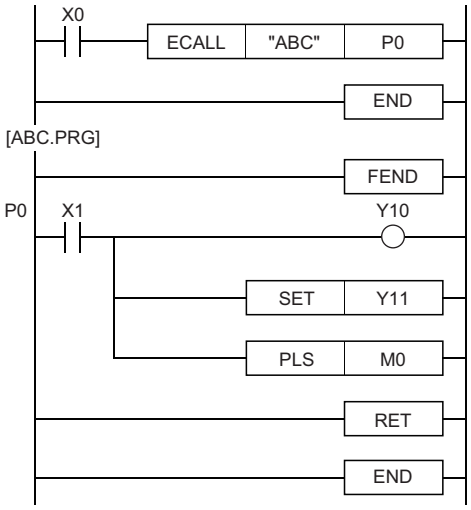


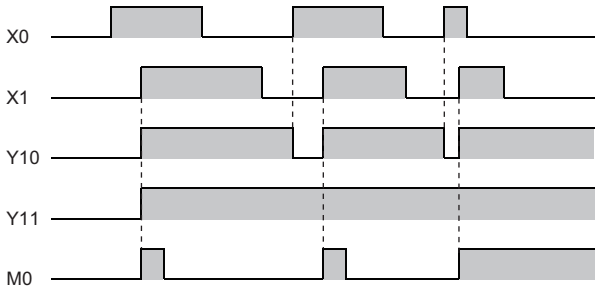


(1) Forced off by the EFCALL instruction

Ex.

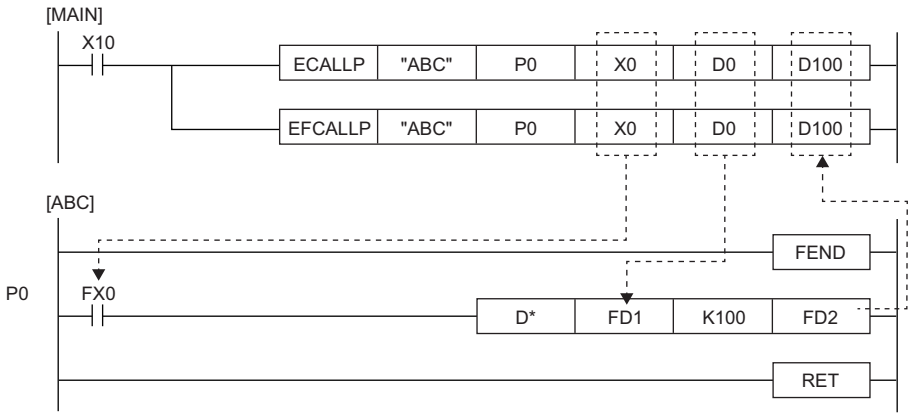
When the EFCALL (P) instruction is not used





- 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).



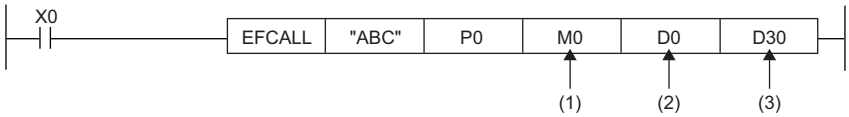
- Before execution of the subroutine program, bit data is transferred to FX and word data is transferred to FD.
- After execution of the subroutine program, the contents of FY and FD are transferred to the corresponding devices.
- Function devices FX and FY are processed in units of bits. Function device FD is processed 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 it cannot be secured for the data size.

- The following table lists the data sizes of individual types of function devices.

Function device	Device	Data size
FX, FY	Bit device	1 point
	When a bit-specified word device is used	1 bit
FD	When the bit device digit is specified*3	4 words*4
	Word device	4 words

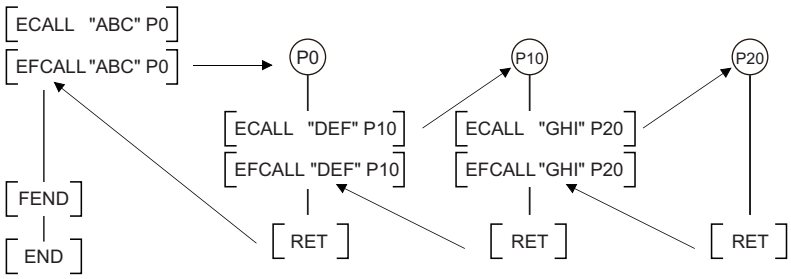
*3 An error does not occur even if the device number specified by (s1) to (s5) is not a multiple of 16 in bit device digit specification mode.
*4 The upper two words of FD are 0.

Ex.
Content of specified device



(1) M0 occupied (The data is transferred to FX0.)
(2) D0 to D3 occupied (The data is transferred to FD1.)
(3) D30 to D33 occupied (The data is transferred to FD2.)

- 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.

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.
3360H	More than 16 EFCALL(P) instructions are nested. (The 17th instruction is executed.)
3380H	The subroutine program corresponding to the pointer specified by (P) does not exist.
3381H	After execution of the EFCALL(P) instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3382H	The RET instruction is executed before the EFCALL(P) instruction.

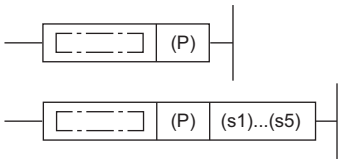
Calling a subroutine program

XCALL

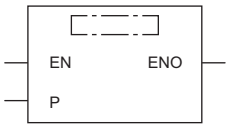
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (MELSEC iQ-R CPU Module User's Manual (Application))

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 is broken, it triggers FCALL.

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 (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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(P)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)···(s5)	○*1	○	○	○	○	○	○	○	○	—	—	—

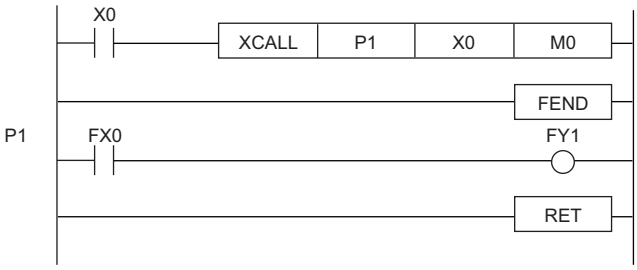
*1 Devices other than F can be used.

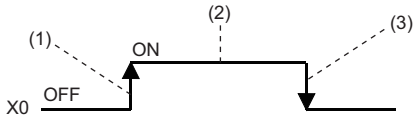
Processing details

- 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.

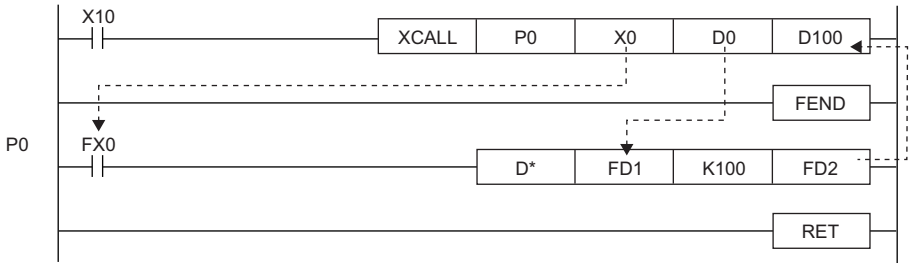
- The following figure shows the operation of the XCALL instruction.





(1) On the rising edge of X0 (off→on): Executes the subroutine program specified by P1.
(2) During X0 is on: Executes the subroutine program specified by P1. ("During X0 is on" does not include the rising edge of X0.)
(3) On the falling edge of X0 (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).



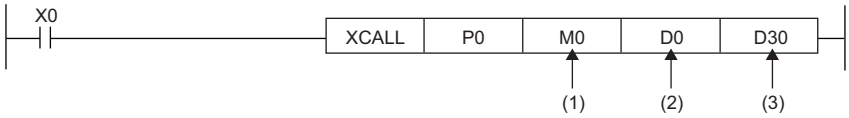
- Before execution of the subroutine program, bit data is transferred to FX and word data is transferred to FD.
- After execution of the subroutine program, the contents of FY and FD are transferred to the corresponding devices.
- Function devices FX and FY are processed in units of bits. Function device FD is processed 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 it cannot be secured for the data size.

- The following table lists the data sizes of individual types of function devices.

Function device	Device	Data size
FX, FY	Bit device	1 point
	When a bit is specified for a word device	1 bit
FD	When the bit device digit is specified*2	4 words*3
	Word device	4 words*3

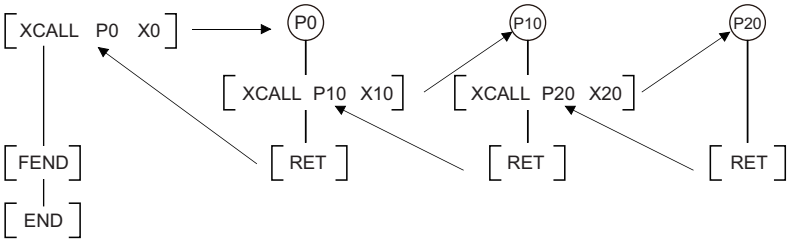
*2 An error does not occur even if the device number specified by (s1) to (s5) is not a multiple of 16 in bit device digit specification mode.
*3 The data size varies depending on the instruction used.

Ex.
Content of specified device



(1) M0 occupied (The data is transferred to FX0.)
(2) D0 to D3 occupied (The data is transferred to FD1.)
(3) D30 to D33 occupied (The data is transferred to FD2.)

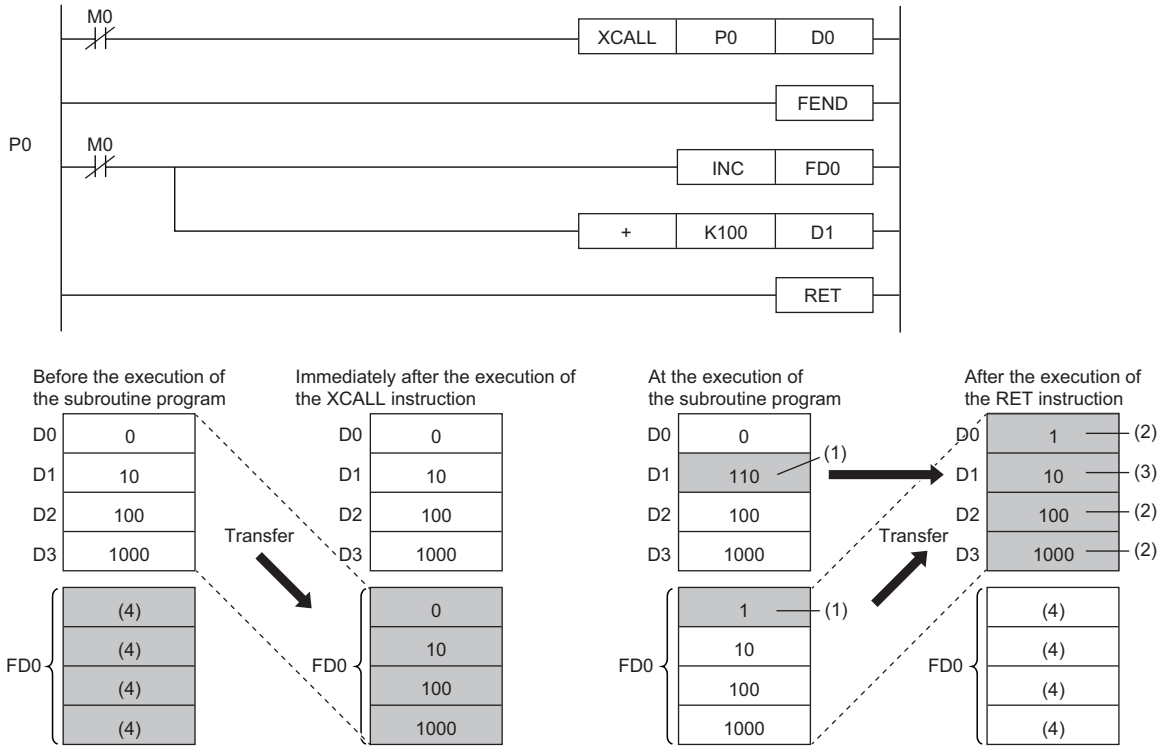
- The XCALL instruction can use (s1) to (s5).
- The number of function devices used in the subroutine program must be identical to the number of arguments of the XCALL instruction. Also, the types of function devices and XCALL instruction arguments must be the same.
- The device numbers specified by 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.

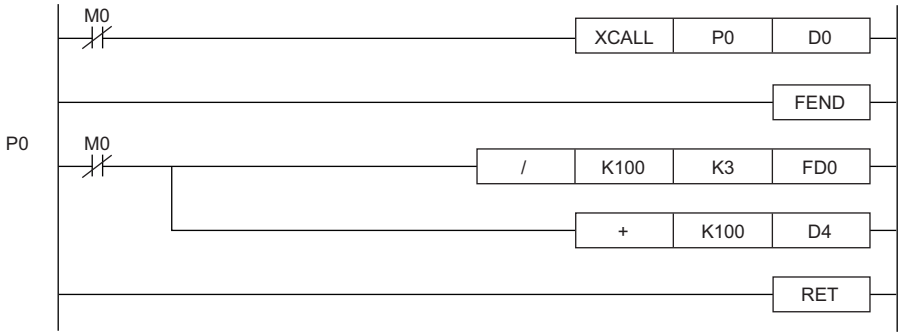
Wrong operation: While D0 is specified for FD0 in the subroutine program, D1 is used in the subroutine program.

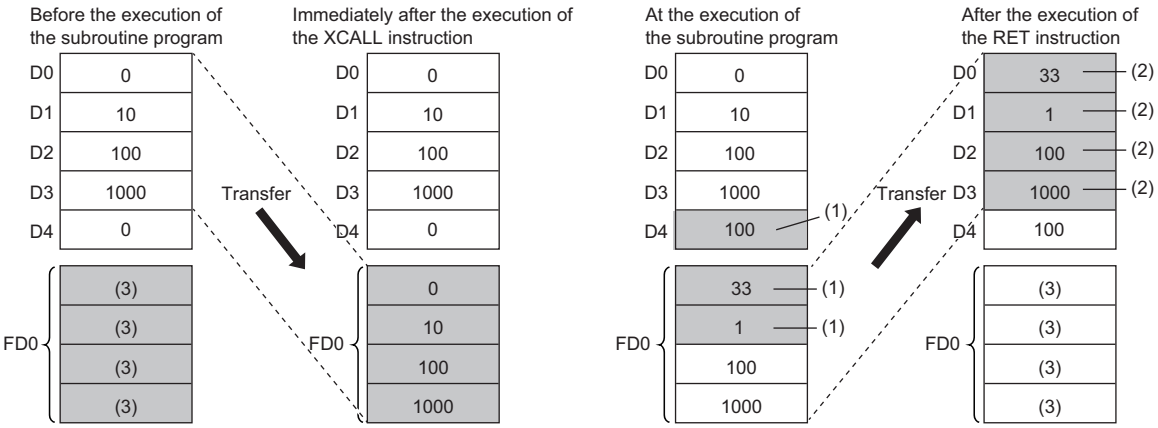


(1) The operation result of the subroutine program is stored.
(2) These values are replaced with the function device values.
(3) The value of D1 is not replaced with the function device value.
(4) Undefined values are stored.

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.

Operation error

Error code (SD0)	Description
2820H	The device specified in an argument from (s1) to (s5) cannot be secured for the data size.
3360H	More than 16 XCALL instructions are nested. (The 17th instruction is executed.)
3380H	The subroutine program corresponding to the pointer specified by (P) does not exist.
3381H	After execution of the XCALL instruction, the END, FEND, GOEND, or STOP instruction is executed before the RET instruction.
3382H	The RET instruction is executed before the XCALL instruction.

7.4 Program Control Instructions

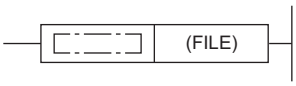
Changing the program execution type to standby type

PSTOP(P)

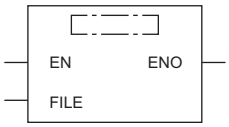
RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, the execution type is not taken over when the systems are switched. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These instructions change the execution type of the program with the file name stored in the specified device to a standby type.

Ladder	ST
 FILE: File name	ENO:=PSTOP(EN,filename); ENO:=PSTOPP(EN,filename);

FBD/LD



FILE: File name

■Execution condition

Instruction	Execution condition
PSTOP	
PSTOPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(File name)	Character string data of the file name of the program to be changed to a standby type, or the start device where the character string data is stored	—	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(File name)	—	—	○	—	—	—	—	○	—	—	○	—

Processing details

- These instructions change the execution type of the program with the file name stored in the device specified by (file name) to a standby type.
- Only programs stored in program memory can be changed to a standby type.
- The execution type of the specified program changes to a standby type during END processing.
- The PSTOP(P) instruction takes precedence even when the execution type is specified in parameter.
- Extension ".PRG" does not need to be specified as a part of file name. (Only .PRG files can be processed by these instructions.)

Operation error

Error code (SD0)	Description
2840H	The program with the file name specified by (file name) does not exist.
2841H	The program with the file name specified by (file name) is not registered in parameter.
2842H	The type of the program with the file name specified by (file name) is the SFC program.

Point

For how to change the program execution type, refer to the following.

MELSEC iQ-R CPU Module User's Manual (Application)

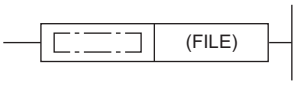
Changing the program execution type to standby type (output off)

POFF(P)

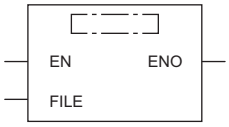
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, the execution type is not taken over when the systems are switched. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- [Process CPU (redundant mode)] These instructions cannot be used in SFC programs.

These instructions change the program execution type of the program with the file name stored in the specified device.

Ladder	ST
 FILE: File name	ENO:=POFF(EN,filename); ENO:=POFFP(EN,filename);

FBD/LD



FILE: File name

■Execution condition

Instruction	Execution condition
POFF	
POFFP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(File name)	File name of the program to be changed to the standby type (with output set to off), or the device where the file name is stored	—	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(File name)	—	—	○	—	—	—	—	○	—	—	○	—

Processing details

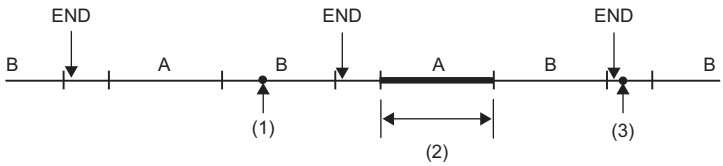
- These instructions change the execution type of the program with the file name stored in the device specified by (file name). If the program is a scan execution type, the output is turned off (non-execution processing) in the next scan. The program will be a standby type in the following scan and later. When the program is a fixed scan execution type or event execution type, it becomes a scan execution type in the next scan and turns off (non-execution processing) the output. The program will be a standby type in the following scan and later.
- Only programs stored in program memory can be changed to a standby type.
- The POFF(P) instruction takes precedence even when the execution type is specified in parameter.
- Extension ".PRG" does not need to be specified as a part of file name. (Only .PRG files can be processed by these instructions.)

Operation error

Error code (SD0)	Description
1BB0H	The program with the file name specified by (file name) is a scan execution type SFC program. (Only for the Process CPU (redundant mode))
2840H	The program with the file name specified by (file name) does not exist.
2841H	The program with the file name specified by (file name) is not registered in parameter.
2842H	The type of the program with the file name specified by (file name) is not supported.

Point

Non-execution processing is the same as the processing performed by each coil instruction with the condition contact set to off.



A, B: Program name

(1)Specify the program A, and execute the POFF instruction.

(2)The program A is not executed.

(3)The program A changes to a standby type program.

The operation results of each coil instruction after the non-execution processing will be as follows, regardless of the on/off state of the condition contact.

- OUT instruction: The output is forcibly turned off.
- SET, RST, SFT, basic, and application instructions: Status is held.
- PLS and PLS conversion instructions (□P): Same processing as when the condition contact is set to off
- OUT T instruction: The current value of the low-speed/high-speed timer is 0.
- OUT ST and OUT C instructions: Current value is held.

For how to change the program execution type, refer to the following.

📖 MELSEC iQ-R CPU Module User's Manual (Application)

For the operation when the SFC program is specified, refer to the following.

📖 MELSEC iQ-R Programming Manual (Program Design)

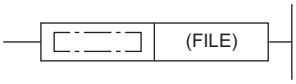
Changing the program execution type to scan execution type

PSCAN(P)

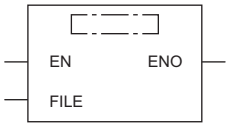
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, the execution type is not taken over when the systems are switched. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- [Process CPU (redundant mode)] These instructions cannot be used in SFC programs.

These instructions change the execution type of the program with the file name stored in the specified device to a scan execution type.

Ladder	ST
 FILE: File name	ENO:=PSCAN(EN,filename); ENO:=PSCANP(EN,filename);

FBD/LD



FILE: File name

■Execution condition

Instruction	Execution condition
PSCAN	
PSCANP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(File name)	File name of the program to be changed to a scan execution type, or the start device where the file name is stored	—	Unicode string	ANYSTRING_DOUBLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

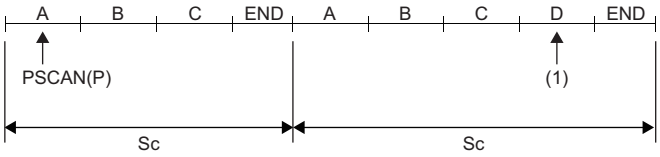
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(File name)	—	—	○	—	—	—	—	○	—	—	○	—

Processing details

- These instructions change the execution type of the program with the file name stored in the device specified by (filename) to a scan execution type.
- Only programs stored in program memory can be changed to a scan execution type.
- The execution type of the specified program changes to a scan execution type during END processing.

Ex.

While there are programs A, B, and C, the PSCAN(P) instruction is executed for program D within program A.



Sc: Scan

(1) The program D is executed.

- The PSCAN(P) instruction takes precedence even when the execution type is specified in parameter.
- Extension ".PRG" does not need to be specified as a part of file name. (Only .PRG files can be processed by these instructions.)

Operation error

Error code (SD0)	Description
1BB0H	The program with the file name specified by (file name) is a standby type SFC program. (Only for the Process CPU (redundant mode))
2840H	The program with the file name specified by (file name) does not exist.
2841H	The program with the file name specified by (file name) is not registered in parameter.
2842H	The type of the program with the file name specified by (file name) is not supported.
3424H	The file name specified by (file name) is the one for an SFC program and an SFC program with another file name is already operating.

Point

For how to change the program execution type, refer to the following.

MELSEC iQ-R CPU Module User's Manual (Application)

For the operation when the SFC program is specified, refer to the following.

MELSEC iQ-R Programming Manual (Program Design)

8 DATA PROCESSING

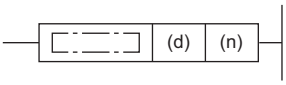
8.1 Rotation Instructions

Rotating 16-bit binary data to the right

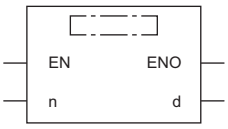
ROR(P), RCR(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- 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 1475 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

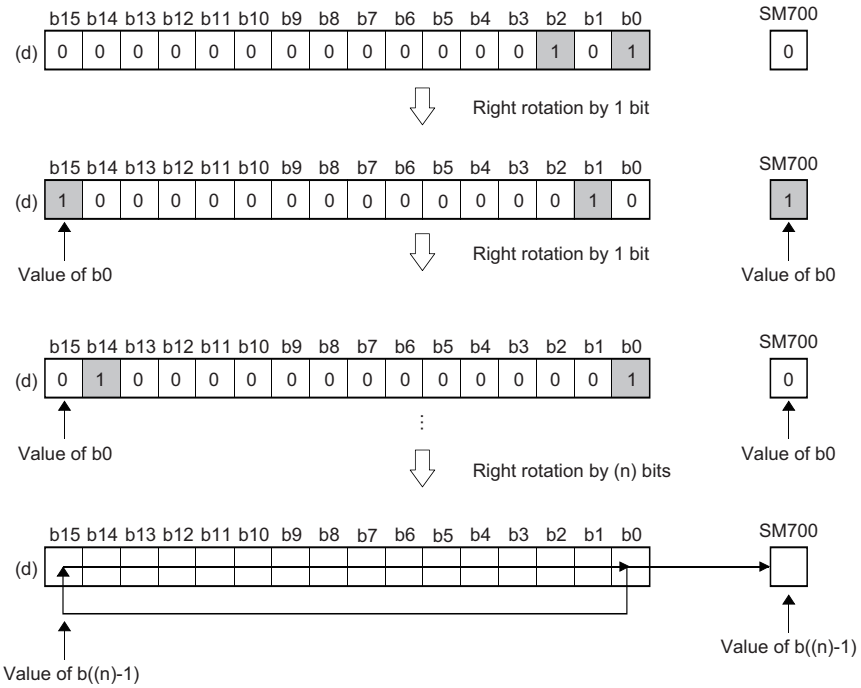
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

■ROR(P)

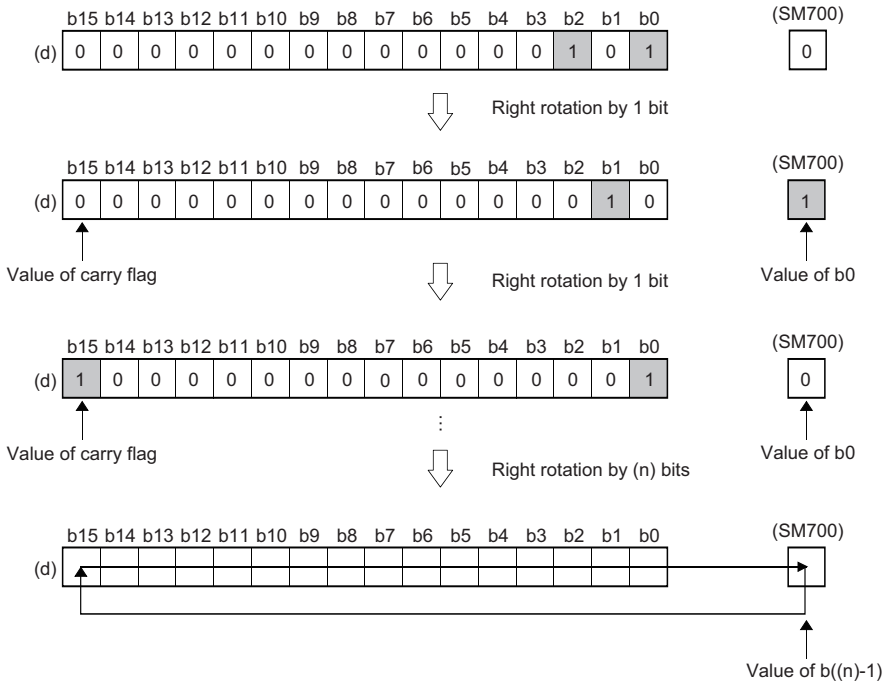
- These instructions rotate the 16-bit binary data in the device specified by (d) to the right by (n) bit(s), excluding the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the ROR(P) instruction.



- 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

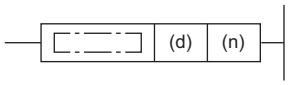
There is no operation error.

Rotating 16-bit binary data to the left

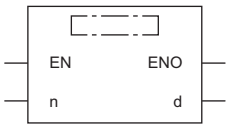
ROL(P), RCL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- 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 1473 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

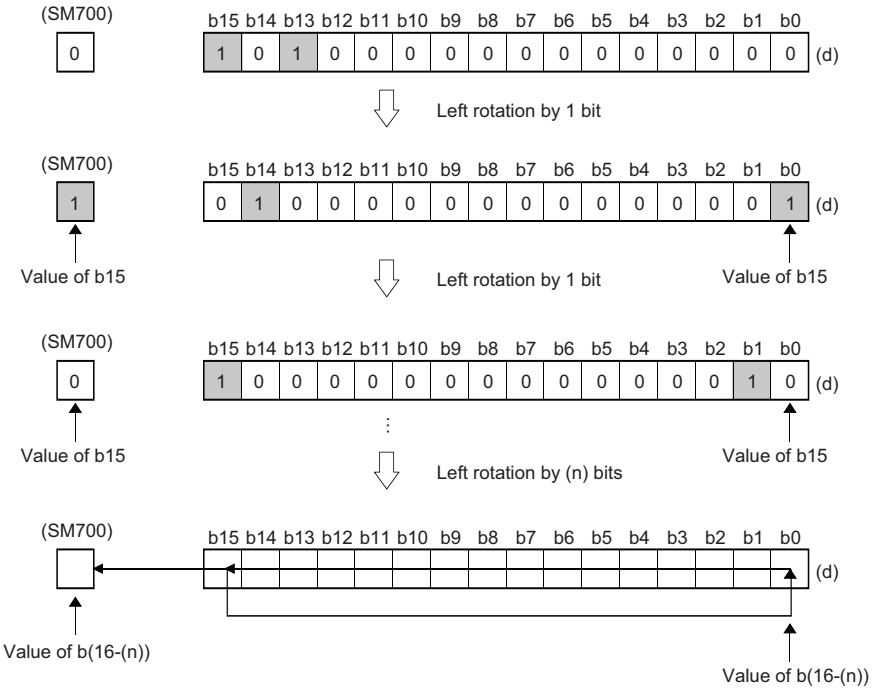
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

■ROL(P)

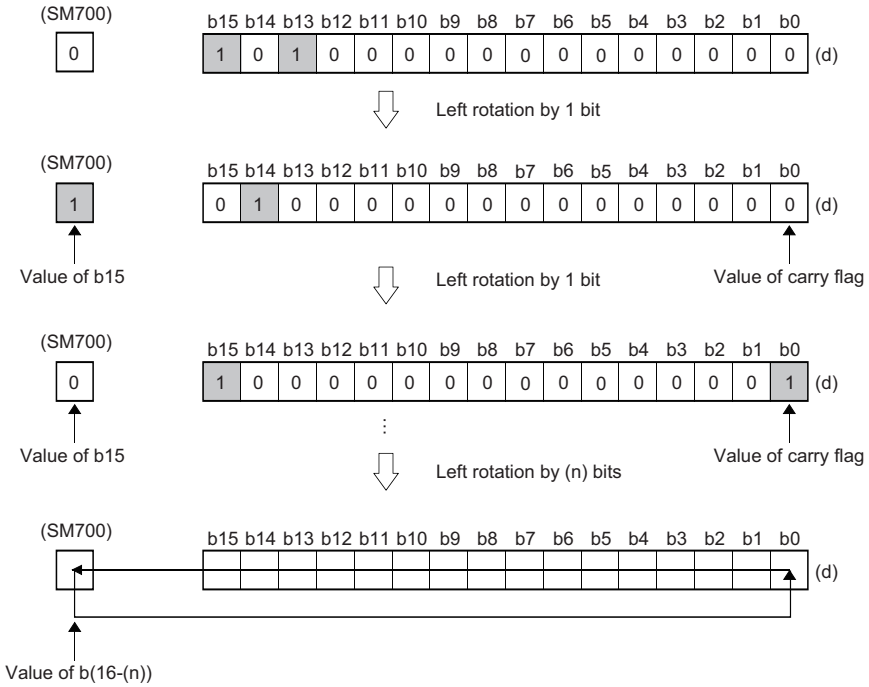
- These instructions rotate the 16-bit binary data in the device specified by (d) to the left by (n) bit(s), excluding the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the ROL(P) instruction.



- 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

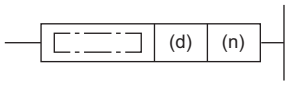
There is no operation error.

Rotating 32-bit binary data to the right

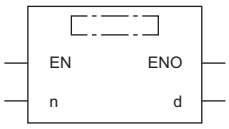
DROR(P), DRCR(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- 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 1475 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

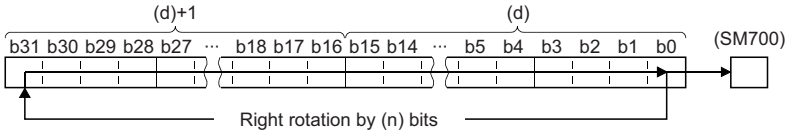
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	○	○	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

■DROR(P)

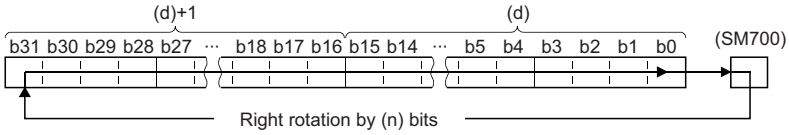
- These instructions rotate the 32-bit binary data in the device specified by (d) to the right by (n) bit(s), excluding the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the DROR(P) instruction.



- 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

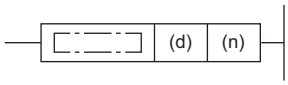
There is no operation error.

Rotating 32-bit binary data to the left

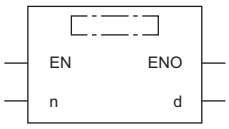
DROL(P), DRCL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- 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 1473 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

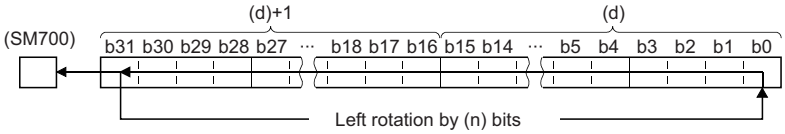
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	○	○	○	○	○	○	○	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

■DROL(P)

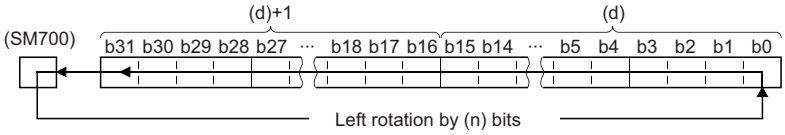
- These instructions rotate the 32-bit binary data in the device specified by (d) to the left by (n) bit(s), excluding the carry flag (SM700). The carry flag (SM700) is on or off depending on the status prior to the execution of the DROL(P) instruction.



- 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.

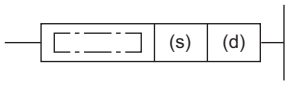
8.2 Data Table Operation Instructions

Reading the oldest data from the data table

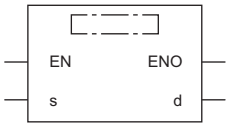
FIFR(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

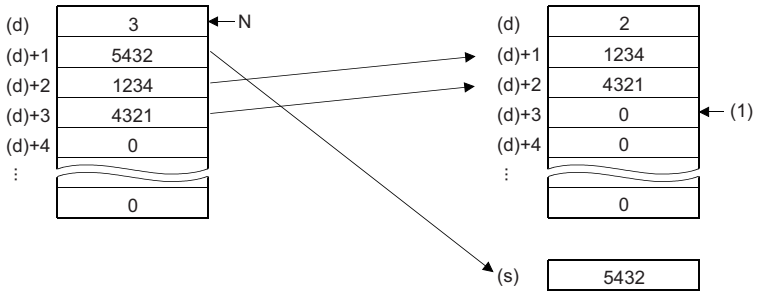
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

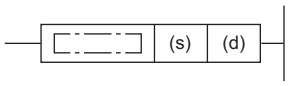
Error code (SD0)	Description
3405H	The FIFR(P) instruction is executed when the value in the device specified by (d) is 0.

Reading the newest data from the data table

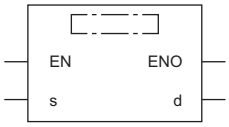
FPOP(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

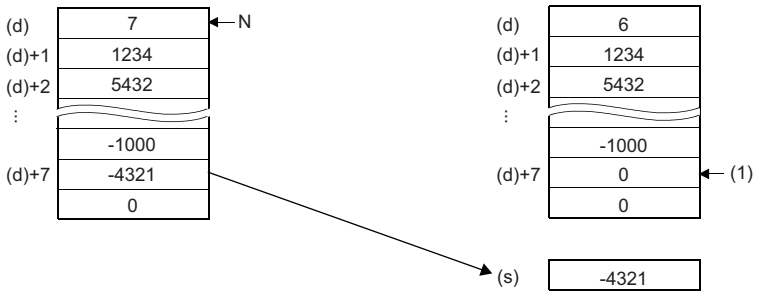
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

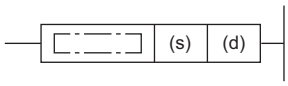
Error code (SD0)	Description
3405H	The FPOP(P) instruction is executed when the value in the device specified by (d) is 0.

Writing data to the data table

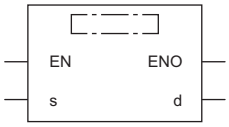
FIFW(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

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

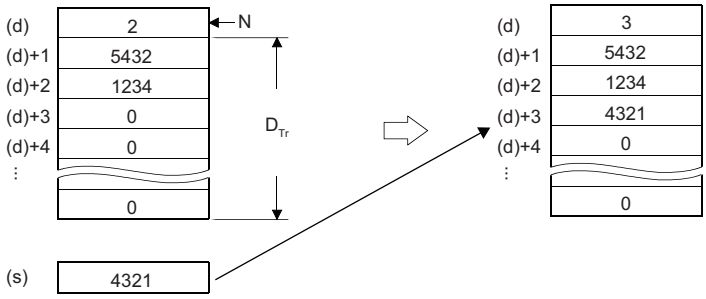
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

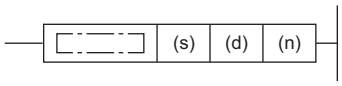
Error code (SD0)	Description
3405H	The FIFW(P) instruction is executed when the value in the device specified by (d) is FFFFH.

Inserting data to the data table

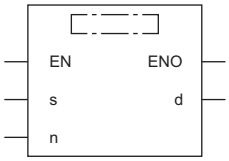
FINS(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

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

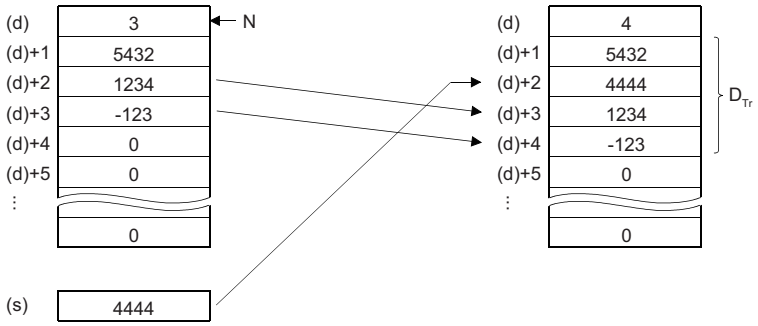
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

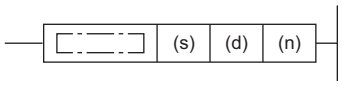
Error code (SD0)	Description
3405H	The FINS(P) instruction is executed when the value specified by (n) is 0.
	The FINS(P) instruction is executed when the value in the device specified by (d) is FFFFH.

Deleting data from data table

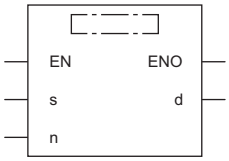
FDEL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

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

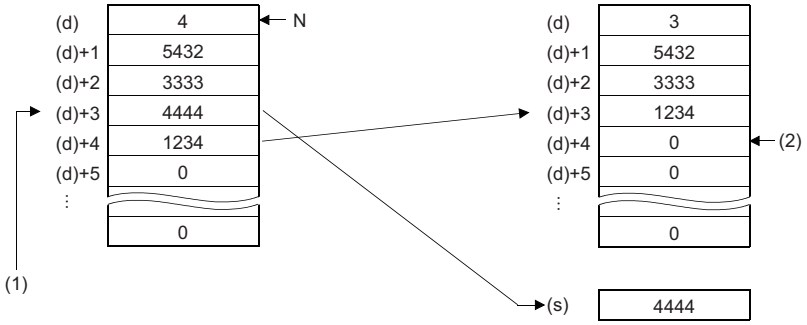
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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
3405H	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.

8.3 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

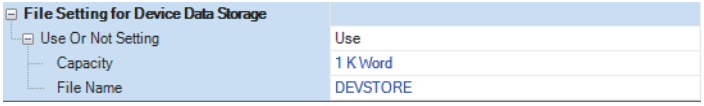
Use the SP.DEVST instruction to write device data to data memory.

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.

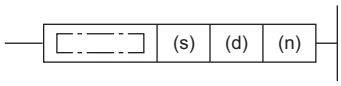
When the CPU module is switched from STOP to RUN, it is checked to see whether the actual device data storage file matches the parameter setting. If they do not match, an error (error code: 21A0H) occurs.

Reading data from the data memory

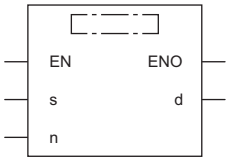
S(P).DEVLD

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

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

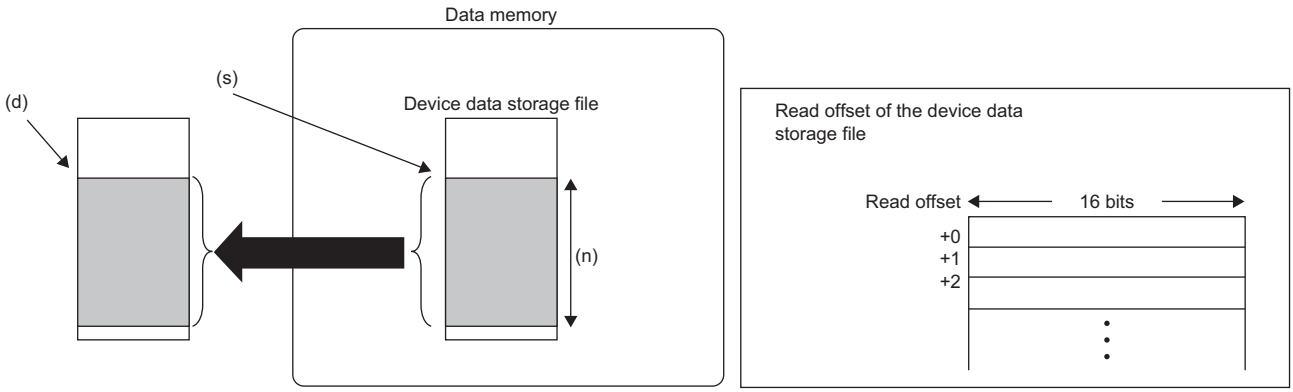
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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 635 Setting method)
- Use the SP.DEVST instruction to write device data to the device data storage file. (☞ Page 638 SP.DEVST)

Operation error

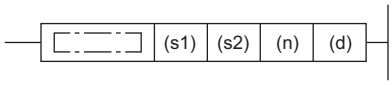
Error code (SD0)	Description
2840H	The device data storage file is not set in parameter.
3405H	The value specified by (n) is 0.

Writing data to the data memory

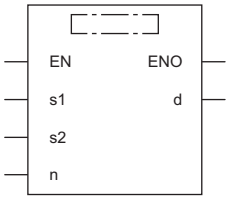
SP.DEVST

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

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

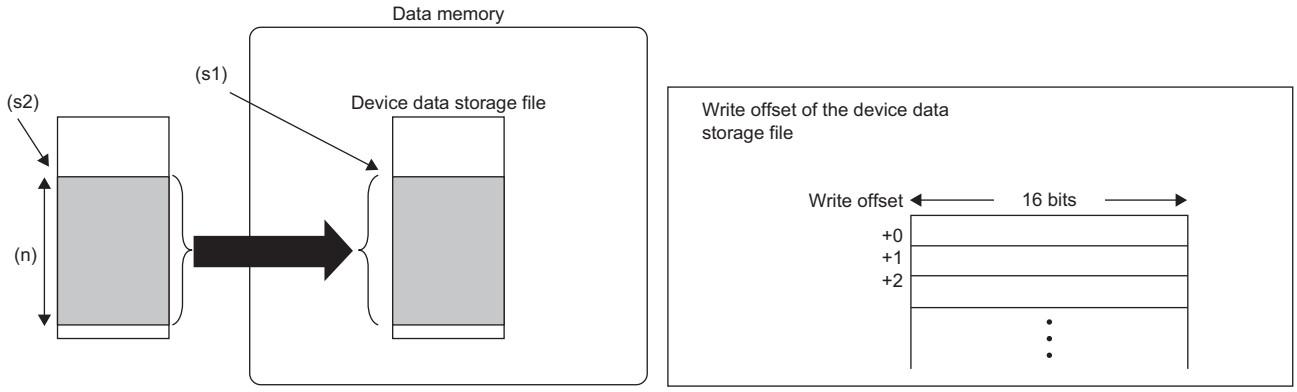
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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 635 Setting method)
- Use the S(P).DEVLD instruction to read device data from the device data storage file to any specified device. (☞ Page 636 S(P).DEVLD)

Precautions

- The value written to data memory is the one at execution of the SP.DEVST instruction.
- Execution of the SP.DEVST instruction increases SD634 and SD635. The number of writes to the data memory of the CPU module is limited. If the data memory write count index exceeds 100,000, an error occurs with error code 1080H.
- 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 3421H. 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.
  - CPU module 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.
3405H	The value specified by (n) is 0.
3421H	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.

8.4 File Operation Instructions

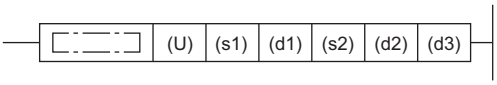
Reading data from the specified file

SP.FREAD

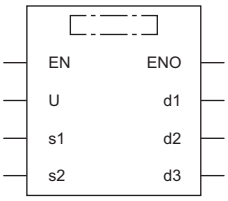
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- The R00CPU does not support this instruction.

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 643 Type 1 ☞ Page 645 Type 2	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(d3)	○	—	○	—	—	—	—	—	—	—	—	—

The control data (d1), file name (s2), and read data (d2) vary depending on the CPU module to be used.

CPU module to be used			Reference
RnCPU	R01CPU, R02CPU		Page 643 Type 1
	R04CPU, R08CPU, R16CPU, R32CPU, R120CPU	Firmware version "28" or later	Page 643 Type 1
		Firmware version "27" or earlier	Page 645 Type 2
RnENCPU		Firmware version "28" or later	Page 643 Type 1
		Firmware version "27" or earlier	Page 645 Type 2
RnPCPU			Page 645 Type 2
RnPSFCPU			Page 645 Type 2
RnSFCPU			Page 645 Type 2

■Type 1

• Control data (d1)

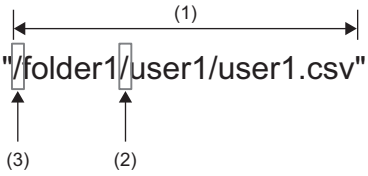
Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution/ completion type	Specify the execution type. ■00**H: Reading binary data • 0000H: 16-bit binary data • 0001H: 32-bit binary data ■01**H: Reading data after converted to CSV format • 0100H: Decimal (16-bit data) • 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 700 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)*2*3 • 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)*2*3 • 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*2 • 2: Unit of the data type specified by (d1)+0 • 3: Odd number of bytes*1*2*3 "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 To specify "3: Odd number of bytes", use the following models. Note that the applicable firmware version varies depending on models.
· R01CPU and R02CPU: "19" or later
· R04CPU, R08CPU, R16CPU, R32CPU, R120CPU: "51" or later
· RnENCPU: "51" or later

*2 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.

*3 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 character 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 must be within 60 characters (a period and extension excluded). If 61 or more characters are used, 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

■Type 2

• Control data (d1)

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution/completion type	Specify the execution type. • 0000H: Reading binary data • 0100H: Reading data after converted to CSV format	0000H 0100H	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 700 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). Even when "1: Even number of bytes" is specified by (d1)+7, specify the number in units of words.	1 to 65535*1	User
+3	Not used	—	—	—
+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	0, 1	User

*1 When "1: Even number of bytes" is specified by (d1)+7 (Data type specification), the setting range is 1 to 32767.

• File name (s2)

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +□	File name character string	Specify the string of the file name. When omitting an extension, omit the ". (period)" as well. Specify the name within 60 characters (a period and extension excluded). If 61 or more characters are used, the extension is ignored and replaced with ".BIN" or ".CSV".	Unicode string	User

• Read data (d2)

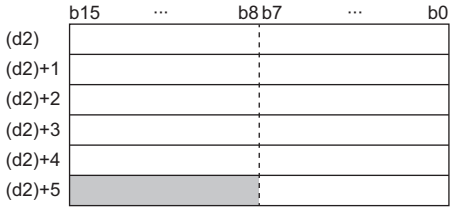
Operand: (d2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Number of data actually read	The number of data actually read is set. The data unit is determined by (d1)+7.	—	System
+1 to +□	Read data	The read data is stored.	—	System

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.

CPU module to be used			Reference
RnCPU	R01CPU, R02CPU		Page 643 Type 1
	R04CPU, R08CPU, R16CPU, R32CPU, R120CPU	Firmware version "28" or later	Page 643 Type 1
		Firmware version "27" or earlier	Page 645 Type 2
RnENCPU		Firmware version "28" or later	Page 643 Type 1
		Firmware version "27" or earlier	Page 645 Type 2
RnPCPU			Page 645 Type 2
RnPSFCPU			Page 645 Type 2
RnSFCPU			Page 645 Type 2

- 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)		

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 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)		

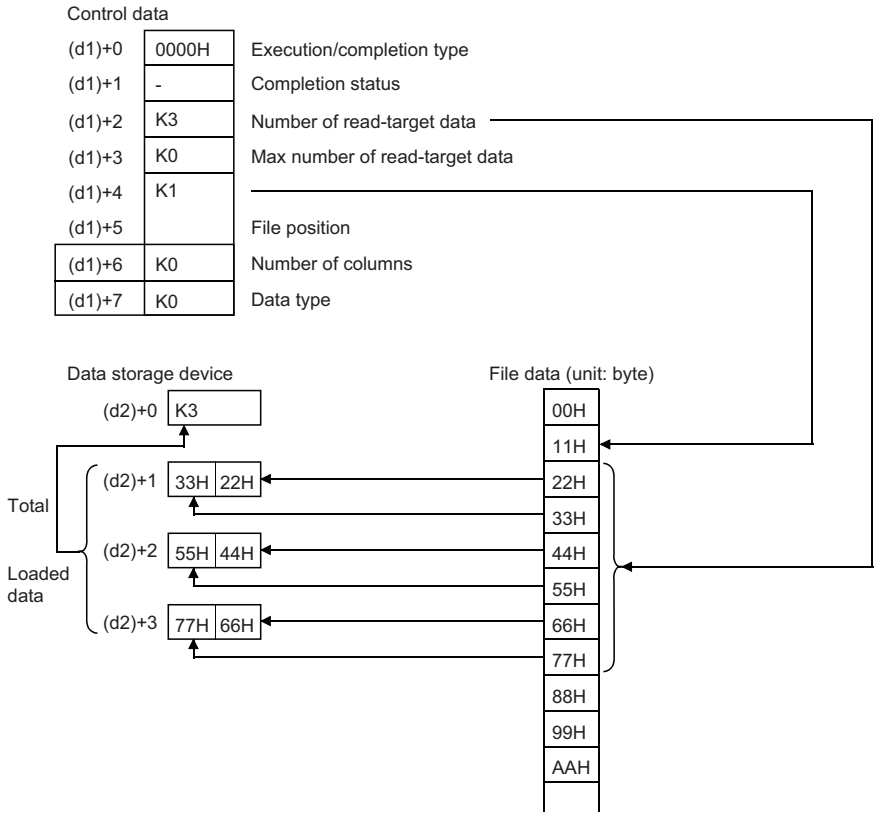
■When reading binary data

- 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.

Execution/completion type	Data (one element) in the CSV file	Value stored in the device	Remarks
0140H: Floating point real number (single-precision real number)	Values within the range of: -2^{128} < data ≤ -2^{-126}, 0, 2^{-126} ≤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} < data ≤ -2^{-1022}, 0, 2^{-1022} ≤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

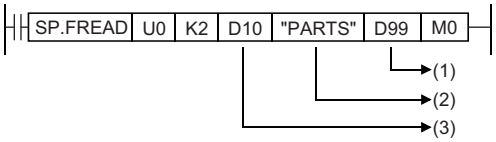
[Data stored in CSV format]

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]

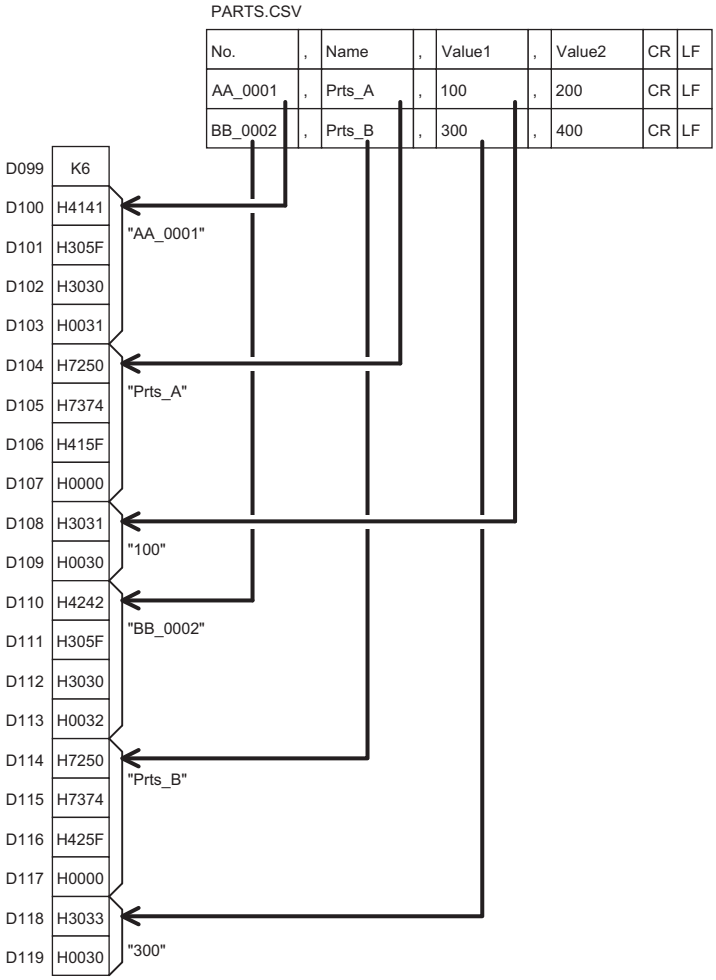


(1) Loaded data
(2) File name
(3) Control data

[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 position
D16: Number of columns
D17: Data type specification



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

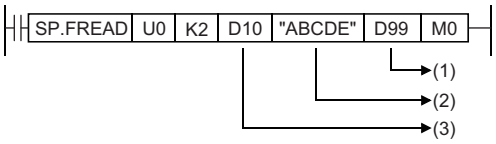
[Data created in Excel]

	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 position
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.

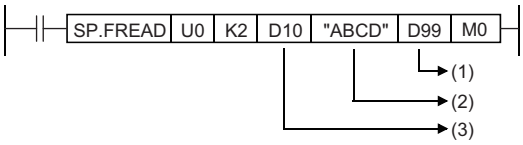
Ex.

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 position
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]

SP.FREAD	U0	K2	D10	"ABCD"	D99	M0

(2)
(3)
(4)

(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 position
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

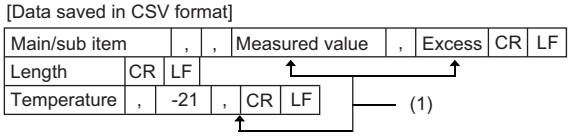
D99: Number of data actually read
D100: Since "Main/sub item" is not a numerical value, the conversion data, 0, is stored.
D101: Since " " is not a numerical value, the conversion data, 0, is stored.
D102: Since "Length" is not a numerical value, the conversion data, 0, is stored.
D103: Since "1" is 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.

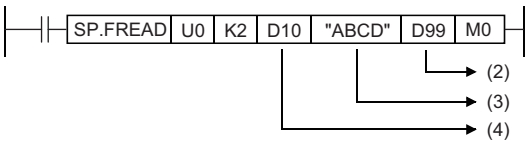
When the number of columns differs in each row

[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 position
D16: Number of columns
D17: Data type specification

[Loaded data]

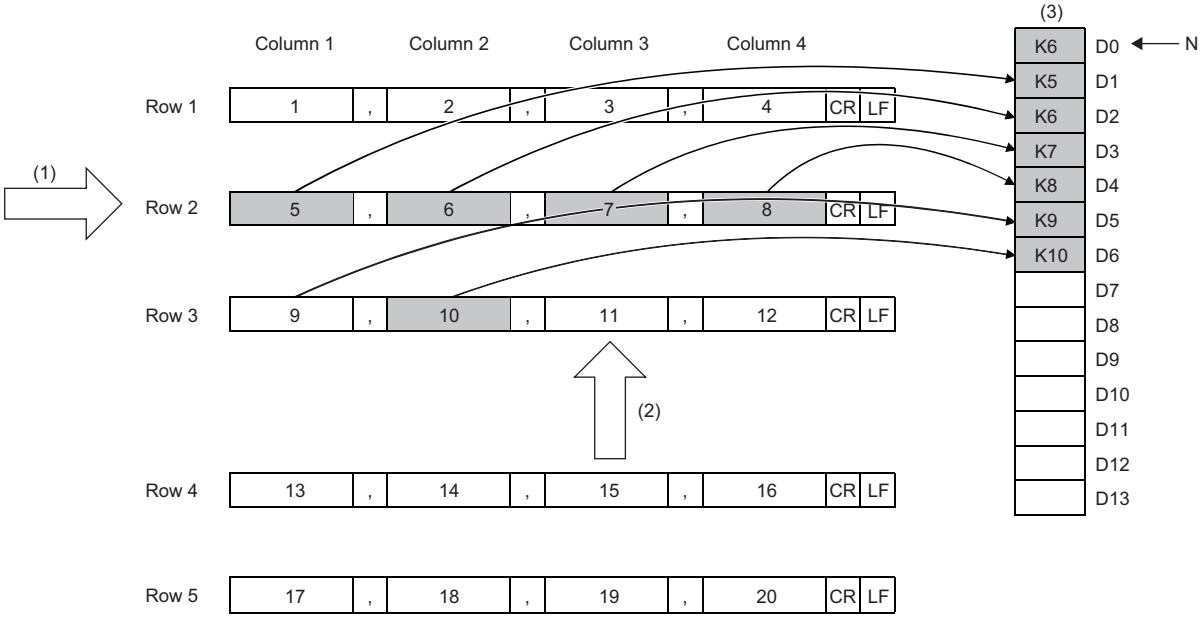
D99	K6
D100	K0
D101	K0
D102	K0
D103	K0
D104	K0
D105	K-21

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 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

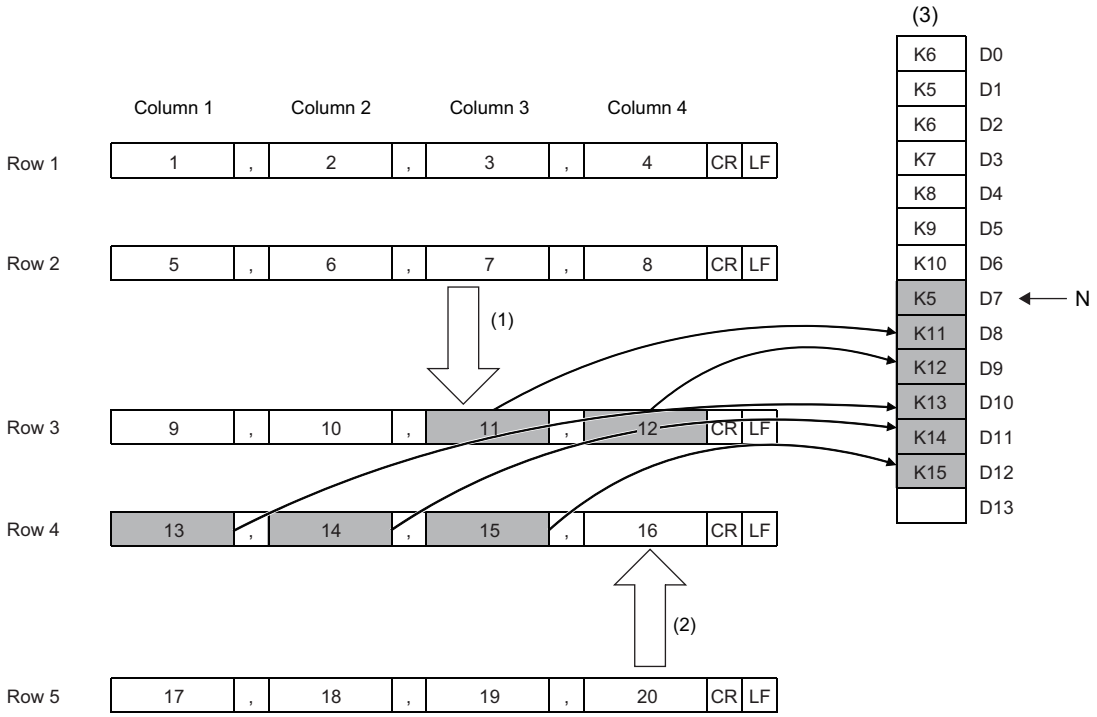
(2) Next starting position

(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 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.

Operation error

Error code (SD0)	Description
2820H	Data is read exceeding the size of the device.
3405H	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 643 Type 1 ☞ Page 645 Type 2
	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. ☞ Page 643 Type 1 ☞ Page 645 Type 2
3427H	An invalid combination of (d1)+0 (Execution/completion type) and (d1)+7 (Data type specification) is specified. ☞ Page 643 Type 1

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 700 Error codes generated for file operation instructions

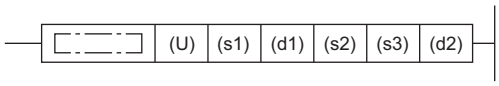
Writing data to the specified file

SP.FWRITE

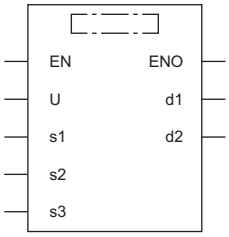
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- The R00CPU does not support this instruction.

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

■Descriptions, ranges, and data types

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 662 Type 1 ☞ Page 665 Type 2	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○	—	—	—	—	—	—	—	—	—

The control data (d1), file name (s2), and write data (s3) vary depending on the CPU module to be used.

CPU module to be used			Reference
RnCPU	R01CPU, R02CPU		Page 662 Type 1
	R04CPU, R08CPU, R16CPU, R32CPU, R120CPU	Firmware version "28" or later	Page 662 Type 1
		Firmware version "27" or earlier	Page 665 Type 2
RnENCPU		Firmware version "28" or later	Page 662 Type 1
		Firmware version "27" or earlier	Page 665 Type 2
RnPCPU			Page 665 Type 2
RnPSFCPU			Page 665 Type 2
RnSFCPU			Page 665 Type 2

■Type 1

• 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 700 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*4	b15 … b0 0 | 1/0 b0: Write start position setting*5 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*2 • 2: Unit of the data type specified by (d1)+0 • 3: Odd number of bytes*1*2*3 "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 To specify "3: Odd number of bytes", use the following models. Note that the applicable firmware version varies depending on models.
· R01CPU and R02CPU: "19" or later
· R04CPU, R08CPU, R16CPU, R32CPU, R120CPU: "51" or later
· RnENCPU: "51" or later

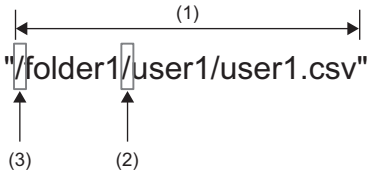
*2 When "1: Even number of bytes" or "3: Odd number of bytes" is specified by (d1)+7 (Data type specification), the setting range of (s3)+0 (Number of request write data) is 1 to 32767.

*3 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).

*4 The setting can be used only for the following models. Note that the applicable firmware version varies depending on models.
· R01CPU and R02CPU: "24" or later
· Rn(EN)CPU (excluding the R01CPU and R02CPU): "57" or later

*5 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 character 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 must be within 60 characters (a period and extension excluded). If 61 or more characters are used, 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)*6*7 • 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)*6*7 • 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

*6 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.

*7 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).

■Type 2

• Control data (d1)

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Execution/completion type	Specify the execution type. • 0000H: Writing binary data • 0100H: Writing data after converted to CSV format	0000H 0100H	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 700 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.	—	System
+3	Not used	—	—	—
+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	0, 1	User

• File name (s2)

Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +□	File name character string	Specify the string of the file name. When omitting an extension, omit the ". (period)" as well. Specify the name within 60 characters (a period and extension excluded). If 61 or more characters are used, the extension is ignored and replaced with ".BIN" or ".CSV".	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 (in units of words). Even when "1: Even number of bytes" is specified by (d1)+7, specify the number in units of words.	1 to 65535*1	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" is specified by (d1)+7 (Data type specification), the setting range is 1 to 32767.

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.

CPU module to be used			Reference
RnCPU	R01CPU, R02CPU		Page 662 Type 1
	R04CPU, R08CPU, R16CPU, R32CPU, R120CPU	Firmware version "28" or later	Page 662 Type 1
		Firmware version "27" or earlier	Page 665 Type 2
RnENCPU		Firmware version "28" or later	Page 662 Type 1
		Firmware version "27" or earlier	Page 665 Type 2
RnPCPU			Page 665 Type 2
RnPSFCPU			Page 665 Type 2
RnSFCPU			Page 665 Type 2

- 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)		

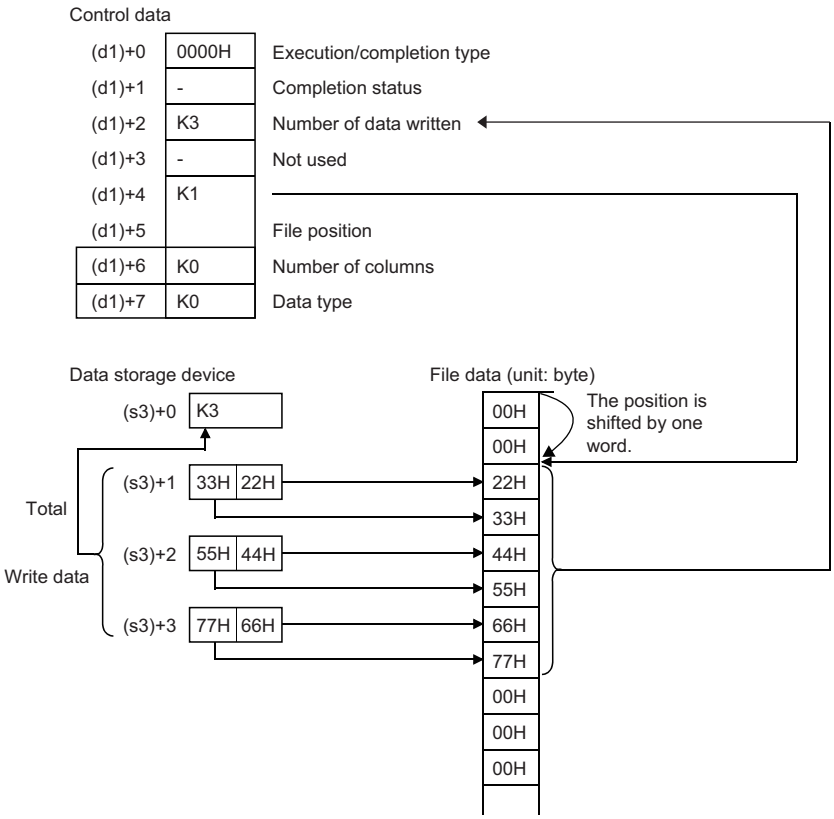
Execution/completion type (d1)+0		Data type specification (d1)+7	Processing unit and setting range		
			Number of request write data (s3)+0	File position (d1)+4, (d1)+5	Number of data actually written (d1)+2
Writing data after converted to CSV format	0100H: Decimal (signed 16-bit data)	0: Word	Word (1 to 65535)	Head/end*2	Word
		1: Even number of bytes	Word (1 to 32767)	Head/end*2	Byte
		2: Unit of the data type specified by the execution/completion type	Word (1 to 65535)	Head/end*2	Word
		3: Odd number of bytes	Word (1 to 32767)	Head/end*2	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*2	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*2	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*2	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*2	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*2	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*2	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*2	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	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*2	4 words
		3: Odd number of bytes	(Cannot be specified)		

*2 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. The newly created file has the archive attribute.
- 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.

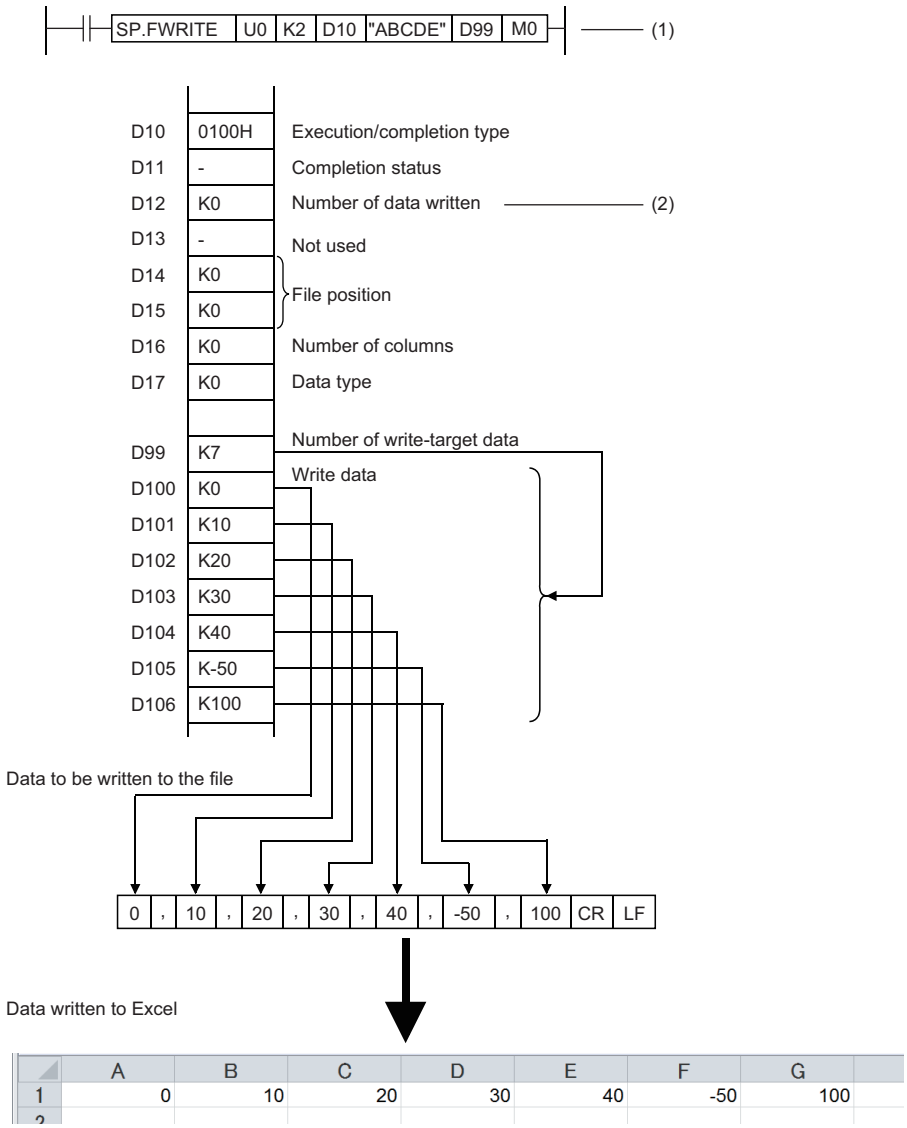


■Converting and writing data in CSV format

- 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. The newly created file has the archive attribute.
- 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.



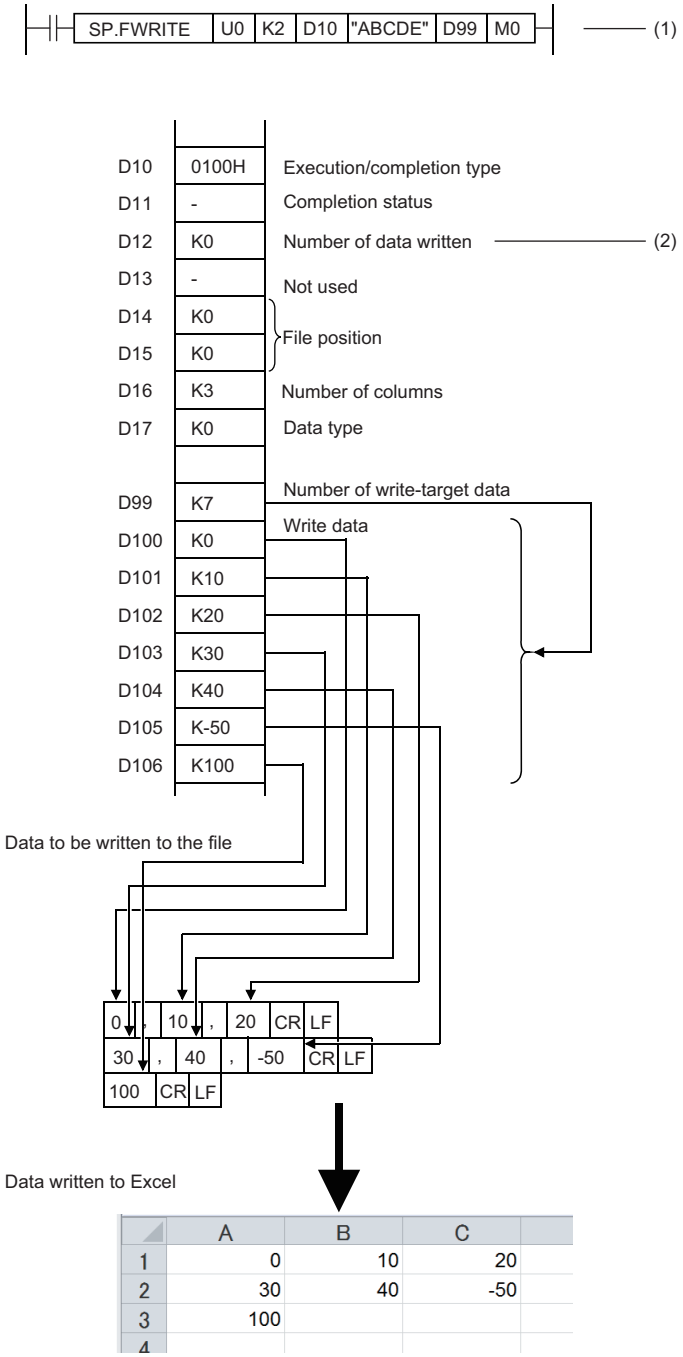
(1) Specified in units of words

(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.



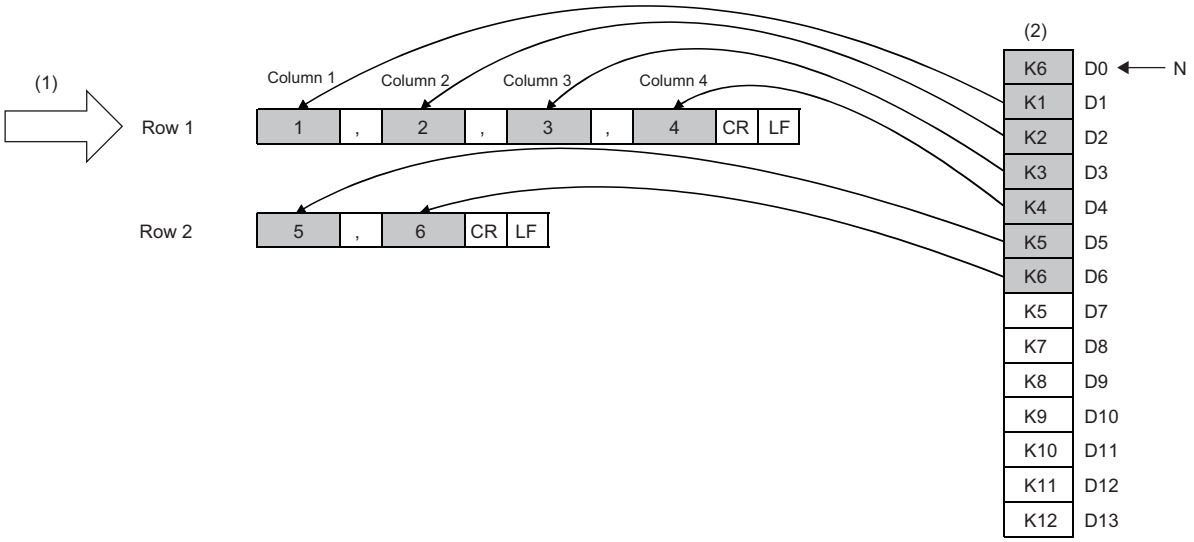
(1) Specified in units of words

(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



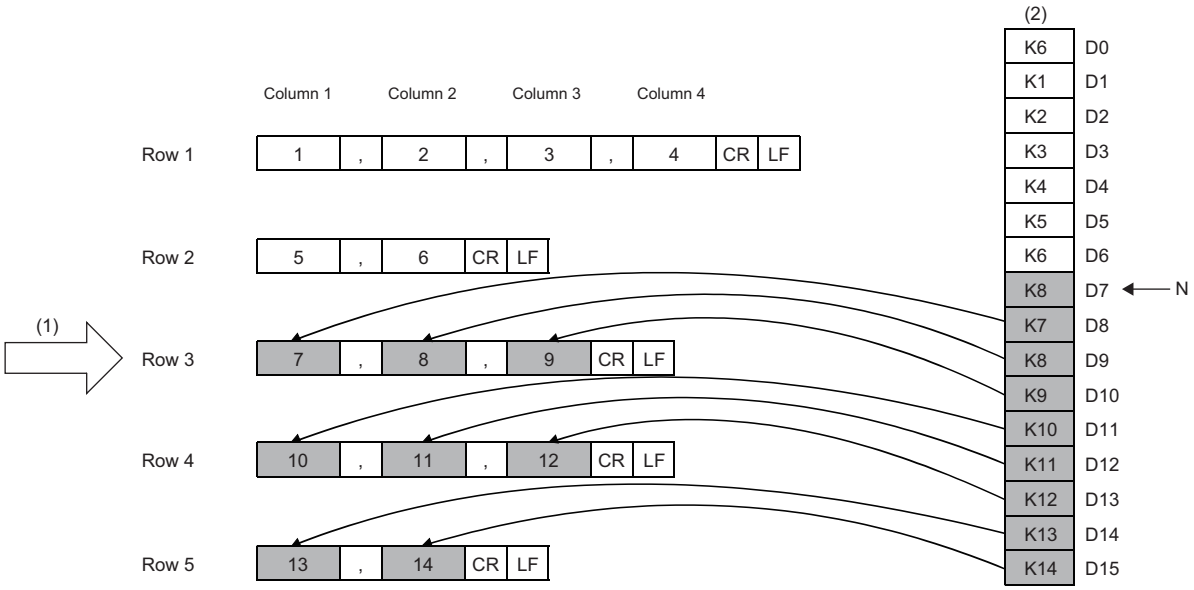
(1) Starting row

(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)

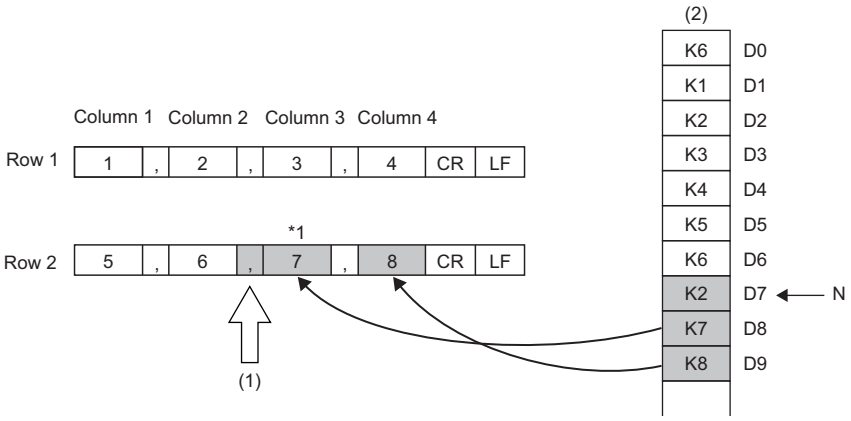
N: Number of data

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

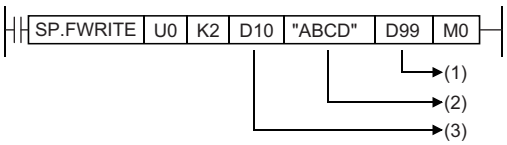
*1 If 1 is set in the application setting area, the return code at the end of the file will be converted to a comma, and data can be added from the last line.

- 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]

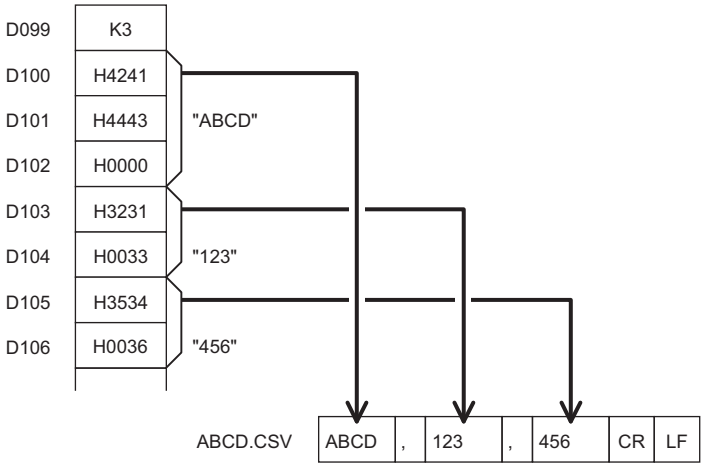


(1) Data to be written
(2) File name
(3) Control data

[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 position
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 ≤ -2^{-126}, 0, 2^{-126} ≤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 ≤ -2^{-1022}, 0, 2^{-1022} ≤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.

Operation error

Error code (SD0)	Description
2820H	The value in the device specified by (s3)+0 is out of the range (1 to 65535), or exceeds the setting area specified by (s3)+1 and later in the device/label memory.
3405H	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 662 Type 1 ☞ Page 665 Type 2
	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. ☞ Page 662 Type 1 ☞ Page 665 Type 2
3427H	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 662 Type 1

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 700 Error codes generated for file operation instructions

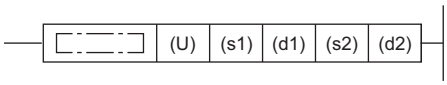
Deleting the specified file

SP.FDELETE

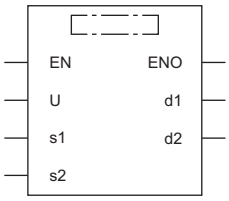
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- The R00CPU does not support this instruction.
- The R01CPU and R02CPU with firmware version "08" or later support this instruction. Use an engineering tool with version "1.050C" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "40" or later support this instruction. Use an engineering tool with version "1.050C" or later.

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

■Descriptions, ranges, and data types

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 679 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

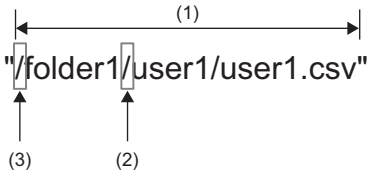
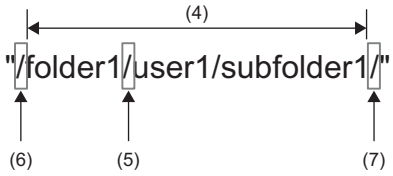
Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○	—	—	—	—	—	—	—	—	—

■Control data (d1)

Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Application setting area	b15 … b1 b0 0 1/0 1/0 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 700 Error codes generated for file operation instructions)	—	System

8

■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. • For a file name, specify the extension as well. • 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.) • 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 CPU module is switched from RUN to STOP during instruction execution, the CPU module 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
3405H	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 700 Error codes generated for file operation instructions

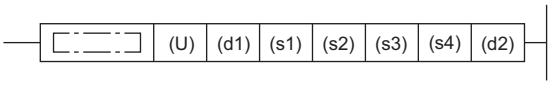
Copying the specified file

SP.FCOPY

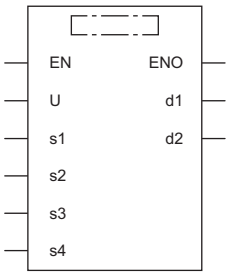
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- The R00CPU does not support this instruction.
- The R01CPU and R02CPU with firmware version "08" or later support this instruction. Use an engineering tool with version "1.050C" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "40" or later support this instruction. Use an engineering tool with version "1.050C" or later.

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

■Descriptions, ranges, and data types

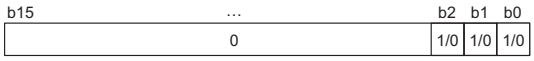
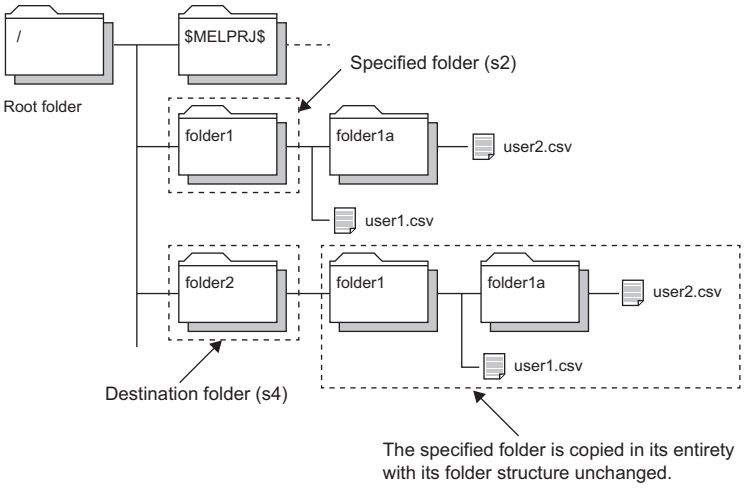
Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	Device name	ANY16
(d1)	Start device where the control data is stored	☞ Page 683 Control data (d1)	Word	ANY16_ARRAY (Number of elements: 2)
(s1)	Copy source drive	2 (fixed)*1	Word	ANY16
(s2)	Start device where the file name or folder name 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.

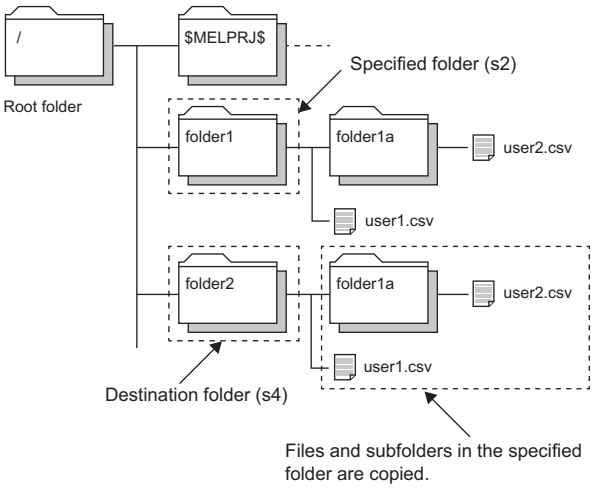
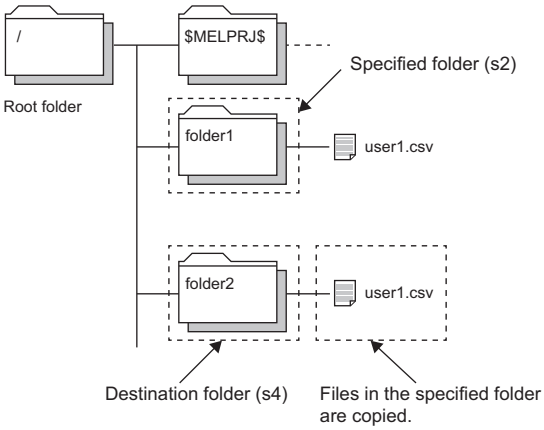
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	○	—	○	—	—	—	—	○	○	—	—	—
(s4)	—	—	○	—	—	—	—	○	—	—	○	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—

■Control data (d1)

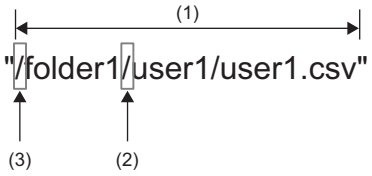
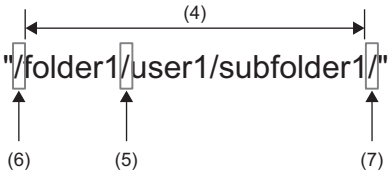
Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Application setting area	 ■b0, b2: Target type setting b0: Specify the type of data (file or folder) to copy. 0: File 1: Folder b2: Specify how the data is copied. (This bit is valid when b0 is set to 1 (Folder).)*1 0: 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

8

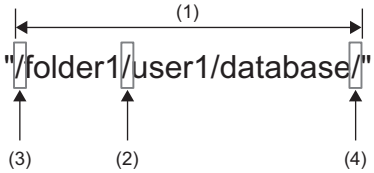
Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Application setting area	• Operation when b2 is set to 1 (Copy all the files and subfolders in the folder), and the specified folder includes files and a subfolder All the files and subfolders in the specified folder are 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 700 Error codes generated for file operation instructions)	—	System

*1 The setting can be used only for the following models. Note that the applicable firmware version varies depending on models.
· R01CPU and R02CPU: "18" or later
· Rn(EN)CPU (excluding the R01CPU and R02CPU): "50" or later

■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. • For a file name, specify the extension as well. • 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.) • 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 CPU module is switched from RUN to STOP during instruction execution, the CPU module 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
3405H	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 700 Error codes generated for file operation instructions

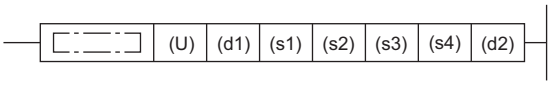
Moving the specified file

SP.FMOVE

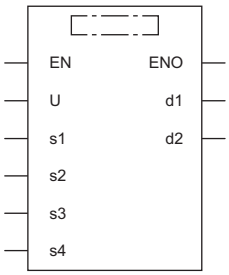
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- The R00CPU does not support this instruction.
- The R01CPU and R02CPU with firmware version "08" or later support this instruction. Use an engineering tool with version "1.050C" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "40" or later support this instruction. Use an engineering tool with version "1.050C" or later.

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

■Descriptions, ranges, and data types

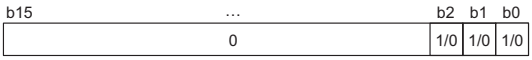
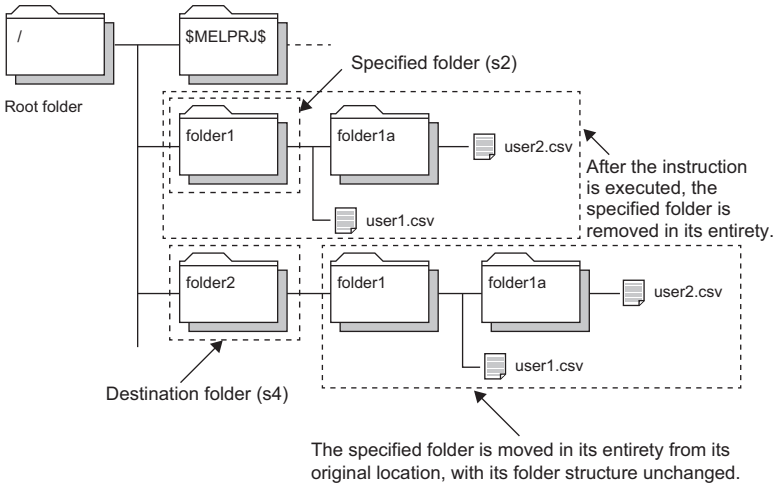
Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	Device name	ANY16
(d1)	Start device where the control data is stored	☞ Page 688 Control data (d1)	Word	ANY16_ARRAY (Number of elements: 2)
(s1)	Source drive	2 (fixed)*1	Word	ANY16
(s2)	Start device where the file name or folder name to 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

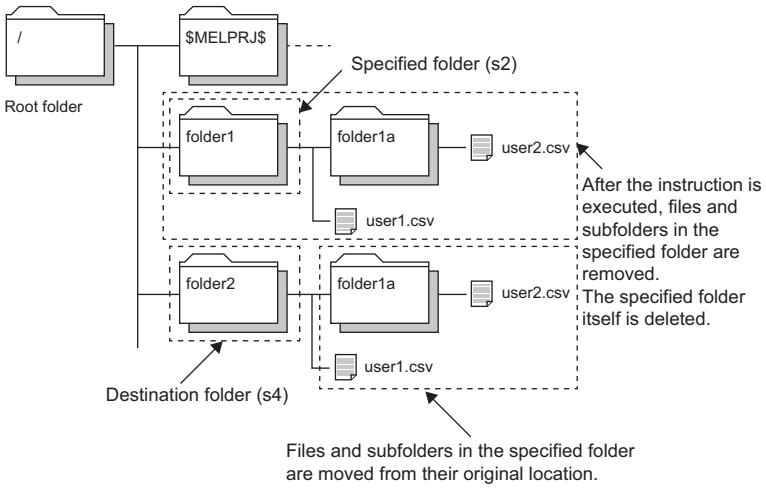
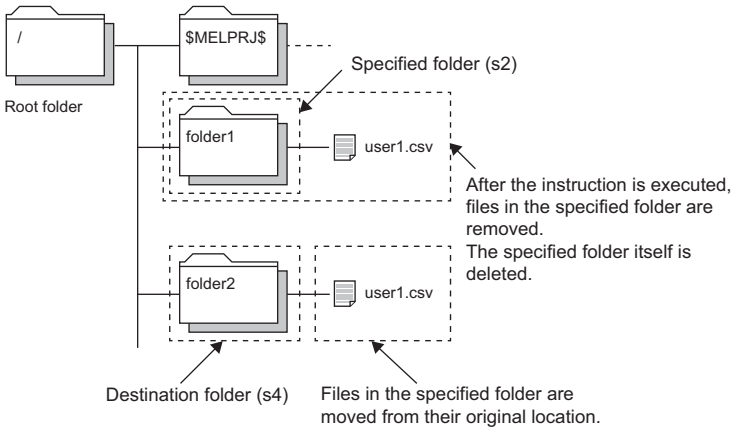
*1 Only drive 2 (for the SD memory card) can be set.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	○	—	○	—	—	—	—	○	○	—	—	—
(s4)	—	—	○	—	—	—	—	○	—	—	○	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—

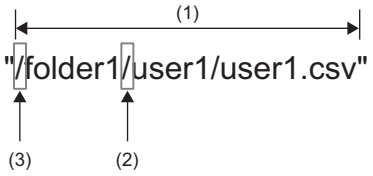
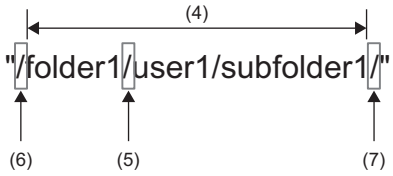
■Control data (d1)

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).)*1 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

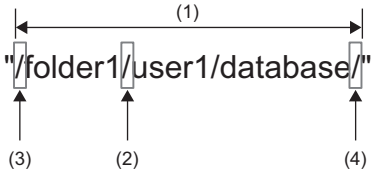
Operand: (d1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Application setting area	• Operation when b2 is set to 1 (Move all the files and subfolders in the folder), and the specified folder includes files and a subfolder All the files and subfolders in the specified folder are 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 700 Error codes generated for file operation instructions)	—	System

*1 The setting can be used only for the following models. Note that the applicable firmware version varies depending on models.
· R01CPU and R02CPU: "18" or later
· Rn(EN)CPU (excluding the R01CPU and R02CPU): "50" or later

■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. • For a file name, specify the extension as well. • 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.) • 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 CPU module is switched from RUN to STOP during instruction execution, the CPU module 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
3405H	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 700 Error codes generated for file operation instructions

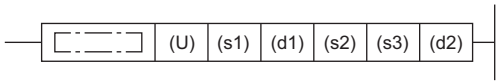
Renaming the specified file

SP.FRENAME

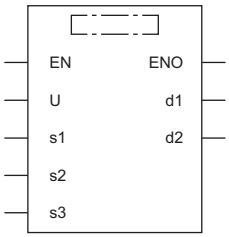
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- The R00CPU does not support this instruction.
- The R01CPU and R02CPU with firmware version "08" or later support this instruction. Use an engineering tool with version "1.050C" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "40" or later support this instruction. Use an engineering tool with version "1.050C" or later.

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

■Descriptions, ranges, and data types

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 693 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

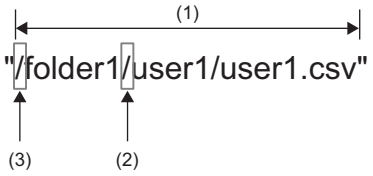
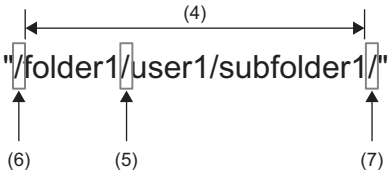
Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(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 rename. • 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 700 Error codes generated for file operation instructions)	—	System

8

■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. • For a file name, specify the extension as well. • 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.) • 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. • For a file name, specify the extension as well. • Do not specify a folder path. • The file name (including an extension) must be within 252 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 CPU module is switched from RUN to STOP during instruction execution, the CPU module 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
3405H	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 700 Error codes generated for file operation instructions

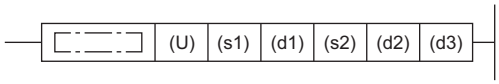
Acquiring the status of the specified file

SP.FSTATUS

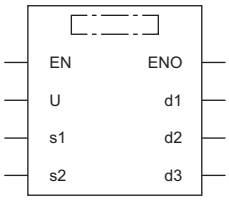
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- The R00CPU does not support this instruction.
- The R01CPU and R02CPU with firmware version "08" or later support this instruction. Use an engineering tool with version "1.050C" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "40" or later support this instruction. Use an engineering tool with version "1.050C" or later.

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

■Descriptions, ranges, and data types

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 697 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

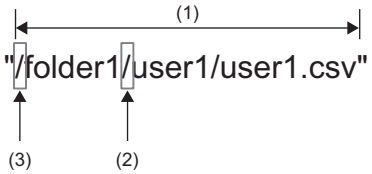
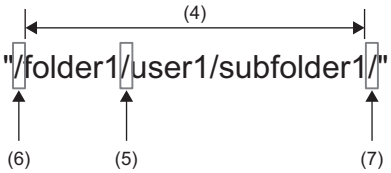
Operand	Bit		Word			Double word		Indirect specification	Constant			Others (U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	—	—	—	—	—	—	—	—	—	○
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(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 700 Error codes generated for file operation instructions)	—	System

8

■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. • For a file name, specify the extension as well. • 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.) • 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 bit0: Turns on for a read-only file. bit1: Turns on for a hidden file. bit2: Turns on for a system file. bit3: Reserved (fixed to 0) bit4: Turns on for a directory. bit5: Turns on for an archive. bit6 to bit15: Reserved (fixed to 0)	Refer to the "Item" column.	System
+1	Reserved	0	
+2 to +3	File size (in units of bytes)	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 In the case of files/folders accessed in an environment (OS) other than the CPU module, 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, (d2) and (d2)+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 CPU module is switched from RUN to STOP during instruction execution, the CPU module continues the processing of the instruction.

Operation error

Error code (SD0)	Description
3405H	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 700 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 CPU module 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.
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.

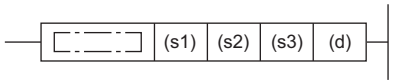
8.5 Data Control Instructions

Upper and lower limit control of 16-bit binary data

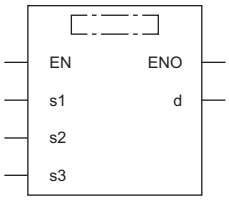
LIMIT(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1485 LIMIT(_E)

■Execution condition

Instruction	Execution condition
LIMIT LIMIT_U	
LIMITP LIMITP_U	

Setting data

■Descriptions, ranges, and data types

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

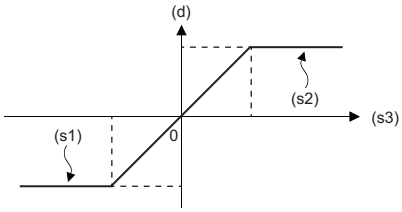
■Applicable devices

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□\□, U3E□\(H)G□	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

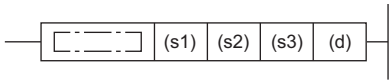
Error code (SD0)	Description
3405H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

Upper and lower limit control of 32-bit binary data

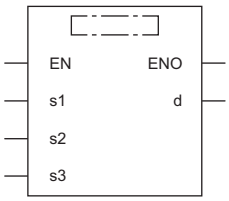
DLIMIT(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1485 LIMIT(_E)

■Execution condition

Instruction	Execution condition
DLIMIT DLIMIT_U	
DLIMITP DLIMITP_U	

Setting data

■Descriptions, ranges, and data types

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

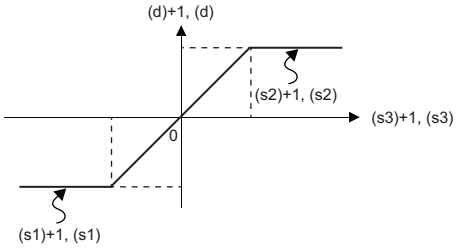
■Applicable devices

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□\□, U3E□\(H)G□	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

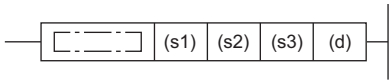
Error code (SD0)	Description
3405H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

Dead band control of 16-bit binary data

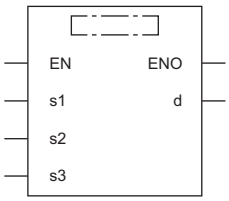
BAND(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

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

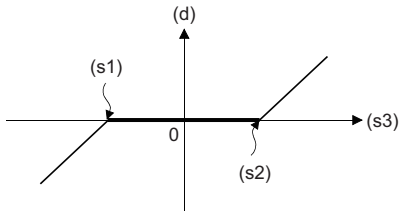
■Applicable devices

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□\□, U3E□\(H)G□	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

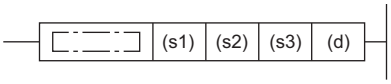
Error code (SD0)	Description
3405H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

Dead band control of 32-bit binary data

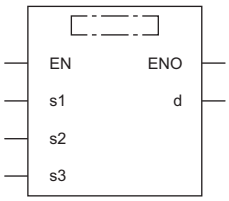
DBAND(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

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

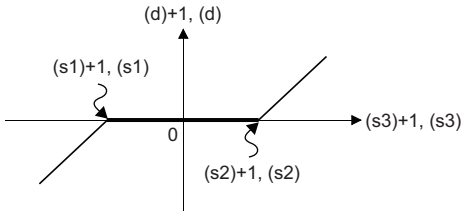
■Applicable devices

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□\□, U3E□\(H)G□	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

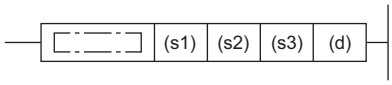
Error code (SD0)	Description
3405H	The lower limit value specified by (s1) is greater than the upper limit value specified by (s2).

Zone control of 16-bit binary data

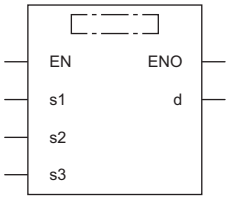
ZONE(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

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

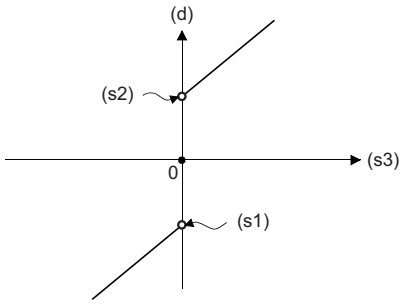
■Applicable devices

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□\□, U3E□\(H)G□	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

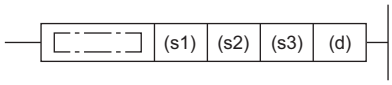
There is no operation error.

Zone control of 32-bit binary data

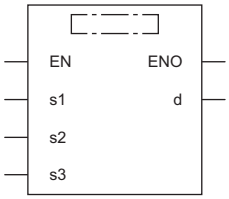
DZONE(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

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

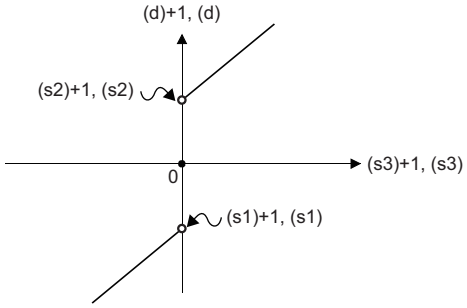
■Applicable devices

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□\□, U3E□\(H)G□	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

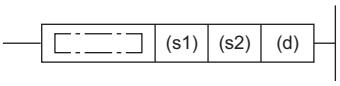
There is no operation error.

Scaling 16-bit binary data (point coordinates)

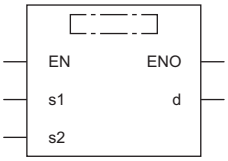
SCL(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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.

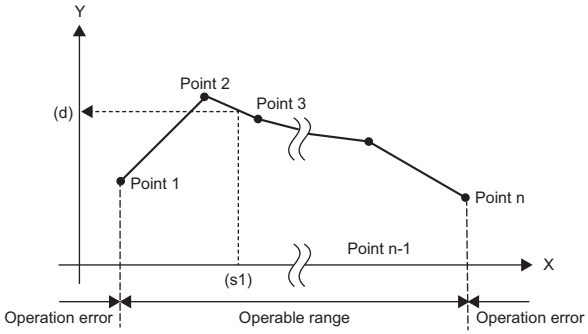
■Applicable devices

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□\□, U3E□\(H)G□	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

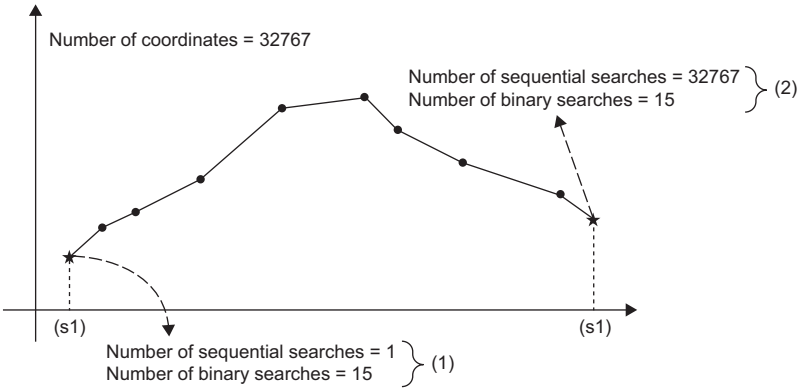
- When the scaling conversion data is sorted in ascending order, the search method varies depending on the status of SM755 and therefore the processing speed also varies. The processing speed depends on the number of searches and is faster as the number of searches is less.

• 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 equal to or greater 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

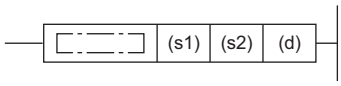
Error code (SD0)	Description
3405H	The X-coordinate data of the scaling conversion data before the point specified by (s1) is not sorted in ascending order. (Note that this error is not detected when SM755 is on.)
	The input value specified by (s1) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s2) is out of the range, 1 to 65535.

Scaling 32-bit binary data (point coordinates)

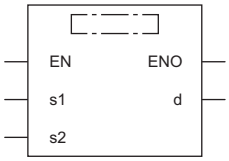
DSCL(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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.

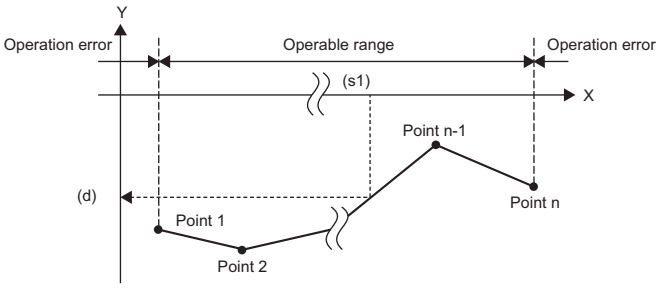
■Applicable devices

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□\□, U3E□\(H)G□	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

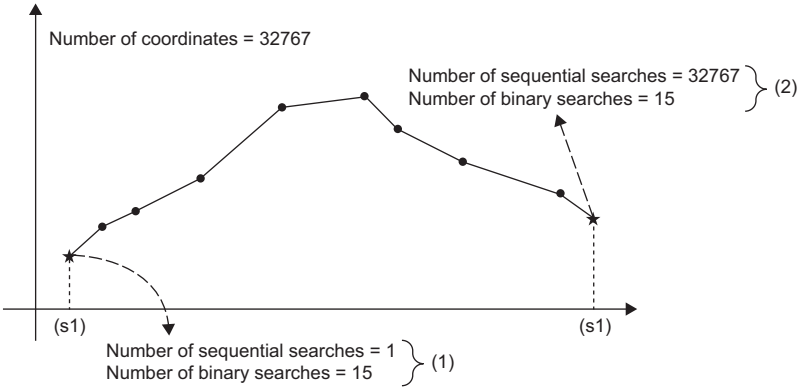
- When the scaling conversion data is sorted in ascending order, the search method varies depending on the status of SM755 and therefore the processing speed also varies. The processing speed depends on the number of searches and is faster as the number of searches is less.

• 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 equal to or greater 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

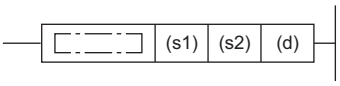
Error code (SD0)	Description
3405H	The X-coordinate data of the scaling conversion data before the point specified by (s1) is not sorted in ascending order. (Note that this error is not detected when SM755 is on.)
	The input value specified by (s1) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s2) is out of the range, 1 to 4294967295.

Scaling 16-bit binary data (XY coordinates)

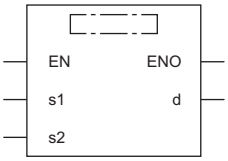
SCL2(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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.

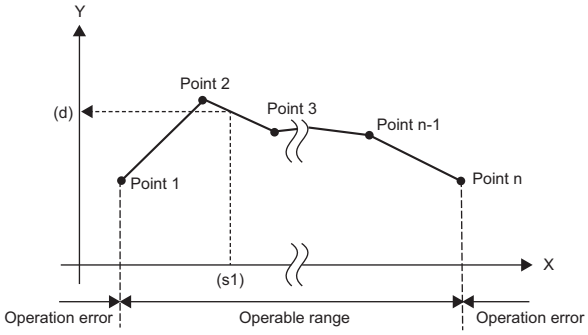
■Applicable devices

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□\□, U3E□\(H)G□	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 713 SCL(P)(_U)

Operation error

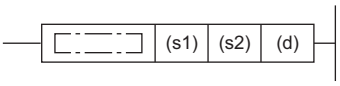
Error code (SD0)	Description
3405H	The X-coordinate data is not sorted in ascending order.
	The input value specified by (s1) is out of the range of the specified scaling conversion data.
	The number of coordinate points starting from the device specified by (s2) is out of the range, 1 to 65535.

Scaling 32-bit binary data (XY coordinates)

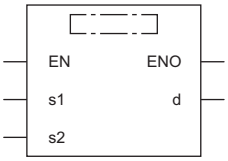
DSCL2(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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.

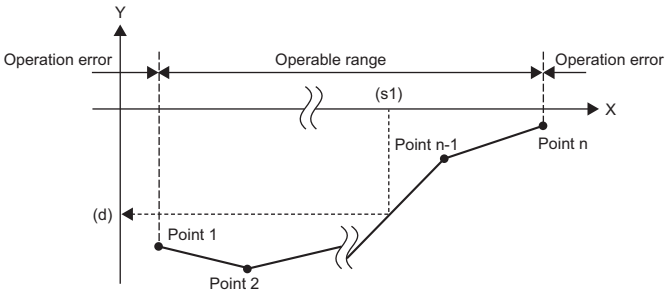
■Applicable devices

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□\□, U3E□\(H)G□	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 716 DSCL(P)(_U)

Operation error

Error code (SD0)	Description
3405H	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.

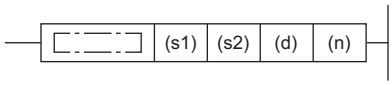
8.6 Data Processing Instructions

Searching 16-bit binary data

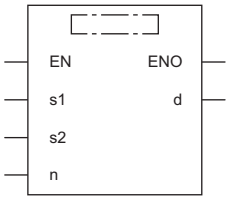
SERDATA(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

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)	Data to be searched or the start device containing the data to be searched	—	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

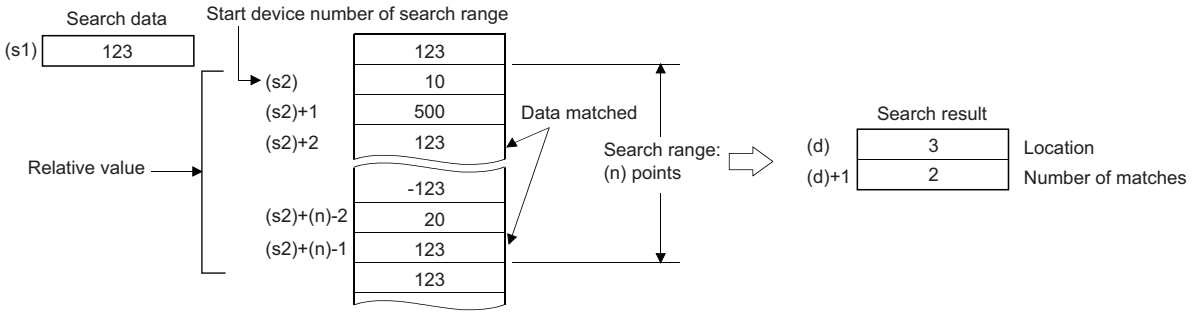
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

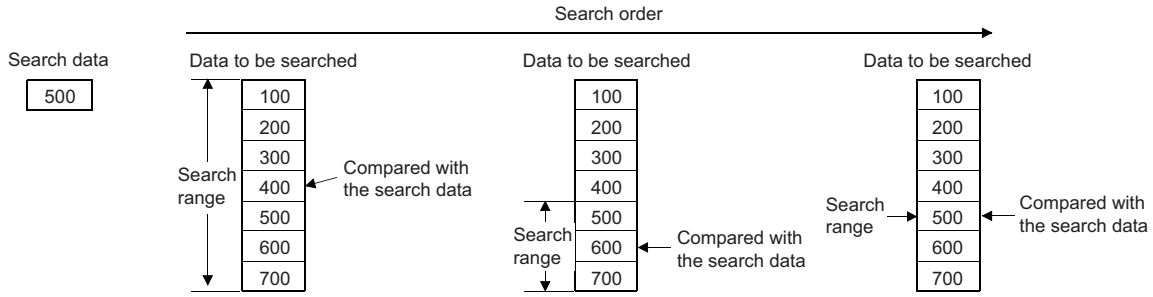
- These instructions search the (n) points in units of 16-bit binary data from the device specified by (s2) using the 16-bit binary data in the device specified by (s1) as a keyword. Each instruction stores the number of data which matches the keyword in the device specified by (d)+1 and also stores 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

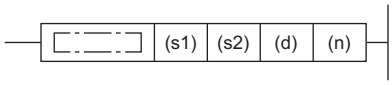
There is no operation error.

Searching 32-bit binary data

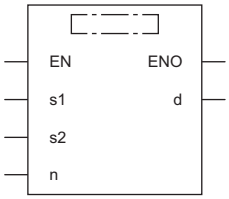
DSERDATA(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

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)	Data to be searched or the start device containing the data to be searched	—	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

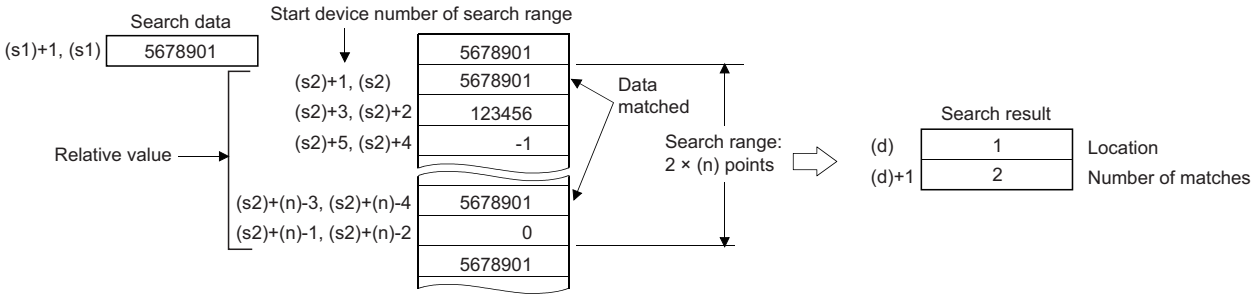
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

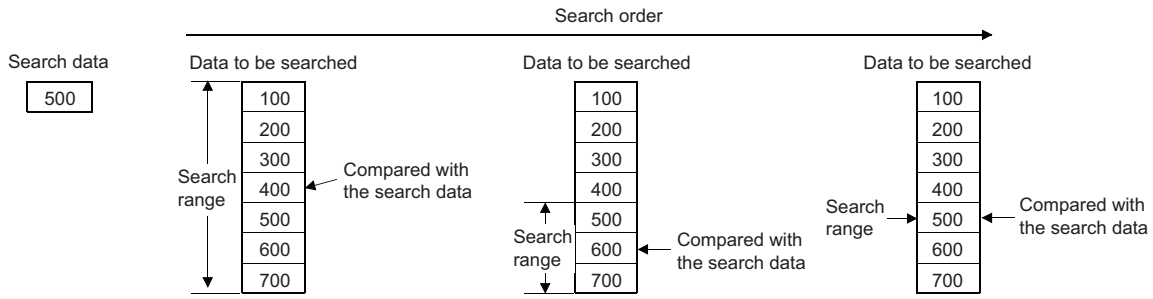
- These instructions search the (n) points of data in units of 32-bit binary data (2 × (n) points of data in units of 16 bits) from the device specified by (s2), using the 16-bit binary data in the device specified by (s1) as a keyword. Each instruction stores the number of data which matches the keyword 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 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.

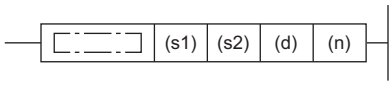
Searching 16-bit binary data (minimum, match, maximum)

SERMM(P)

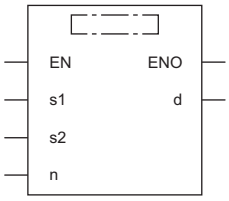
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

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□\□, U3E□\(H)G□	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.

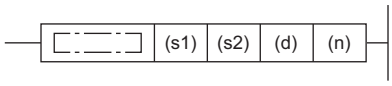
Searching 32-bit binary data (minimum, match, maximum)

DSERMM(P)

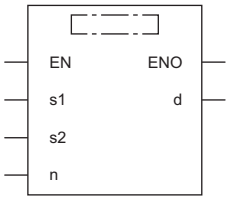
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

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□\□, U3E□\(H)G□	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

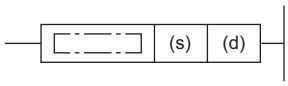
There is no operation error.

Checking 16-bit binary data

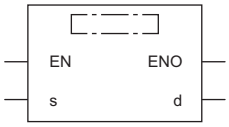
SUM(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

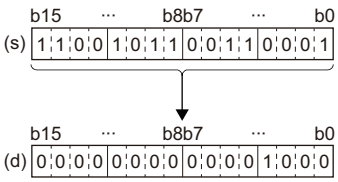
Operand	Description	Range	Data type	Data type (label)
(s)	Device containing data in which the total number of "1" bits is to be counted	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the total number of bits	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution 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□\□, U3E□\(H)G□	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

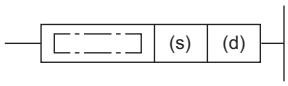
There is no operation error.

Checking 32-bit binary data

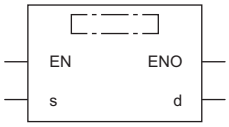
DSUM(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

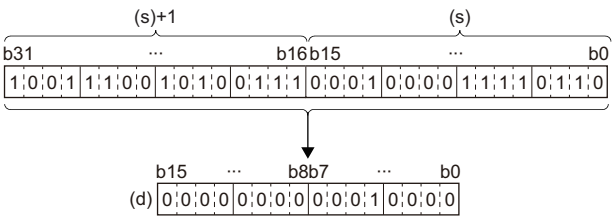
Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing data in which the total number of "1" bits is to be counted	-2147483648 to 2147483647	32-bit signed binary	ANY32
(d)	Device for storing the total number of bits	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	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□\□, U3E□\(H)G□	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.

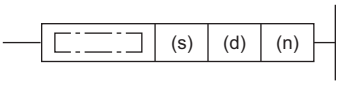
Checking the bit status in 16-bit binary data

BON(P)

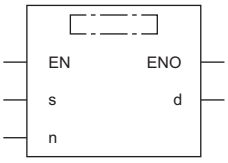
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions check whether (n) bit(s) of the specified device are on or off, and stores the result in the device specified by (d).

Ladder	ST
	ENO:=BON(EN,s,n,d); ENO:=BONP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
BON	
BONP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device storing the data	—	16-bit signed binary	ANY16
(d)	Bit device for storing the result	—	Bit	ANY_BOOL
(n)	Bit position to be checked	0 to 15	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

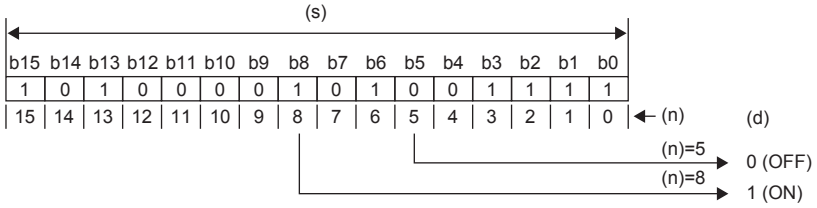
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	○	○	○	○	—	—	○	○	—	—	—
(d)	○	○	○*2	○	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

*1 FX and FY cannot be used.
*2 T, ST, and C cannot be used.

Processing details

- These instructions check whether (n) bit(s) of the device specified by (s) are on or off, and stores the result in the device specified by (d).
- When the result above is on, these instructions turn (d) on. When the result above is off, these instructions turn (d) off.



Operation error

Error code (SD0)	Description
3405H	The value specified for (n) is outside the following range. 0 to 15

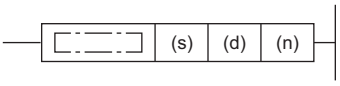
Checking the bit status in 32-bit binary data

DBON(P)

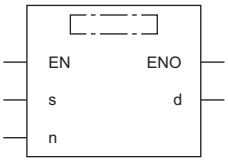
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions check whether (n) bit(s) of the specified device are on or off, and stores the result in the device specified by (d).

Ladder	ST
	ENO:=DBON(EN,s,n,d); ENO:=DBONP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DBON	
DBONP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Device storing the data	—	32-bit signed binary	ANY32
(d)	Bit device for storing the result	—	Bit	ANY_BOOL
(n)	Bit position to be checked	0 to 31	16-bit unsigned binary	ANY16_U
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

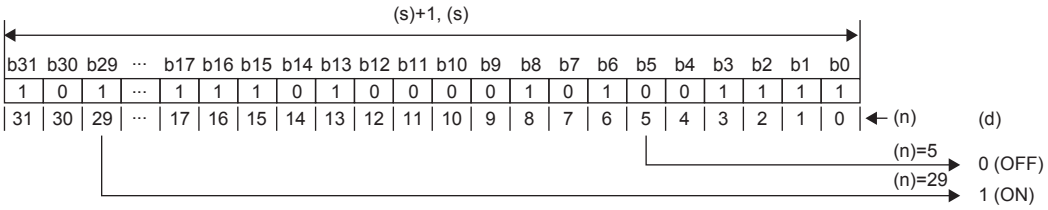
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○*2	○	—	—	—	○	—	—	—	—
(n)	○*1	○	○	○	○	—	—	○	○	—	—	—

*1 FX and FY cannot be used.
*2 T, ST, and C cannot be used.

Processing details

- These instructions check whether (n) bit(s) of the device specified by (s) are on or off, and stores the result in the device specified by (d).
- When the result above is on, these instructions turn (d) on. When the result above is off, these instructions turn (d) off.



Operation error

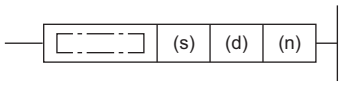
Error code (SD0)	Description
3405H	The value specified for (n) is outside the following range. 0 to 31

Searching the maximum value of 16-bit binary data

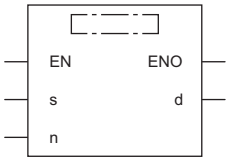
MAX(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1483 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

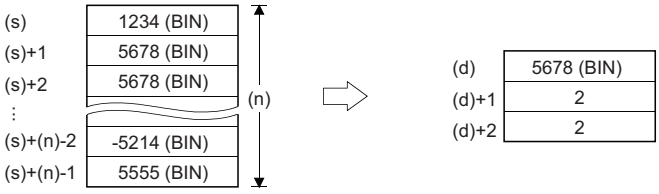
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

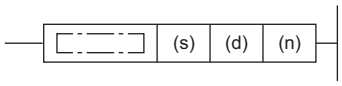
There is no operation error.

Searching the maximum value of 32-bit binary data

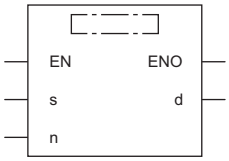
DMAX(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1483 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

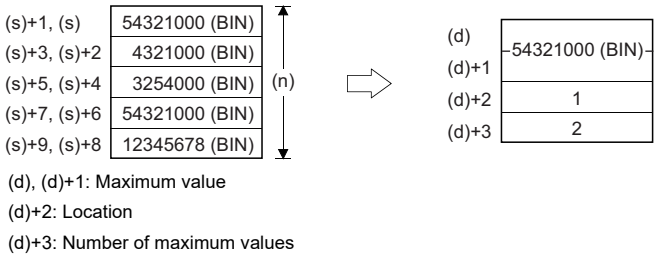
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.
*2 Specify a device regardless of the programming language used. Do not specify labels.

■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□\□, U3E□\(H)G□	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: Location
(d)+3: Number of maximum values

Operation error

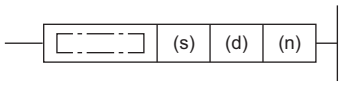
There is no operation error.

Searching the minimum value of 16-bit binary data

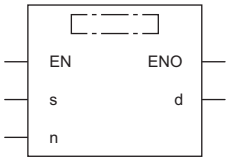
MIN(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1483 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

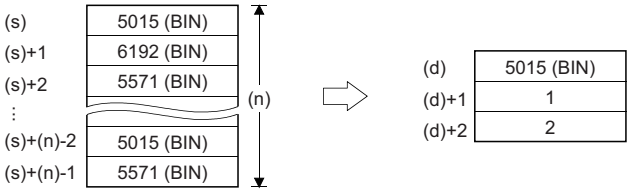
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

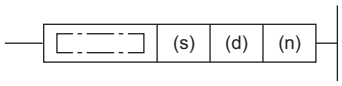
There is no operation error.

Searching the minimum value of 32-bit binary data

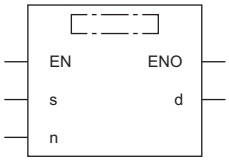
DMIN(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1483 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

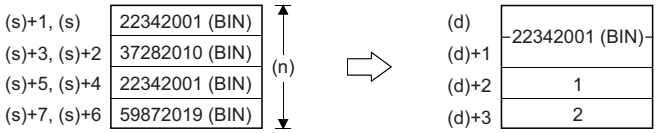
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.
*2 Specify a device regardless of the programming language used. Do not specify labels.

■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□\□, U3E□\(H)G□	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: Location
(d)+3: Number of minimum values

Operation error

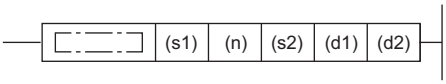
There is no operation error.

Sorting 16-bit binary data

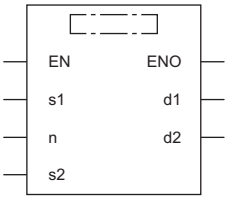
SORTD(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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	0 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.

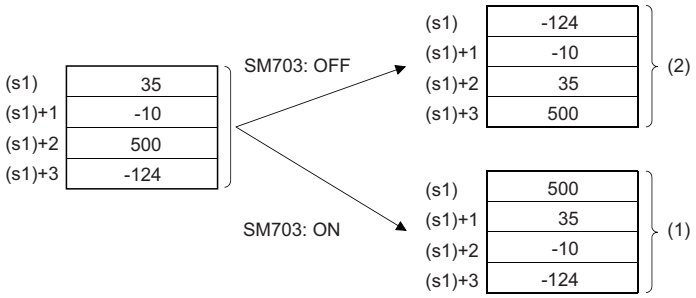
■Applicable devices

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□\□, U3E□\(H)G□	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: 3405H)
- 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

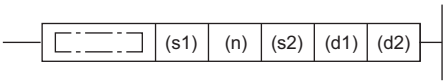
Error code (SD0)	Description
2821H	The device range of (n) points from the device specified by (s1) and the device range of two points from the device specified by (d2) are overlapping.
3405H	The value in (s2) is 0.
	In the second scan or after, the value in (d2) used by the system is equal to or greater 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

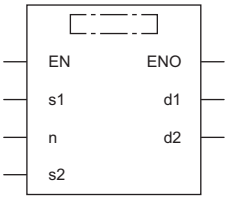
DSORTD(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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	0 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.

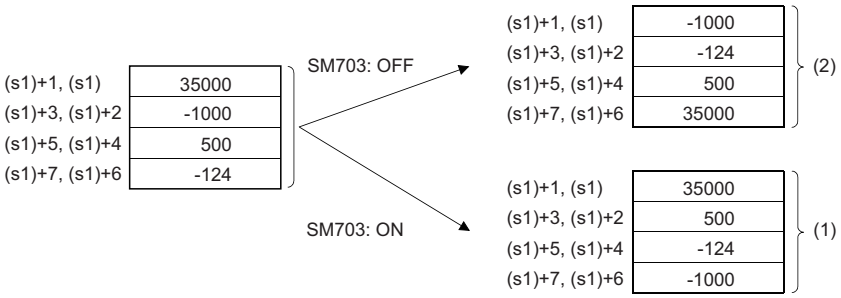
■Applicable devices

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□\□, U3E□\(H)G□	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: 3405H)
- 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.
3405H	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 16-bit binary data table

SORTTBL(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- The R00CPU, R01CPU, and R02CPU with firmware version "08" or later support these instructions. Use an engineering tool with version "1.050C" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "40" or later support these instructions. Use an engineering tool with version "1.050C" or later.

These instructions sort the data rows in the 16-bit binary data table (sorting source) of ((n1) × (n2)) points specified by (s), based on the data in the column (n3) in ascending or descending order. (For the sorting, the data table where the value to the right of (s) in each column increases consecutively is used.) The result is stored in the 16-bit binary data table (sorting result) of ((n1) × (n2)) points specified by (d).

(n3)

	(n2)			
(n1)	(s) 1	(s)+5 2	(s)+10 45	(s)+15 20
	(s)+1 2	(s)+6 5	(s)+11 50	(s)+16 40
	(s)+2 3	(s)+7 4	(s)+12 70	(s)+17 30
	(s)+3 4	(s)+8 1	(s)+13 20	(s)+18 8
	(s)+4 5	(s)+9 3	(s)+14 50	(s)+19 45

⇨

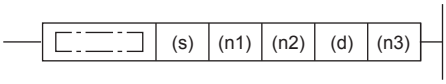
(n3)

	(n2)			
(n1)	(d) 4	(d)+5 1	(d)+10 20	(d)+15 8
	(d)+1 1	(d)+6 2	(d)+11 45	(d)+16 20
	(d)+2 5	(d)+7 3	(d)+12 50	(d)+17 45
	(d)+3 3	(d)+8 4	(d)+13 70	(d)+18 30
	(d)+4 2	(d)+9 5	(d)+14 50	(d)+19 40

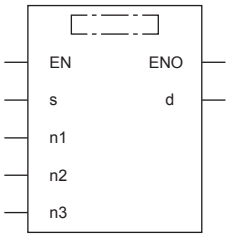
Point

When configuring the table with devices consecutive in the row direction, use the following instruction.

☞ Page 755 SORTTBL2(_U)

Ladder	ST	
	ENO:=SORTTBL(EN,s,n1,n2,n3,d);	ENO:=SORTTBL_U(EN,s,n1,n2,n3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SORTTBL SORTTBL_U	↑

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s)	SORTTBL	Start device where a data table is stored	—	16-bit signed binary	ANY16_S*1
	SORTTBL_U			16-bit unsigned binary	ANY16_U*1
(n1)		Number of data (rows)	1 to 32	16-bit unsigned binary	ANY16_U
(n2)		Number of data (columns)	1 to 6	16-bit unsigned binary	ANY16_U
(d)	SORTTBL	Start device for storing the operation result	—	16-bit signed binary	ANY16_S*1
	SORTTBL_U			16-bit unsigned binary	ANY16_U*1
(n3)		Column number of data (column) used as the basis of sorting	—	16-bit unsigned binary	ANY16_U
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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○	○	○	○	○	—	—	○	○	—	—	—
(n2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n3)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions sort the data rows in the data table (sorting source) of ((n1) × (n2)) points specified by (s), based on the data in the column (n3) in ascending or descending order. (For the sorting, the data table where the value to the right of (s) in each column increases consecutively is used.) The result is stored in the data table (sorting result) of ((n1) × (n2)) points specified by (d).
- The following example shows a data table configuration when the sorting source data table has 3 rows and 4 columns ((n1) = 3, (n2) = 4). For the sorting result data table, understand (s) as (d).

Ex.

Data table configuration (rows (n1) = 3, columns (n2) = 4)

—	Column No.1	Column No.2	Column No.3	Column No.4
Row No.1	(s)	(s)+3	(s)+6	(s)+9
Row No.2	(s)+1	(s)+4	(s)+7	(s)+10
Row No.3	(s)+2	(s)+5	(s)+8	(s)+11

- The sorting order is determined according to the value (on or off) of SM774 (Data table sort instruction sort order) set at the start of execution of the instruction.

SM774 set value	Sorting order
OFF	Ascending order
ON	Descending order

- When the execution command is turned on, bit 2 of SD774 (Execution status of data table sort instructions) turns on and the sorting is started. After (n1) scans, data sorting completes at the execution of the END instruction and bit 2 of SD774 turns off at the same time that bit 0 (completion flag) of SD774 turns on. The completion flag turns off when the END instruction of the next scan is executed.
- The SORTTBL(_U) instruction cannot be executed while bit 2 of SD774 is on. (If the instruction is executed, no processing is performed.)

■Operation example

The following figures show operation examples when the following data table (rows (n1) = 5, columns (n2) = 4) is sorted under each condition.

		(n2)			
		Column No.1	Column No.2	Column No.3	Column No.4
		Control number	Dimension X	Dimension Y	Dimension Z
(n1)	Line No.1	(s) 1	(s)+5 150	(s)+10 45	(s)+15 20
	Line No.2	(s)+1 2	(s)+6 180	(s)+11 50	(s)+16 40
	Line No.3	(s)+2 3	(s)+7 160	(s)+12 70	(s)+17 30
	Line No.4	(s)+3 4	(s)+8 100	(s)+13 20	(s)+18 8
	Line No.5	(s)+4 5	(s)+9 150	(s)+14 50	(s)+19 45

Point

Adding consecutive numbers such as control numbers in the first column helps to know the original row numbers.

Ex.

When (n3) is set to 2 and SM774 is off (ascending order)

		(n2)			
		Column No.1	Column No.2	Column No.3	Column No.4
		Control number	Dimension X	Dimension Y	Dimension Z
(n1)	Line No.1	(s) 1	(s)+5 150	(s)+10 45	(s)+15 20
	Line No.2	(s)+1 2	(s)+6 180	(s)+11 50	(s)+16 40
	Line No.3	(s)+2 3	(s)+7 160	(s)+12 70	(s)+17 30
	Line No.4	(s)+3 4	(s)+8 100	(s)+13 20	(s)+18 8
	Line No.5	(s)+4 5	(s)+9 150	(s)+14 50	(s)+19 45

⇨

		(n2)			
		Column No.1	Column No.2	Column No.3	Column No.4
		Control number	Dimension X	Dimension Y	Dimension Z
(n1)	Line No.1	(d) 4	(d)+5 100	(d)+10 20	(d)+15 8
	Line No.2	(d)+1 1	(d)+6 150	(d)+11 45	(d)+16 20
	Line No.3	(d)+2 5	(d)+7 150	(d)+12 50	(d)+17 45
	Line No.4	(d)+3 3	(d)+8 160	(d)+13 70	(d)+18 30
	Line No.5	(d)+4 2	(d)+9 180	(d)+14 50	(d)+19 40

Ex.

When (n3) is set to 3 and SM774 is on (descending order)

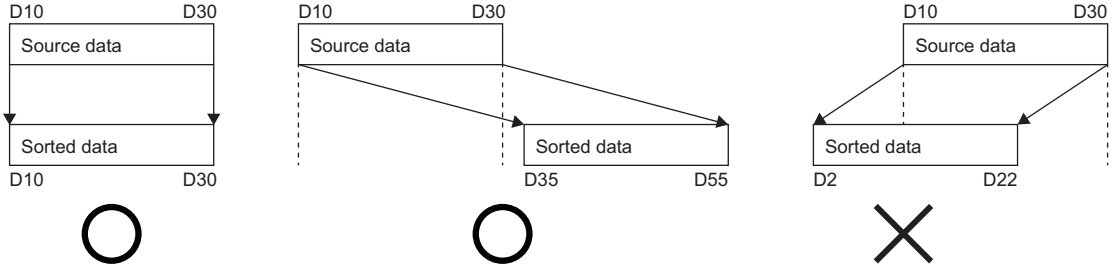
		(n2)			
		Column No.1	Column No.2	Column No.3	Column No.4
		Control number	Dimension X	Dimension Y	Dimension Z
(n1)	Line No.1	(s) 1	(s)+5 150	(s)+10 45	(s)+15 20
	Line No.2	(s)+1 2	(s)+6 180	(s)+11 50	(s)+16 40
	Line No.3	(s)+2 3	(s)+7 160	(s)+12 70	(s)+17 30
	Line No.4	(s)+3 4	(s)+8 100	(s)+13 20	(s)+18 8
	Line No.5	(s)+4 5	(s)+9 150	(s)+14 50	(s)+19 45

⇨

		(n2)			
		Column No.1	Column No.2	Column No.3	Column No.4
		Control number	Dimension X	Dimension Y	Dimension Z
(n1)	Line No.1	(d) 3	(d)+5 160	(d)+10 70	(d)+15 30
	Line No.2	(d)+1 2	(d)+6 180	(d)+11 50	(d)+16 40
	Line No.3	(d)+2 5	(d)+7 150	(d)+12 50	(d)+17 45
	Line No.4	(d)+3 1	(d)+8 150	(d)+13 45	(d)+18 20
	Line No.5	(d)+4 4	(d)+9 100	(d)+14 20	(d)+19 8

Precautions

- Do not change the contents of operands and data during operation. The correct result may not be obtained when sorting is completed.
- When the same device is specified by (s) and (d), the source data is overwritten by the sorting result. Do not change the contents of (s) until execution is completed (bit 0 of SD774 turns on).
- When specifying (s) and (d), specify the same device range or specify devices whose ranges do not overlap.



Operation error

Error code (SD0)	Description
2820H	The device range specified by (s) exceeds the corresponding device range.
	The device range specified by (d) exceeds the corresponding device range.
2821H	The device ranges specified by (s) and (d) overlap partly.
3405H	The value specified by (n1) is outside the following range. 1 to 32
	The value specified by (n2) is outside the following range. 1 to 6
	The value specified by (n3) is outside the following range. 1 to (n2)

Sorting 16-bit binary data table 2

SORTTBL2(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- The R00CPU, R01CPU, and R02CPU with firmware version "08" or later support these instructions. Use an engineering tool with version "1.050C" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "40" or later support these instructions. Use an engineering tool with version "1.050C" or later.

These instructions sort the data rows in the 16-bit binary data table (sorting source) of ((n1) × (n2)) points specified by (s), based on the data in the column (n3) in ascending or descending order. (For the sorting, the data table where the value to the right of (s) in each row increases consecutively is used.) The result is stored in the 16-bit binary data table (sorting result) of ((n1) × (n2)) points specified by (d).

(n3)

	(n2)			
(n1)	(s) 1	(s)+1 2	(s)+2 45	(s)+3 20
	(s)+4 2	(s)+5 5	(s)+6 50	(s)+7 40
	(s)+8 3	(s)+9 4	(s)+10 70	(s)+11 30
	(s)+12 4	(s)+13 1	(s)+14 20	(s)+15 8
	(s)+16 5	(s)+17 3	(s)+18 50	(s)+19 45

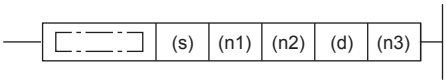
⇨

(n3)

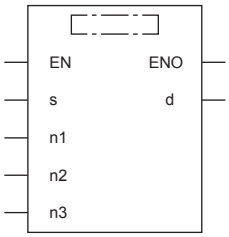
	(n2)			
(n1)	(d) 4	(d)+1 1	(d)+2 20	(d)+3 8
	(d)+4 1	(d)+5 2	(d)+6 45	(d)+7 20
	(d)+8 5	(d)+9 3	(d)+10 50	(d)+11 45
	(d)+12 3	(d)+13 4	(d)+14 70	(d)+15 30
	(d)+16 2	(d)+17 5	(d)+18 50	(d)+19 40

Point

When configuring the table with devices consecutive in the column direction, use the following instruction.
☞ Page 751 SORTTBL(_U)

Ladder	ST	
	ENO:=SORTTBL2(EN,s,n1,n2,n3,d);	ENO:=SORTTBL2_U(EN,s,n1,n2,n3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SORTTBL2 SORTTBL2_U	↑

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s)	SORTTBL2	Start device where a data table is stored	—	16-bit signed binary	ANY16_S*1
	SORTTBL2_U			16-bit unsigned binary	ANY16_U*1
(n1)		Number of data (rows)	1 to 32	16-bit unsigned binary	ANY16_U
(n2)		Number of data (columns)	1 to 6	16-bit unsigned binary	ANY16_U
(d)	SORTTBL2	Start device for storing the operation result	—	16-bit signed binary	ANY16_S*1
	SORTTBL2_U			16-bit unsigned binary	ANY16_U*1
(n3)		Column number of data (column) used as the basis of sorting	—	16-bit unsigned binary	ANY16_U
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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○	○	○	○	○	—	—	○	○	—	—	—
(n2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n3)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions sort the data rows in the data table (sorting source) of ((n1) × (n2)) points specified by (s), based on the data in the column (n3) in ascending or descending order. (For the sorting, the data table where the value to the right of (s) in each row increases consecutively is used.) The result is stored in the data table (sorting result) of ((n1) × (n2)) points specified by (d).
- The following example shows a data table configuration when the sorting source data table has 3 rows and 4 columns ((n1) = 3, (n2) = 4). For the sorting result data table, understand (s) as (d).

Ex.

Data table configuration (rows (n1) = 3, columns (n2) = 4)

—	Column No.1	Column No.2	Column No.3	Column No.4
Row No.1	(s)	(s)+1	(s)+2	(s)+3
Row No.2	(s)+4	(s)+5	(s)+6	(s)+7
Row No.3	(s)+8	(s)+9	(s)+10	(s)+11

- The sorting order is determined according to the value (on or off) of SM774 (Data table sort instruction sort order) set at the start of execution of the instruction.

SM774 set value	Sorting order
OFF	Ascending order
ON	Descending order

- When the execution command is turned on, bit 8 of SD774 (Execution status of data table sort instructions) turns on and the sorting is started. After (n1) scans, data sorting completes at the execution of the END instruction and bit 8 of SD774 turns off at the same time that bit 6 (completion flag) of SD774 turns on. The completion flag turns off when the END instruction of the next scan is executed.
- The SORTTBL2(_U) instruction cannot be executed while bit 8 of SD774 is on. (If the instruction is executed, no processing is performed.)

■Operation example

The following figures show operation examples when the following data table (rows (n1) = 5, columns (n2) = 4) is sorted under each condition.

		(n2)			
		Column No.1	Column No.2	Column No.3	Column No.4
		Control number	Dimension X	Dimension Y	Dimension Z
(n1)	Line No.1	(s) 1	(s)+1 150	(s)+2 45	(s)+3 20
	Line No.2	(s)+4 2	(s)+5 180	(s)+6 50	(s)+7 40
	Line No.3	(s)+8 3	(s)+9 160	(s)+10 70	(s)+11 30
	Line No.4	(s)+12 4	(s)+13 100	(s)+14 20	(s)+15 8
	Line No.5	(s)+16 5	(s)+17 150	(s)+18 50	(s)+19 45

Point

Adding consecutive numbers such as control numbers in the first column helps to know the original row numbers.

Ex.

When (n3) is set to 2 and SM774 is off (ascending order)

		(n2)			
		Column No.1	Column No.2	Column No.3	Column No.4
		Control number	Dimension X	Dimension Y	Dimension Z
(n1)	Line No.1	(s) 1	(s)+1 150	(s)+2 45	(s)+3 20
	Line No.2	(s)+4 2	(s)+5 180	(s)+6 50	(s)+7 40
	Line No.3	(s)+8 3	(s)+9 160	(s)+10 70	(s)+11 30
	Line No.4	(s)+12 4	(s)+13 100	(s)+14 20	(s)+15 8
	Line No.5	(s)+16 5	(s)+17 150	(s)+18 50	(s)+19 45

⇨

		(n2)			
		Column No.1	Column No.2	Column No.3	Column No.4
		Control number	Dimension X	Dimension Y	Dimension Z
(n1)	Line No.1	(d) 4	(d)+1 100	(d)+2 20	(d)+3 8
	Line No.2	(d)+4 1	(d)+5 150	(d)+6 45	(d)+7 20
	Line No.3	(d)+8 5	(d)+9 150	(d)+10 50	(d)+11 45
	Line No.4	(d)+12 3	(d)+13 160	(d)+14 70	(d)+15 30
	Line No.5	(d)+16 2	(d)+17 180	(d)+18 50	(d)+19 40

Ex.

When (n3) is set to 3 and SM774 is on (descending order)

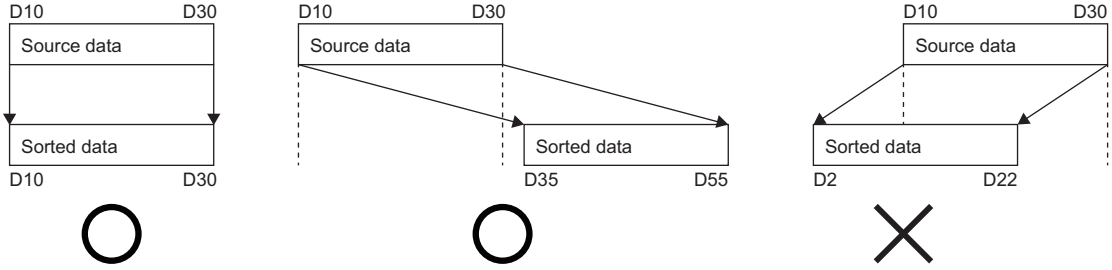
		(n2)			
		Column No.1	Column No.2	Column No.3	Column No.4
		Control number	Dimension X	Dimension Y	Dimension Z
(n1)	Line No.1	(s) 1	(s)+1 150	(s)+2 45	(s)+3 20
	Line No.2	(s)+4 2	(s)+5 180	(s)+6 50	(s)+7 40
	Line No.3	(s)+8 3	(s)+9 160	(s)+10 70	(s)+11 30
	Line No.4	(s)+12 4	(s)+13 100	(s)+14 20	(s)+15 8
	Line No.5	(s)+16 5	(s)+17 150	(s)+18 50	(s)+19 45

⇨

		(n2)			
		Column No.1	Column No.2	Column No.3	Column No.4
		Control number	Dimension X	Dimension Y	Dimension Z
(n1)	Line No.1	(d) 3	(d)+1 160	(d)+2 70	(d)+3 30
	Line No.2	(d)+4 2	(d)+5 180	(d)+6 50	(d)+7 40
	Line No.3	(d)+8 5	(d)+9 150	(d)+10 50	(d)+11 45
	Line No.4	(d)+12 1	(d)+13 150	(d)+14 45	(d)+15 20
	Line No.5	(d)+16 4	(d)+17 100	(d)+18 20	(d)+19 8

Precautions

- Do not change the contents of operands and data during operation. The correct result may not be obtained when sorting is completed.
- When the same device is specified by (s) and (d), the source data is overwritten by the sorting result. Do not change the contents of (s) until execution is completed (bit 6 of SD774 turns on).
- When specifying (s) and (d), specify the same device range or specify devices whose ranges do not overlap.



Operation error

Error code (SD0)	Description
2820H	The device range specified by (s) exceeds the corresponding device range.
	The device range specified by (d) exceeds the corresponding device range.
2821H	The device ranges specified by (s) and (d) overlap partly.
3405H	The value specified by (n1) is outside the following range. 1 to 32
	The value specified by (n2) is outside the following range. 1 to 6
	The value specified by (n3) is outside the following range. 1 to (n2)

Sorting 32-bit binary data table 2

DSORTTBL2(_U)

RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "08" or later support these instructions. Use an engineering tool with version "1.050C" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "40" or later support these instructions. Use an engineering tool with version "1.050C" or later.

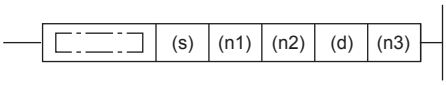
These instructions sort the data rows in the 32-bit binary data table (sorting source) of ((n1) × (n2)) points specified by (s), based on the data in the column (n3) in ascending or descending order. (For the sorting, the data table where the values (odd number and even number) to the right of (s) in each row increase consecutively is used.) The result is stored in the 32-bit binary data table (sorting result) of ((n1) × (n2)) points specified by (d).

(n3)

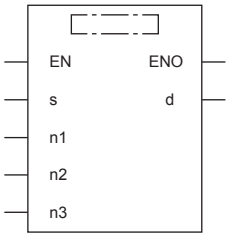
	(n2)			
(n1)	(s)+1, (s) 1	(s)+3, (s)+2 2	(s)+5, (s)+4 45000	(s)+7, (s)+6 20
	(s)+9, (s)+8 2	(s)+11, (s)+10 5	(s)+13, (s)+12 50000	(s)+15, (s)+14 40
	(s)+17, (s)+16 3	(s)+19, (s)+18 4	(s)+21, (s)+20 70000	(s)+23, (s)+22 30
	(s)+25, (s)+24 4	(s)+27, (s)+26 1	(s)+29, (s)+28 20000	(s)+31, (s)+30 8
	(s)+33, (s)+32 5	(s)+35, (s)+34 3	(s)+37, (s)+36 50000	(s)+39, (s)+38 45

(n3)

	(n2)			
(n1)	(d)+1, (d) 4	(d)+3, (d)+2 1	(d)+5, (d)+4 20000	(d)+7, (d)+6 8
	(d)+9, (d)+8 1	(d)+11, (d)+10 2	(d)+13, (d)+12 45000	(d)+15, (d)+14 20
	(d)+17, (d)+16 5	(d)+19, (d)+18 3	(d)+21, (d)+20 50000	(d)+23, (d)+22 45
	(d)+25, (d)+24 3	(d)+27, (d)+26 4	(d)+29, (d)+28 70000	(d)+31, (d)+30 30
	(d)+33, (d)+32 2	(d)+35, (d)+34 5	(d)+37, (d)+36 50000	(d)+39, (d)+38 40

Ladder	ST	
	ENO:=DSORTTBL2(EN,s,n1,n2,n3,d);	ENO:=DSORTTBL2_U(EN,s,n1,n2,n3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
DSORTTBL2 DSORTTBL2_U	↑

Setting data

■Descriptions, ranges, and data types

Operand		Description	Range	Data type	Data type (label)
(s)	DSORTTBL2	Start device where a data table is stored	—	32-bit signed binary	ANY32_S*1
	DSORTTBL2_U			32-bit unsigned binary	ANY32_U*1
(n1)		Number of data (rows)	1 to 32	16-bit unsigned binary	ANY16_U
(n2)		Number of data (columns)	1 to 6	16-bit unsigned binary	ANY16_U
(d)	DSORTTBL2	Start device for storing the operation result	—	32-bit signed binary	ANY32_S*1
	DSORTTBL2_U			32-bit unsigned binary	ANY32_U*1
(n3)		Column number of data (column) used as the basis of sorting	—	16-bit unsigned binary	ANY16_U
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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○*1	—	○	—	—	—	—
(n1)	○	○	○	○	○	—	—	○	○	—	—	—
(n2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	○*1	—	○	—	—	—	—
(n3)	○	○	○	○	○	—	—	○	○	—	—	—

*1 LT and LST cannot be used.

Processing details

- These instructions sort the data rows in the 32-bit binary data table (sorting source) of ((n1) × (n2)) points specified by (s), based on the data in the column (n3) in ascending or descending order. (For the sorting, the data table where the values (odd number and even number) to the right of (s) in each row increase consecutively is used.) The result is stored in the 32-bit binary data table (sorting result) of ((n1) × (n2)) points specified by (d).
- The following example shows a data table configuration when the sorting source data table has 3 rows and 4 columns ((n1) = 3, (n2) = 4). For the sorting result data table, understand (s) as (d).

Ex.

Data table configuration (rows (n1) = 3, columns (n2) = 4)

—	Column No.1	Column No.2	Column No.3	Column No.4
Row No.1	(s)+1, (s)	(s)+3, (s)+2	(s)+5, (s)+4	(s)+7, (s)+6
Row No.2	(s)+9, (s)+8	(s)+11, (s)+10	(s)+13, (s)+12	(s)+15, (s)+14
Row No.3	(s)+17, (s)+16	(s)+19, (s)+18	(s)+21, (s)+20	(s)+23, (s)+22

- The sorting order is determined according to the value (on or off) of SM774 (Data table sort instruction sort order) set at the start of execution of the instruction.

SM774 set value	Sorting order
OFF	Ascending order
ON	Descending order

- When the execution command is turned on, bit 11 of SD774 (Execution status of data table sort instructions) turns on and the sorting is started. After (n1) scans, data sorting completes at the execution of the END instruction and bit 11 of SD774 turns off at the same time that bit 9 (completion flag) of SD774 turns on. The completion flag turns off when the END instruction of the next scan is executed.
- The DSORTTBL2(_U) instruction cannot be executed while bit 11 of SD774 is on. (If the instruction is executed, no processing is performed.)

■Operation example

The following figures show operation examples when the following data table (rows (n1) = 5, columns (n2) = 4) is sorted under each condition.

		(n2)			
		Column No.1	Column No.2	Column No.3	Column No.4
		Control number	Dimension X	Dimension Y	Dimension Z
(n1)	Line No.1	(s)+1, (s) 1	(s)+3, (s)+2 150	(s)+5, (s)+4 45000	(s)+7, (s)+6 20
	Line No.2	(s)+9, (s)+8 2	(s)+11, (s)+10 180	(s)+13, (s)+12 50000	(s)+15, (s)+14 40
	Line No.3	(s)+17, (s)+16 3	(s)+19, (s)+18 160	(s)+21, (s)+20 70000	(s)+23, (s)+22 30
	Line No.4	(s)+25, (s)+24 4	(s)+27, (s)+26 100	(s)+29, (s)+28 20000	(s)+31, (s)+30 8
	Line No.5	(s)+33, (s)+32 5	(s)+35, (s)+34 150	(s)+37, (s)+36 50000	(s)+39, (s)+38 45

Point

Adding consecutive numbers such as control numbers in the first column helps to know the original row numbers.

Ex.

When (n3) is set to 2 and SM774 is off (ascending order)

		(n2)			
		Column No.1	Column No.2	Column No.3	Column No.4
		Control number	Dimension X	Dimension Y	Dimension Z
(n1)	Line No.1	(s)+1, (s) 1	(s)+3, (s)+2 150	(s)+5, (s)+4 45000	(s)+7, (s)+6 20
	Line No.2	(s)+9, (s)+8 2	(s)+11, (s)+10 180	(s)+13, (s)+12 50000	(s)+15, (s)+14 40
	Line No.3	(s)+17, (s)+16 3	(s)+19, (s)+18 160	(s)+21, (s)+20 70000	(s)+23, (s)+22 30
	Line No.4	(s)+25, (s)+24 4	(s)+27, (s)+26 100	(s)+29, (s)+28 20000	(s)+31, (s)+30 8
	Line No.5	(s)+33, (s)+32 5	(s)+35, (s)+34 150	(s)+37, (s)+36 50000	(s)+39, (s)+38 45

⇨

		(n2)			
		Column No.1	Column No.2	Column No.3	Column No.4
		Control number	Dimension X	Dimension Y	Dimension Z
(n1)	Line No.1	(d)+1, (d) 4	(d)+3, (d)+2 100	(d)+5, (d)+4 20000	(d)+7, (d)+6 8
	Line No.2	(d)+9, (d)+8 1	(d)+11, (d)+10 150	(d)+13, (d)+12 45000	(d)+15, (d)+14 20
	Line No.3	(d)+17, (d)+16 5	(d)+19, (d)+18 150	(d)+21, (d)+20 50000	(d)+23, (d)+22 45
	Line No.4	(d)+25, (d)+24 3	(d)+27, (d)+26 160	(d)+29, (d)+28 70000	(d)+31, (d)+30 30
	Line No.5	(d)+33, (d)+32 2	(d)+35, (d)+34 180	(d)+37, (d)+36 50000	(d)+39, (d)+38 40

Ex.

When (n3) is set to 3 and SM774 is on (descending order)

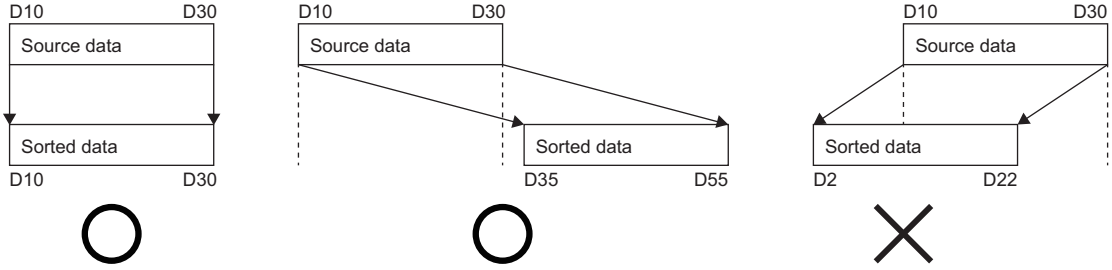
		(n2)			
		Column No.1	Column No.2	Column No.3	Column No.4
		Control number	Dimension X	Dimension Y	Dimension Z
(n1)	Line No.1	(s)+1, (s) 1	(s)+3, (s)+2 150	(s)+5, (s)+4 45000	(s)+7, (s)+6 20
	Line No.2	(s)+9, (s)+8 2	(s)+11, (s)+10 180	(s)+13, (s)+12 50000	(s)+15, (s)+14 40
	Line No.3	(s)+17, (s)+16 3	(s)+19, (s)+18 160	(s)+21, (s)+20 70000	(s)+23, (s)+22 30
	Line No.4	(s)+25, (s)+24 4	(s)+27, (s)+26 100	(s)+29, (s)+28 20000	(s)+31, (s)+30 8
	Line No.5	(s)+33, (s)+32 5	(s)+35, (s)+34 150	(s)+37, (s)+36 50000	(s)+39, (s)+38 45

⇨

		(n2)			
		Column No.1	Column No.2	Column No.3	Column No.4
		Control number	Dimension X	Dimension Y	Dimension Z
(n1)	Line No.1	(d)+1, (d) 3	(d)+3, (d)+2 160	(d)+5, (d)+4 70000	(d)+7, (d)+6 30
	Line No.2	(d)+9, (d)+8 2	(d)+11, (d)+10 180	(d)+13, (d)+12 50000	(d)+15, (d)+14 40
	Line No.3	(d)+17, (d)+16 5	(d)+19, (d)+18 150	(d)+21, (d)+20 50000	(d)+23, (d)+22 45
	Line No.4	(d)+25, (d)+24 1	(d)+27, (d)+26 150	(d)+29, (d)+28 45000	(d)+31, (d)+30 20
	Line No.5	(d)+33, (d)+32 4	(d)+35, (d)+34 100	(d)+37, (d)+36 20000	(d)+39, (d)+38 8

Precautions

- Do not change the contents of operands and data during operation. The correct result may not be obtained when sorting is completed.
- When the same device is specified by (s) and (d), the source data is overwritten by the sorting result. Do not change the contents of (s) until execution is completed (bit 9 of SD774 turns on).
- When specifying (s) and (d), specify the same device range or specify devices whose ranges do not overlap.



Operation error

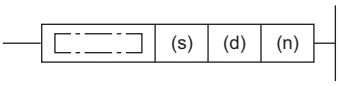
Error code (SD0)	Description
2820H	The device range specified by (s) exceeds the corresponding device range.
	The device range specified by (d) exceeds the corresponding device range.
2821H	The device ranges specified by (s) and (d) overlap partly.
3405H	The value specified by (n1) is outside the following range. 1 to 32
	The value specified by (n2) is outside the following range. 1 to 6
	The value specified by (n3) is outside the following range. 1 to (n2)

Adding 16-bit binary data

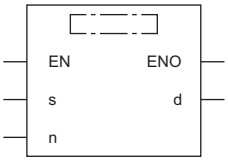
WSUM(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

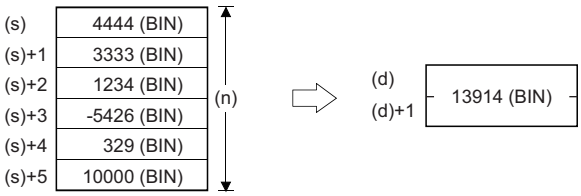
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

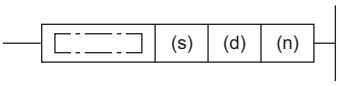
There is no operation error.

Adding 32-bit binary data

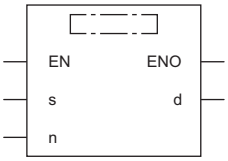
DWSUM(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

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

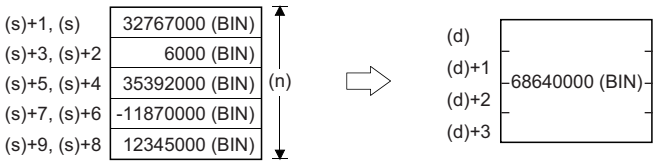
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

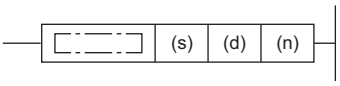
There is no operation error.

Calculating the mean value of 16-bit binary data

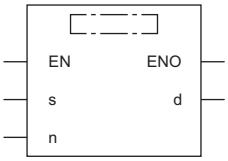
MEAN(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

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

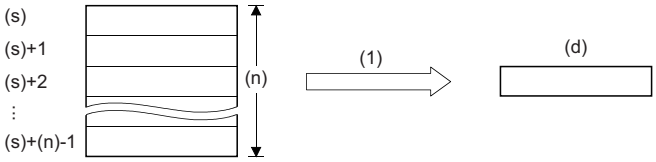
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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

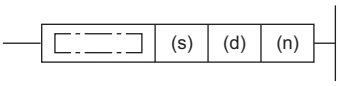
There is no operation error.

Calculating the mean value of 32-bit binary data

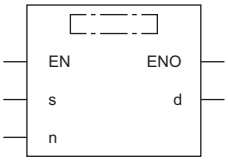
DMEAN(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

■Descriptions, ranges, and data types

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

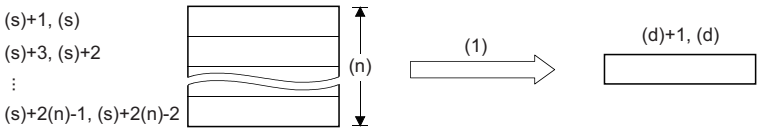
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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.

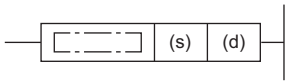
Calculating the square root of 16-bit binary data

SQRT(P)

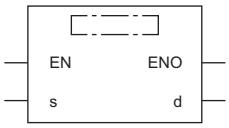
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

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 1443 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□\□, U3E□\(H)G□	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.

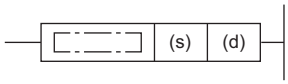
Calculating the square root of 32-bit binary data

DSQRT(P)

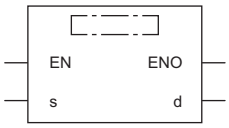
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

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□\□, U3E□\(H)G□	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.

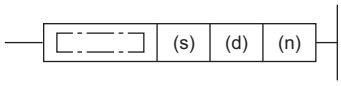
CRC operation

CRC(P)

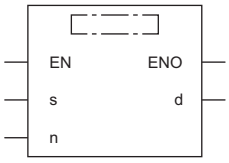
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

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□\□, U3E□\(H)G□	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.
- These instructions calculate the CRC (cyclic redundancy check) value which is an error check method used in communication. In addition to CRC, there are other methods to check an error, such as parity check and sum check (checksum). For obtaining the horizontal parity value and sum check value, the CCD(P) instruction is available. (☞ Page 776 CCD(P))
- 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
3405H	A digit other than 4 is specified in the digit-specified bit device in (s) and (d).

8.7 Check Code Instructions

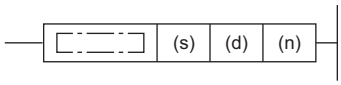
Check code

CCD(P)

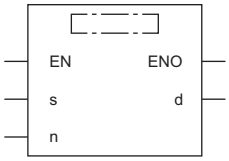
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions perform addition of the data stored in the devices specified by (s) to (s)+(n)-1 and calculate the horizontal parity, and stores the added data in the device specified by (d) and the horizontal parity in the device specified by (d)+1.

Ladder	ST
	ENO:=CCD(EN,s,n,d); ENO:=CCDP(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
CCD	
CCDP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device where target data is stored	—	16-bit signed binary	ANY16
(d)	Start device for storing the calculated data	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
(n)	Number of data	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□\□, U3E□\(H)G□	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 perform addition of the data stored in the devices specified by (s) to (s)+(n)-1 and calculate the horizontal parity, and stores the added data in the device specified by (d) and the horizontal parity in the device specified by (d)+1. 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.
- If an overflow occurs in the calculated result stored in (d), the following operations are performed. SM700 does not turn on.

32767+2=-32767
(7FFFH+0002H=8001H)

-32767+-2=32767
(8001H+FFFEH=7FFFH)

- These instructions calculate the horizontal parity value and sum check value as the error check methods used in communications. There is another check method called CRC (cyclic redundancy check) in addition to the CCD(P) instruction. (☞ Page 774 CRC(P))
- The operation in each conversion mode is described below.

■16-bit conversion mode (while SM772 is OFF)

With regard to (n) data points starting from (s), the addition data and horizontal parity data of upper 8 bits and lower 8 bits are stored to (d) and (d)+1 respectively.

Ex.

When (n)=6

<Calculation of addition data value>

In 16-bit conversion mode, addition data is determined by adding 6 bytes in the following shaded portion. The addition data is thus determined as "0315H", and therefore "0315H" 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

<Calculation of horizontal parity value>

In 16-bit conversion mode, the above shaded portion becomes the horizontal parity calculation target.

The number of ON (1) bits is calculated to determine the parity value which becomes ON (1) when the number of ON (1) bits is finally odd or OFF (0) when it is finally even. The horizontal parity value is stored in the device specified by (d)+1.

In the following table, 5FH is stored in the device specified by (d)+1.

Bit/horizontal parity value	b7	b6	b5	b4	b3	b2	b1	b0	Value
Upper 8 bits of D0	0	1	1	0	0	0	0	1	61H
Lower 8 bits of D0	0	1	1	0	0	1	0	0	64H
Upper 8 bits of D1	0	0	0	1	0	0	0	0	10H
Lower 8 bits of D1	0	1	1	1	1	0	1	1	7BH
Upper 8 bits of D2	1	1	1	1	1	0	1	0	FAH
Lower 8 bits of D2	1	1	0	0	1	0	1	1	CBH
Horizontal parity value	0	1	0	1	1	1	1	1	5FH

■8-bit conversion mode (while SM772 is ON)

With regard to (n) data points starting from (s), the addition data and horizontal parity data of only low-order 8 bits are stored to (d) and (d)+1 respectively.

Ex.

When (n)=6

<Calculation of addition data value>

In 8-bit conversion mode, addition data is determined by adding 6 bytes in the following shaded portion. The addition data is thus determined as "03B2H", and therefore "03B2H" 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

<Calculation of horizontal parity value>

In 8-bit conversion mode, the above shaded portion becomes the horizontal parity calculation target. The number of ON (1) bits is calculated to determine the parity value which becomes ON (1) when the number of ON (1) bits is finally odd or OFF (0) when it is finally even. The horizontal parity value is stored in the device specified by (d)+1.

In the following table, C2H is stored in the device specified by (d)+1.

Bit/horizontal parity value	b7	b6	b5	b4	b3	b2	b1	b0	Value
Lower 8 bits of D0	0	1	1	0	0	1	0	0	64H
Lower 8 bits of D1	0	1	1	1	1	0	1	1	7BH
Lower 8 bits of D2	1	1	0	0	1	0	1	1	CBH
Lower 8 bits of D3	1	1	1	1	1	1	1	1	FFH
Lower 8 bits of D4	1	1	1	1	1	0	0	1	F9H
Lower 8 bits of D5	0	0	0	1	0	0	0	0	10H
Horizontal parity value	1	1	0	0	0	0	1	0	C2H

Operation error

There is no operation error.

9 DEBUGGING AND FAILURE DIAGNOSTIC

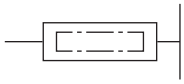
9.1 Debugging and Failure Diagnostic Instructions

Resetting the error display and the annunciator display

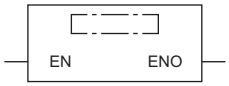
LEDR

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This instruction resets the self-diagnostic error (continuation error) display and the annunciator display of the CPU module.

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 CPU module. Executing the instruction once resets both the error display and the annunciator display.
- When a self-diagnostic error has occurred, the CPU module operates as follows:
  - When a self-diagnostic error (continuation error) has occurred

  The ERROR LED on the front of the CPU module 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 CPU module turns off. At this time, SM51 also turns off.
- When the annunciator (F) is on, the CPU module 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

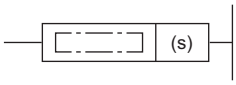
There is no operation error.

Generating a continuation error

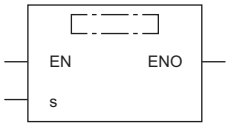
PALERT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

This instruction generates a continuation error in the CPU module.

Ladder	ST
	ENO:=PALERT(EN,s); ENO:=PALERTP(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
PALERT	
PALERTP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Data to be stored in the detailed information 2 of the error code 1810H or device number where the data is stored	-32768 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

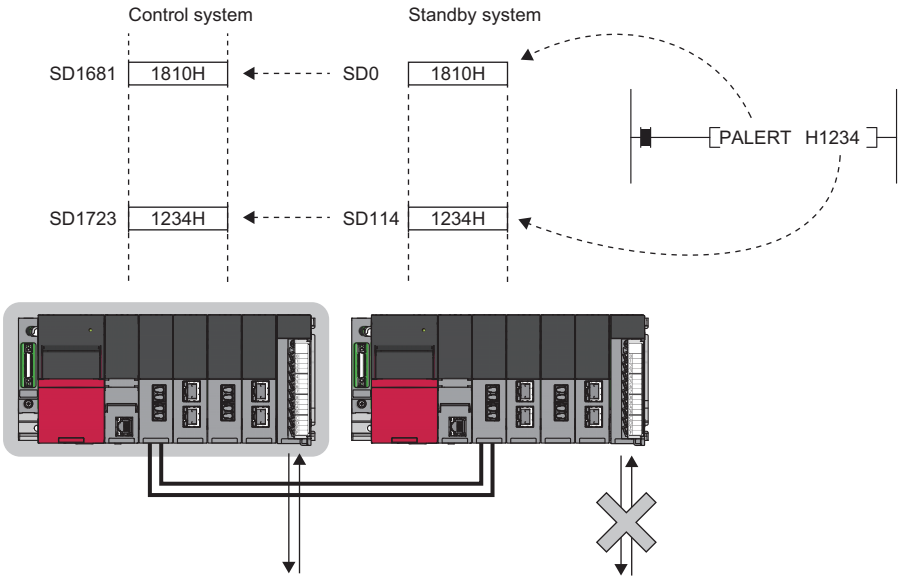
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	○	—	—	○	○	—	—	—

Processing details

- This instruction generates a continuation error (error code: 1810H) in the CPU module. When the instruction is executed, SM0 turns on and the ERROR LED of the CPU module turns on. SM1 does not turn on.
- Data (16-bit signed binary) specified by (s) is stored in the detailed information 2 of the error code 1810H.

Point

- The PALERT(P) instruction is useful for debugging since a continuation error can be simulated at the start-up of the system. The instruction execution point can be identified by checking the detailed information 2 of the error code 1810H.
- An error in an external device or network can be detected even in the standby system by using the PALERT(P) instruction in the program executed in both systems. The error code of the PALERT (P) instruction executed in the standby system and data to be stored in the detailed information 2 are stored in the special registers in both the control system and standby system. Thus, the error definition detected in the standby system can be checked with SD1681 (Latest self-diagnostic error code (the other system)) and SD1723 (Detailed information 2 (the other system)) of the control system. (Only the latest error code of the standby system can be checked in the control system.)



Operation error

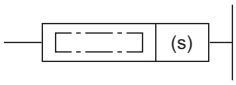
There is no operation error, except the error (error code: 1810H) which occurs by executing the PALERT(P) instruction.

Generating a stop error

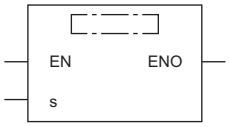
PABORT

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

This instruction stops program execution and generates a stop error in the CPU module.

Ladder	ST
	ENO:=PABORT(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
PABORT	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Data to be stored in the detailed information 2 of the error code 3070H or device number where the 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	○	—	—	○	○	—	—	—

Processing details

- This instruction stops program execution and generates a stop error (error code: 3070H) in the CPU module. When the instruction is executed, SM0 turns on and the ERROR LED of the CPU module flashes. SM1 does not turn on.
- Data (16-bit signed binary) specified by (s) is stored in the detailed information 2 of the error code 3070H.

Point

- The PABORT instruction is useful for debugging since a stop error can be simulated at the start-up of the system. The instruction execution point can be identified by checking the detailed information 2 of the error code 3070H.
- If any problem occurs in an external device connected to the standby system, the PABORT instruction prevents the systems to be switched by generating a stop error in the standby system.
- An error in an external device or network can be detected even in the standby system by using the PABORT instruction in the program executed in both systems. The error code of the PABORT instruction executed in the standby system and the data to be stored in the detailed information 2 can be checked in the control system. (☞ Page 781 PALERT(P))

Operation error

There is no operation error, except the error (error code: 3070H) which occurs by executing the PABORT instruction.

10 STRING PROCESSING

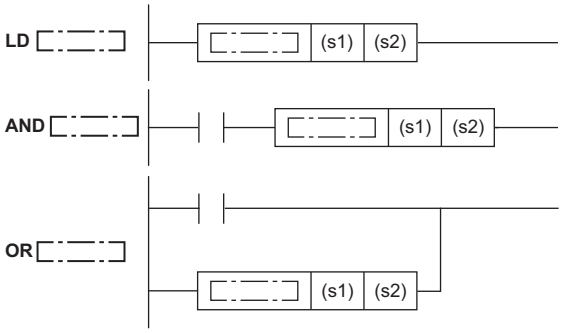
10.1 String Processing Instructions

Comparing string data

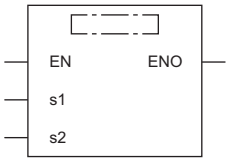
LD$□, AND$□, OR$□

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions compare string data as normally open contacts.

Ladder	ST*1
 (□ 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.)*2

FBD/LD



(□ is replaced by a combination of LDSTRING_, ANDSTRING_, or ORSTRING_ and EQ, NE, GT, LE, LT, or GE.)*2

*1 The engineering tool with version "1.035M" or later supports the ST.
*2 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

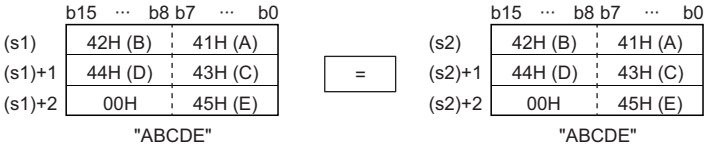
Operand	Description	Range	Data type	Data type (label)
(s1)	Comparison data or the start device where the comparison data is stored	—	String	ANYSTRING_SINGLE
(s2)	Comparison data or the start device where the comparison data is stored	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—

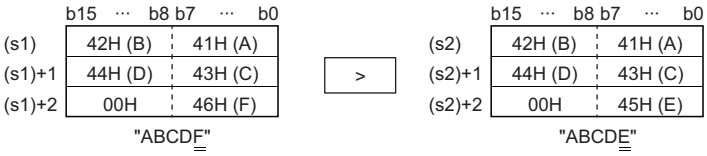
Processing details

- These instructions perform a comparison operation between the character string data in the device specified by (s1) and the character string data in the device specified by (s2). (Devices are used as normally open contacts).
- In comparison operation, the ASCII code of character string is compared character by character from the beginning of the character string.
- The character 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 character strings match.



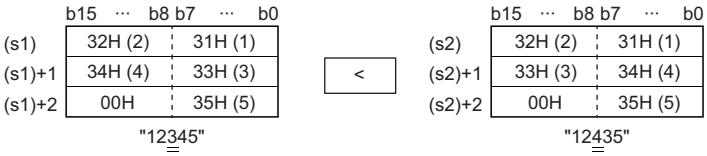
□ Instruction symbol (ladder, FBD/LD)	Result
$=, EQ	Continuity state (ENO is on.)
$<>, NE	Non-continuity state (ENO is off.)
$>, GT	Non-continuity state (ENO is off.)
$<=, LE	Continuity state (ENO is on.)
$<, LT	Non-continuity state (ENO is off.)
$>=, GE	Continuity state (ENO is on.)

  - When different character strings are compared, the character string with a larger character code is greater.



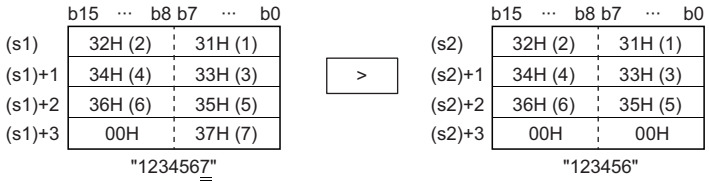
□ Instruction symbol (ladder, FBD/LD)	Result
$=, EQ	Non-continuity state (ENO is off.)
$<>, NE	Continuity state (ENO is on.)
$>, GT	Continuity state (ENO is on.)
$<=, LE	Non-continuity state (ENO is off.)
$<, LT	Non-continuity state (ENO is off.)
$>=, GE	Continuity state (ENO is on.)

  - When different character strings are compared, the relative size of a character 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 character string data in the devices specified by (s1) and (s2) are different, the longer character 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.)

- The character string in the device specified by (s1) or (s2) exceeds 16383 characters, the operation result will be non-continuity (ENO 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.

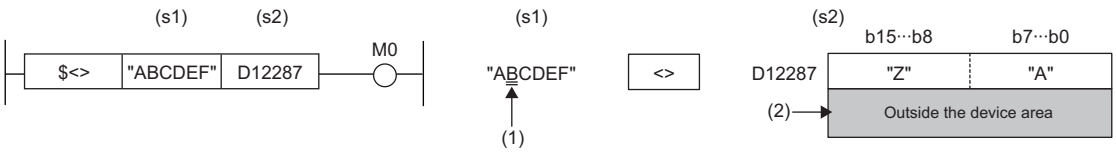
Point

The character string comparison instructions perform the following checks while comparing character strings.

- Checking whether the device area range is exceeded
- Checking whether the character string is within 16383 characters

If 00H does not exist in the device area or the character string exceeds 16383 characters and a character mismatch is detected, the instruction outputs comparison operation results without causing non-continuity (ENO 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 OFF).

(2) D12287 and later are outside the device area, and accordingly character string data comparison is performed using data up to D12287.

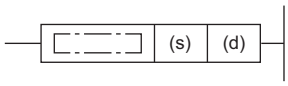
Since a character string mismatch has been detected, the condition is satisfied and processing ends.

Concatenating string data

$+(P) [when two operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions concatenate string data.

Ladder	ST
	Not supported (☞ Page 790 $+(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 790 $+(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
$+	
$+P	

Setting data

■Description, range, data type

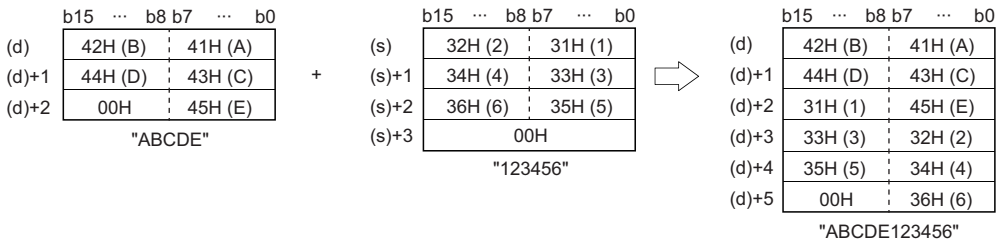
Operand	Description	Range	Data type	Data type (label)
(s)	Data to be concatenated or the start device containing the data	—	String	ANYSTRING_SINGLE
(d)	Start device where the data to be concatenated is stored	—	String	ANYSTRING_SINGLE

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- This instruction connects the character string stored in the device number specified by (s) and later to the end of the character 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 character strings, the instruction ignores 00H that indicates the end of the character string in the device specified by (d) and appends the character string in the device specified by (s) following the last character in the device specified by (d).

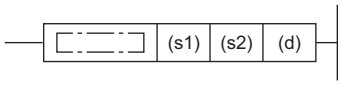
Operation error

Error code (SD0)	Description
2820H	There is no NULL code (00H) in the setting area specified by (s) and later in the device/label memory.
	There is no NULL code (00H) in the setting area specified by (d) and later in the device/label memory.
2821H	The device numbers for storing the strings in the devices specified by (s) and (d) are overlapping.
3405H	The number of characters in the string specified by (s) exceeds 16383.
	The number of characters in the string specified by (d) exceeds 16383.
3406H	The number of characters in the concatenated string ((s)+(d)) exceeds 16383.
	The entire string after concatenate processing cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

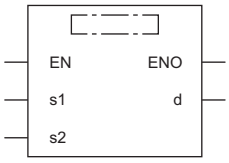
$+(P) [when three operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

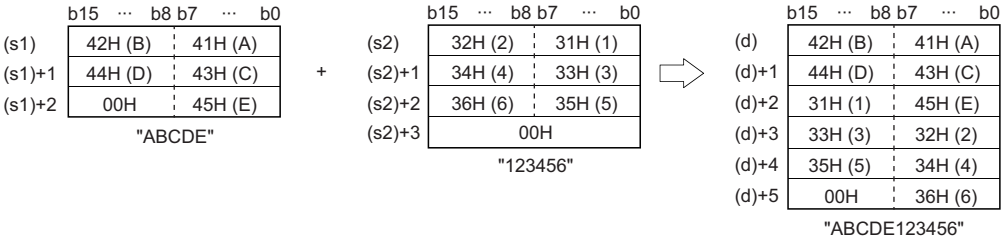
Operand	Description	Range	Data type	Data type (label)
(s1)	Data to be concatenated or the start device containing the data	—	String	ANYSTRING_SINGLE
(s2)	Data to be concatenated or the start device containing the data to be concatenated	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the concatenated 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- This instruction connects the character string stored in the device number specified by (s2) and later to the end of the character 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 character strings, the instruction ignores 00H that indicates the end of the character string in the device specified by (s1) and appends the character string in the device specified by (s2) following the last character in the device specified by (s1).

Operation error

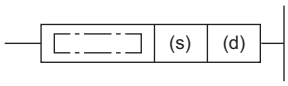
Error code (SD0)	Description
2820H	After the device number specified by (s1) and later, there is no 00H before the relevant device number.
	After the device number specified by (s2) and later, there is no 00H before the relevant device number.
2821H	The device numbers for storing the strings in the devices specified by (s2) and (d) are overlapping.
3405H	The number of characters in the string specified by (s1) exceeds 16383.
	The number of characters in the string specified by (s2) exceeds 16383.
3406H	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 cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

Transferring string data

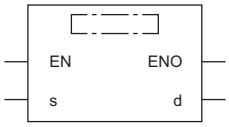
$MOV(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions transfer string data to the specified device number and later.

Ladder	ST*1
	ENO:=STRINGMOV(EN,s,d); ENO:=STRINGMOVP(EN,s,d);

FBD/LD



(□ is replaced by STRINGMOV or STRINGMOVP.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
$MOV	
$MOVP	

Setting data

■Descriptions, ranges, and data types

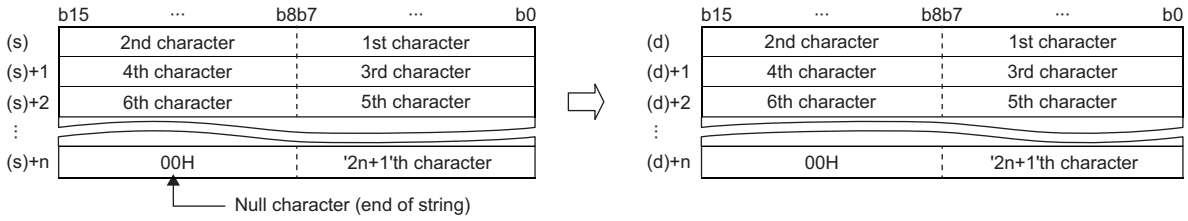
Operand	Description	Range	Data type	Data type (label)
(s)	Character string to be transferred (maximum of 255 characters) or the start device containing such character string	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the transferred character string	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

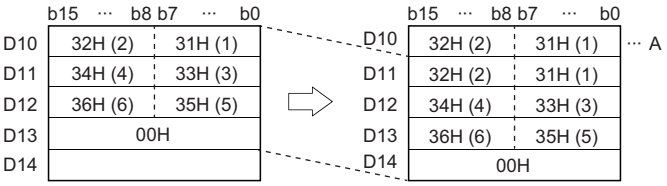
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	—	○	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions transfer the character string data in the device specified by (s) to the device number specified by (d) and later. The character strings specified by (s) or the character strings from the device number specified by (s) to the device number containing 00H are transferred all at once.

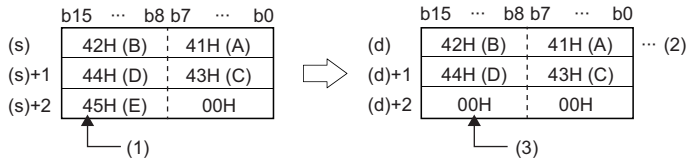


- Even when the device range ((s) to (s)+n) in which the character string data to be transferred and the device range ((d) to (d)+n) for storing the transferred data are overlapping, the processing is performed normally. For example, the character strings stored in the devices specified by D10 to D13 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

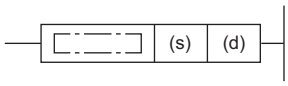
Error code (SD0)	Description
2820H	There is no NULL code (00H) in the setting area specified by (s) and later in the device/label memory.
3405H	The number of characters in the string specified by (s) exceeds 16383.
3406H	The entire string cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

Transferring Unicode string data

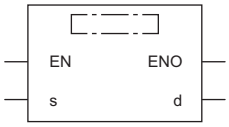
$MOV(P)_WS

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions transfer Unicode string data to the specified device number and later.

Ladder	ST*1
	ENO:=STRINGMOV_WS(EN,s,d); ENO:=STRINGMOVP_WS(EN,s,d);

FBD/LD



(☐ is replaced by STRINGMOV_WS or STRINGMOVP_WS.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
$MOV_WS	
$MOVP_WS	

Setting data

■Descriptions, ranges, and data types

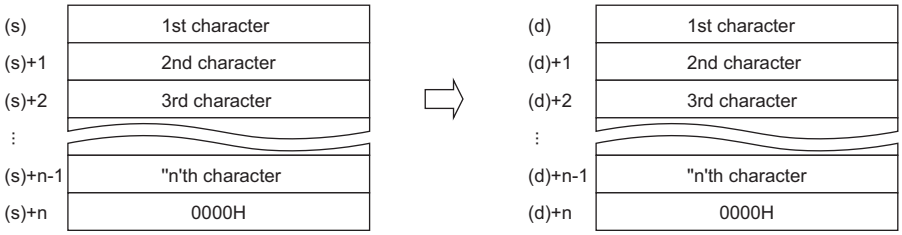
Operand	Description	Range	Data type	Data type (label)
(s)	Unicode character string to be transferred (maximum of 255 characters) or the start device containing the Unicode character string	—	Unicode string	ANYSTRING_DOUBLE
(d)	Start device for storing the transferred Unicode character string	—	Unicode string	ANYSTRING_DOUBLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

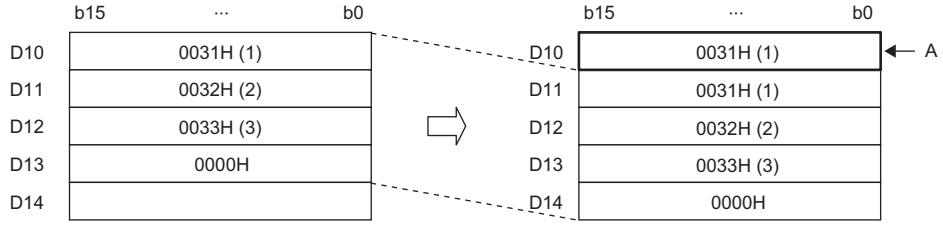
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J☐\☐	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U☐\G☐, J☐\☐, U3E☐\(H)G☐	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	○	—	○	—	—	○	—
(d)	—	—	○	—	—	○	—	○	—	—	—	—

Processing details

- These instructions transfer the Unicode character string data in the device specified by (s) to the device number specified by (d) and later. The Unicode character strings specified by (s) or the Unicode character strings from the device number specified by (s) to the device number containing 0000H are transferred all at once.



- Even when the device range ((s) to (s)+n)) in which the Unicode character 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 character 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

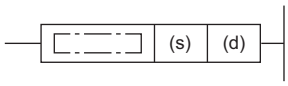
Error code (SD0)	Description
2820H	There is no 0000H in the setting area specified by (s) and later in the device/label memory.
3405H	The number of characters in the Unicode string specified by (s) exceeds 16383.
3406H	The entire Unicode string cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

Converting 16-bit binary data to decimal ASCII

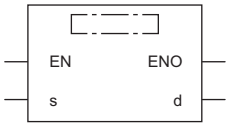
BINDA(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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*1	Storage format of (d)	Reference
OFF	Data is stored in a fixed number of digits (a sign + 5 digits)	Page 798 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 798 Operation of when SM705 (Number of conversion digits selection) is on

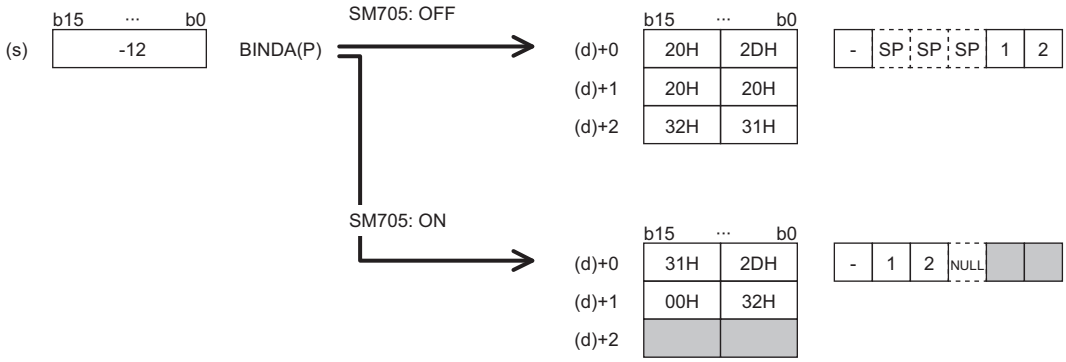
*1 For the firmware version of the CPU module supporting SM705, refer to the following.
MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

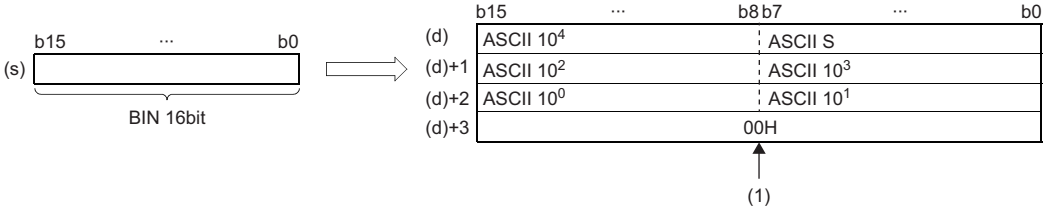
When the BINDA(P) instruction is executed with the numeric value "-12" stored in (s)



- 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.

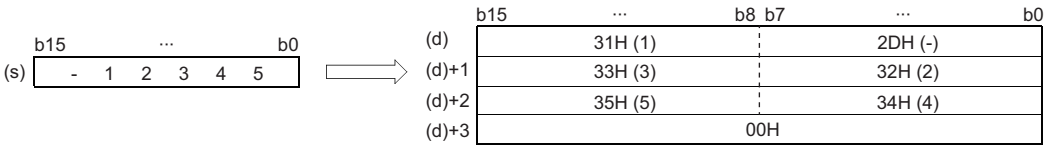
*3 When the length of the numeric part is 3 digits or less, 20H (space) is stored in ASCII 10^3.

*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).

Value of (s)	Data of (d) to (d)+2	Value of (s)	Data of (d) to (d)+2
• 0 • Positive value (1 digit in numeric part)	• The upper byte of (d) is filled with 00H. • Data in (d)+1 and (d)+2 remains unchanged. b15 ··· b8 b7 ··· b0 (d) 00H ASCII 10^0 (d)+1 (d)+2	• Positive value (2 digits in numeric part) • Negative value (1 digit in numeric part)	• (d)+1 is filled with 00H. • Data in (d)+2 remains unchanged. b15 ··· b8 b7 ··· b0 (d) ASCII 10^0 ASCII 10^1 / 2DH (-) (d)+1 00H (d)+2
⋮			
• Positive value (5 digits in numeric part) • Negative value (4 digits in numeric part)	• The upper byte of (d)+2 is filled with 00H. b15 ··· b8 b7 ··· b0 (d) ASCII 10^3 ASCII 10^4 / 2DH (-) (d)+1 ASCII 10^1 ASCII 10^2 (d)+2 00H ASCII 10^0	• Negative value (5 digits in numeric part)	• (1): (d)+3 is filled with 00H only when SM701 (Number of output characters selection) is off. b15 ··· b8 b7 ··· b0 (d) ASCII 10^4 2DH (-) (d)+1 ASCII 10^2 ASCII 10^3 (d)+2 ASCII 10^0 ASCII 10^1 (d)+3 00H (1)

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.

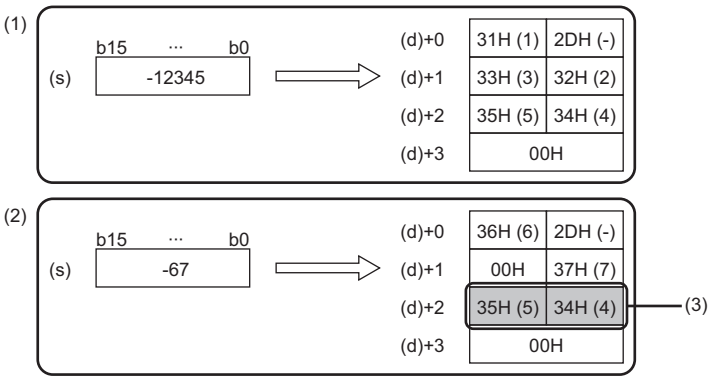
Operation error

There is no operation error.

Precautions

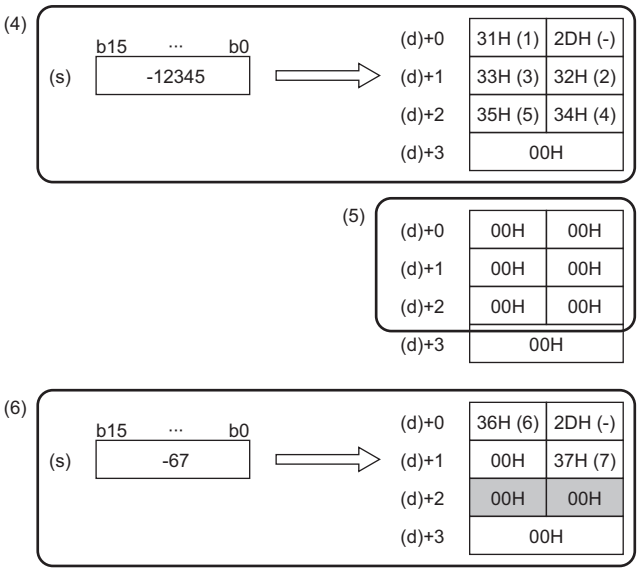
When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the BINDA(P)(_U) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation 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.



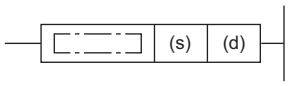
(4) "-12345" is converted into a string.
(5) (d)+0 to (d)+2 are cleared.
(6) "-67" is converted into a string.

Converting 32-bit binary data to decimal ASCII

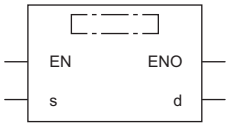
DBINDA(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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*1	Storage format of (d)	Reference
OFF	Data is stored in a fixed number of digits (a sign + 10 digits)	Page 802 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 803 Operation of when SM705 (Number of conversion digits selection) is on

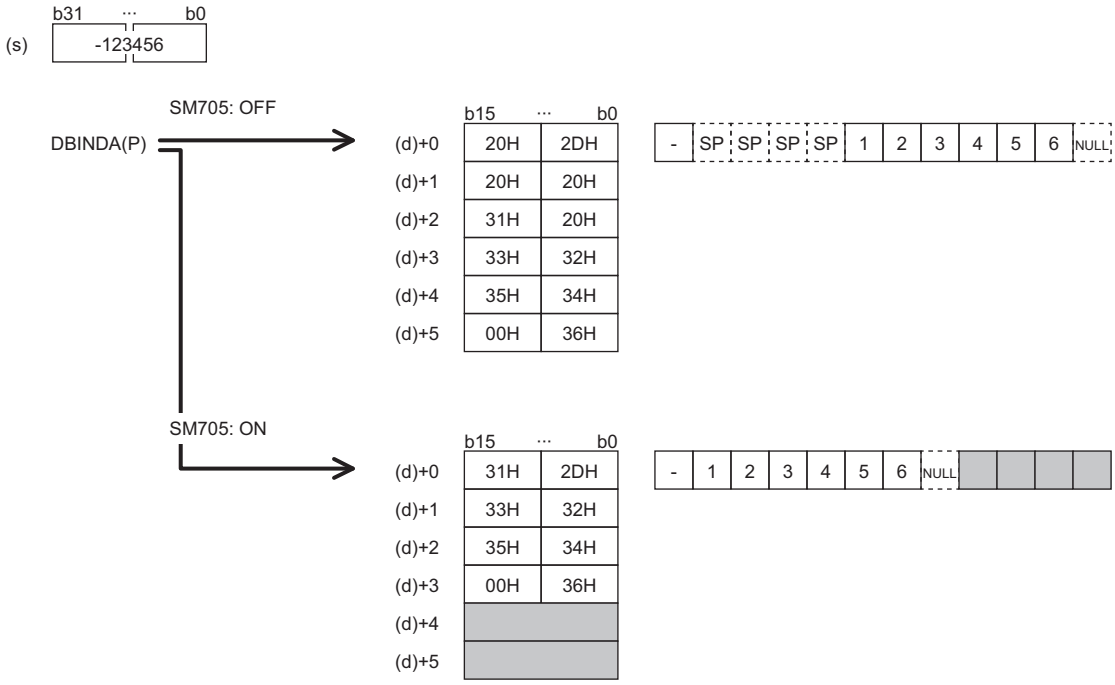
*1 For the firmware version of the CPU module supporting SM705, refer to the following.
MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

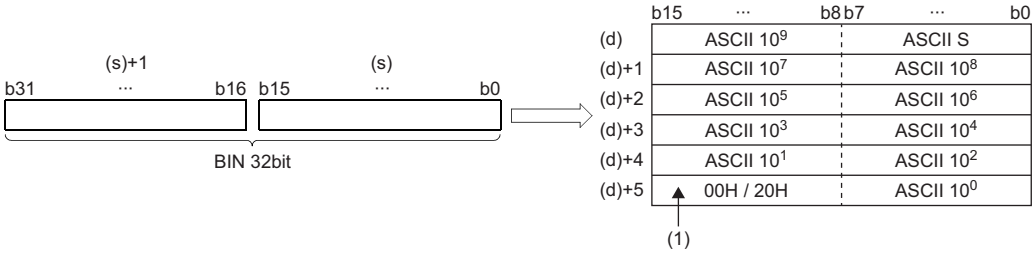
When the DBINDA(P) instruction is executed with a numeric value "-123456" stored in (s)



- 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.

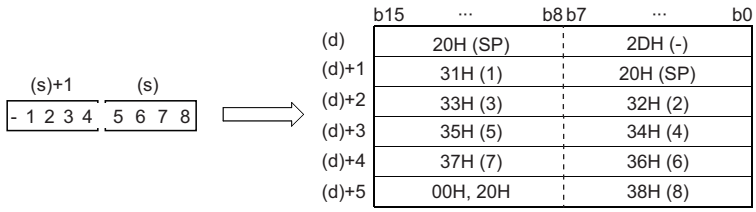
*2 When the length of the numeric part is 9 digits or less, 20H (space) is stored in ASCII 10^9.

*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).

Value of (s)	Data of (d) to (d)+2	Value of (s)	Data of (d) to (d)+2
• 0 • Positive value (1 digit in numeric part)	• The upper byte of (d) is filled with 00H. • (d)+1 and later remain unchanged. b15 ··· b8 b7 ··· b0 (d): 00H / ASCII 10^0 (d)+1 (d)+2 (d)+3 (d)+4 (d)+5	• Positive value (2 digits in numeric part) • Negative value (1 digit in numeric part)	• (d)+1 is filled with 00H. • (d)+2 and later remain unchanged. b15 ··· b8 b7 ··· b0 (d): ASCII 10^0 / ASCII 10^1 / 2DH (-) (d)+1: 00H (d)+2 (d)+3 (d)+4 (d)+5
⋮			
• Positive value (9 digits in numeric part) • Negative value (8 digits in numeric part)	• The upper byte of (d)+4 is filled with 00H. • (d)+5 and later remain unchanged. b15 ··· b8 b7 ··· b0 (d): ASCII 10^7 / ASCII 10^8 / 2DH (-) (d)+1: ASCII 10^5 / ASCII 10^6 (d)+2: ASCII 10^3 / ASCII 10^4 (d)+3: ASCII 10^1 / ASCII 10^2 (d)+4: 00H / ASCII 10^0 (d)+5	• Positive value (10 digits in numeric part) • Negative value (9 digits in numeric part)	• (d)+5 is filled with 00H. b15 ··· b8 b7 ··· b0 (d): ASCII 10^8 / ASCII 10^9 / 2DH (-) (d)+1: ASCII 10^6 / ASCII 10^7 (d)+2: ASCII 10^4 / ASCII 10^5 (d)+3: ASCII 10^2 / ASCII 10^3 (d)+4: ASCII 10^0 / ASCII 10^1 (d)+5: 00H
• Negative value (10 digits in numeric part)	• 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. b15 ··· b8 b7 ··· b0 (d): ASCII 10^9 / 2DH (-) (d)+1: ASCII 10^7 / ASCII 10^8 (d)+2: ASCII 10^5 / ASCII 10^6 (d)+3: ASCII 10^3 / ASCII 10^4 (d)+4: ASCII 10^1 / ASCII 10^2 (d)+5: 00H / 20H (1) / ASCII 10^0	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	

- 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

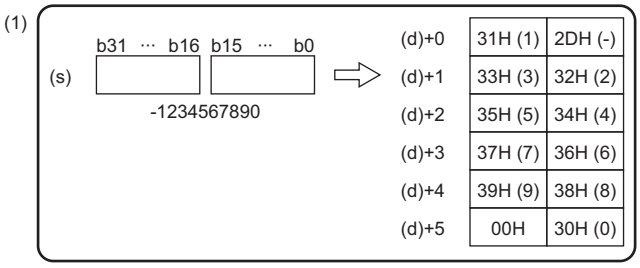
There is no operation error.

Precautions

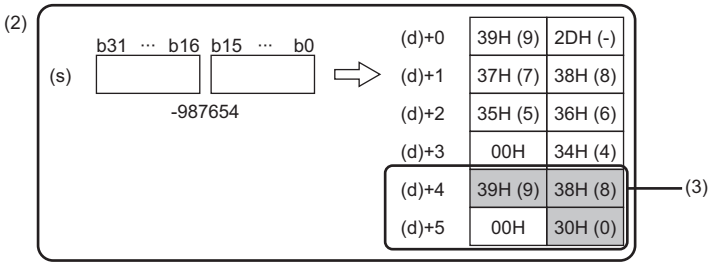
When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the DBINDA(P)(_U) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation 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)



(2)



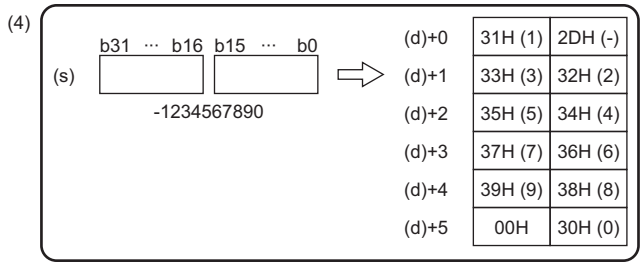
(1) "-1234567890" is converted into a string.

(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.

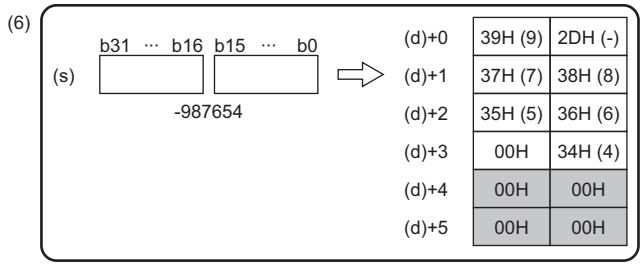
(4)



(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)



(4) "-1234567890" is converted into a string.

(5) (d)+0 to (d)+5 are cleared.

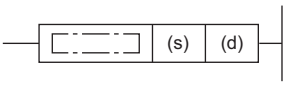
(6) "-987654" is converted into a string.

Converting 16-bit binary data to hexadecimal ASCII

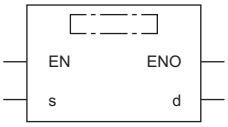
BINHA(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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*1	Storage format of (d)	Reference
OFF	Data is stored in a fixed number of digits (4 digits)	Page 807 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 807 Operation of when SM705 (Number of conversion digits selection) is on

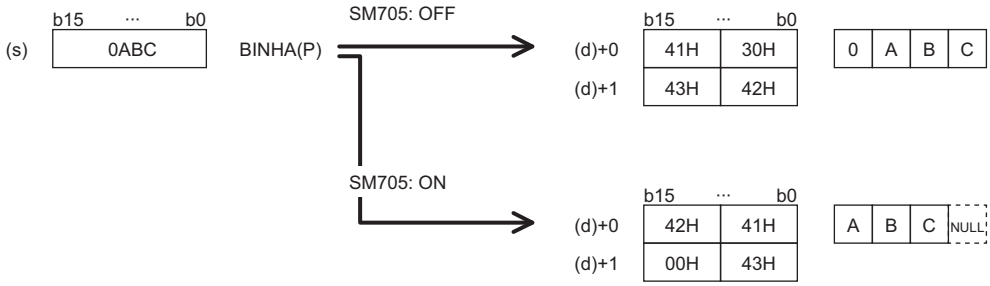
*1 For the firmware version of the CPU module supporting SM705, refer to the following.
MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

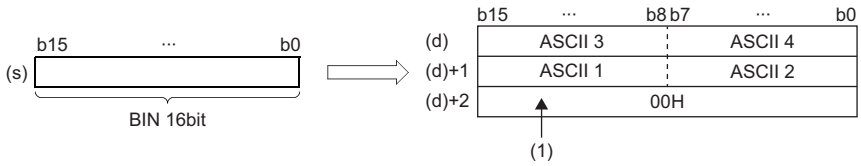
When the BINHA(P) instruction is executed with the 16-bit binary data "0ABCH" stored in (s)



- 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).



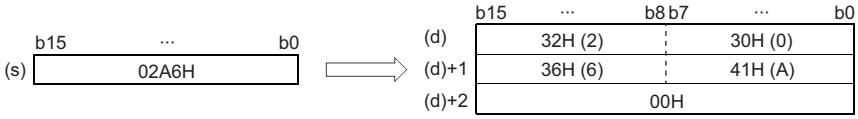
ASCII □: ASCII code (□th digit)

(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. b15 ··· b8 b7 ··· b0 (d) 00H ASCII 1 (d)+1	• 10H to FFH	• (d)+1 is filled with 00H. b15 ··· b8 b7 ··· b0 (d) ASCII 1 ASCII 2 (d)+1 00H
• 100H to FFFH	• The upper byte of (d)+1 are filled with 00H. b15 ··· b8 b7 ··· b0 (d) ASCII 2 ASCII 3 (d)+1 00H ASCII 1	• 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. b15 ··· b8 b7 ··· b0 (d) ASCII 3 ASCII 4 (d)+1 ASCII 1 ASCII 2 (d)+2 00H (1)

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.

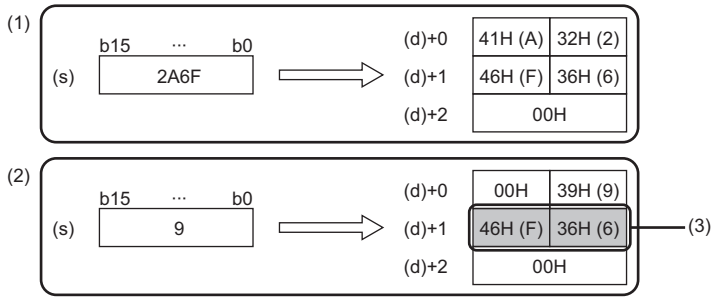
Operation error

There is no operation error.

Precautions

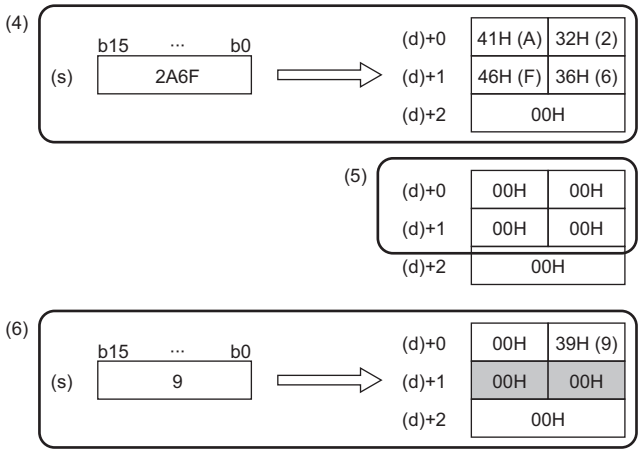
When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the BINHA(P) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation 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.



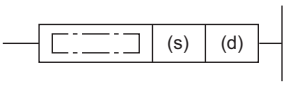
(4) "2A6F" is converted into a string.
(5) (d)+0 to (d)+1 are cleared.
(6) "9" is converted into a string.

Converting 32-bit binary data to hexadecimal ASCII

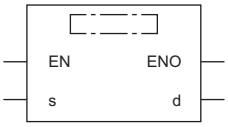
DBINHA(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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*1	Storage format of (d)	Reference
OFF	Data is stored in a fixed number of digits (8 digits)	Page 811 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 811 Operation of when SM705 (Number of conversion digits selection) is on

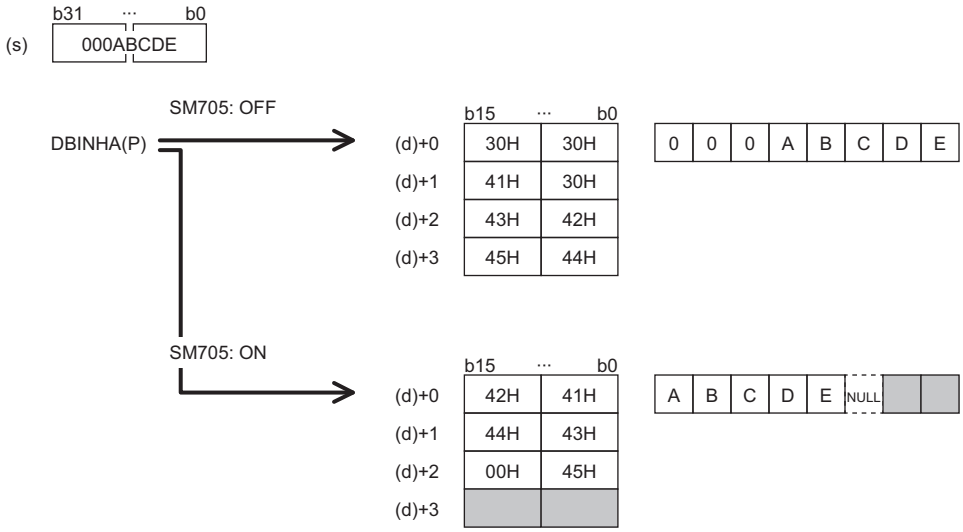
*1 For the firmware version of the CPU module supporting SM705, refer to the following.
MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

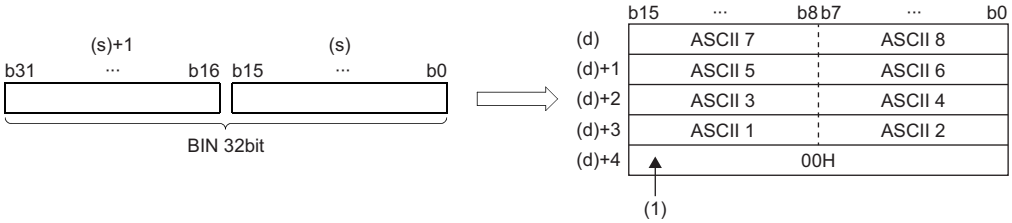
When the DBINHA(P) instruction is executed with a 32-bit binary data "000ABCDEH" stored in (s)



- 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.



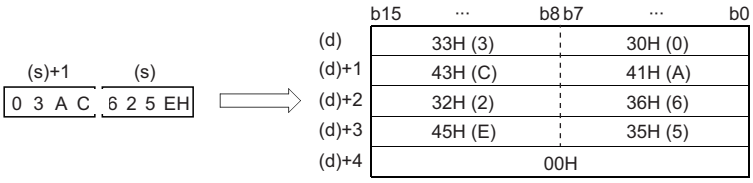
ASCII □: ASCII code (□th digit)

(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.

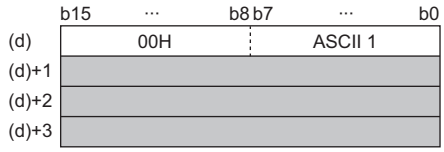
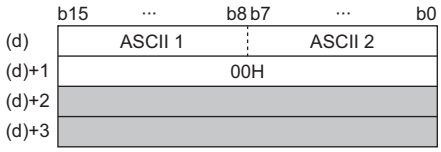
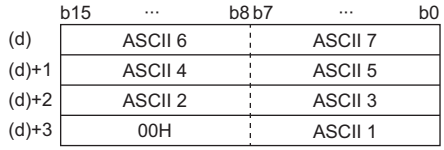
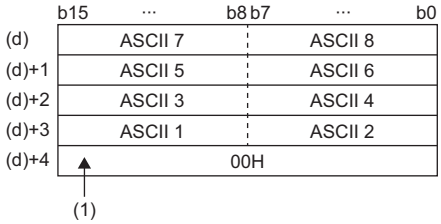
When 03AC625EH is stored in the device specified by (s)



■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.

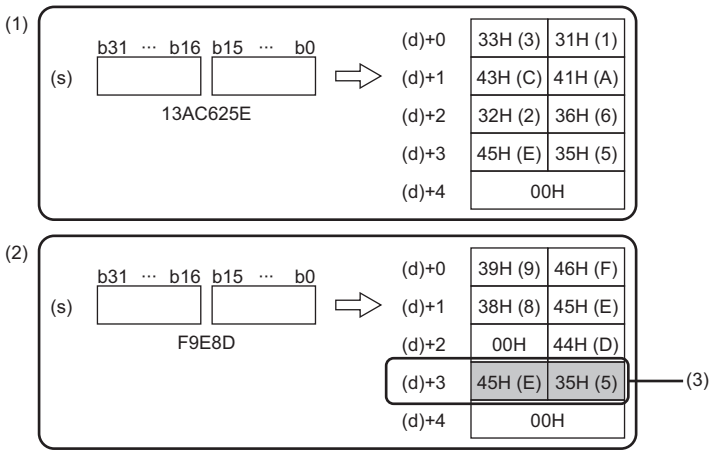
Operation error

There is no operation error.

Precautions

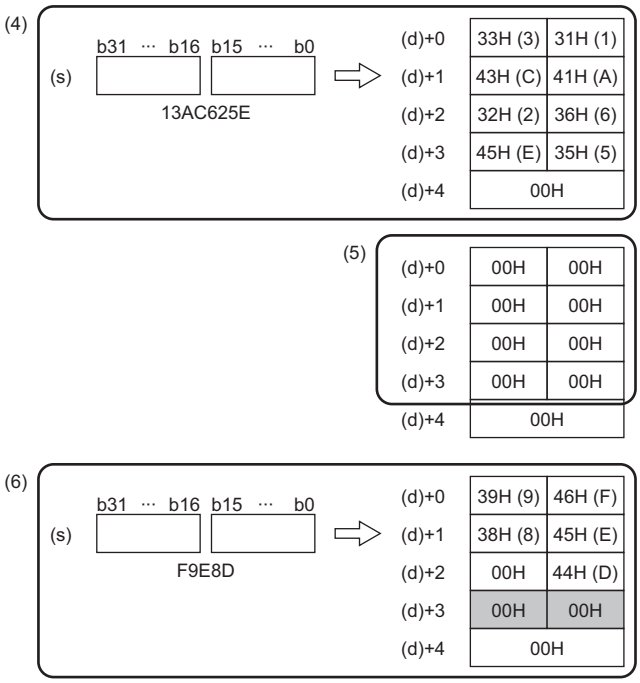
When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the DBINHA(P) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation 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.



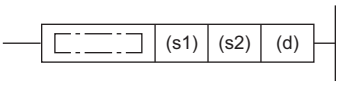
(4) "13AC625E" is converted into a string.
(5) (d)+0 to (d)+3 are cleared.
(6) "F9E8D" is converted into a string.

Converting 16-bit binary data to string data

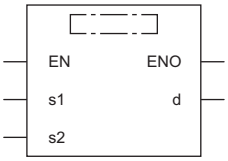
STR(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

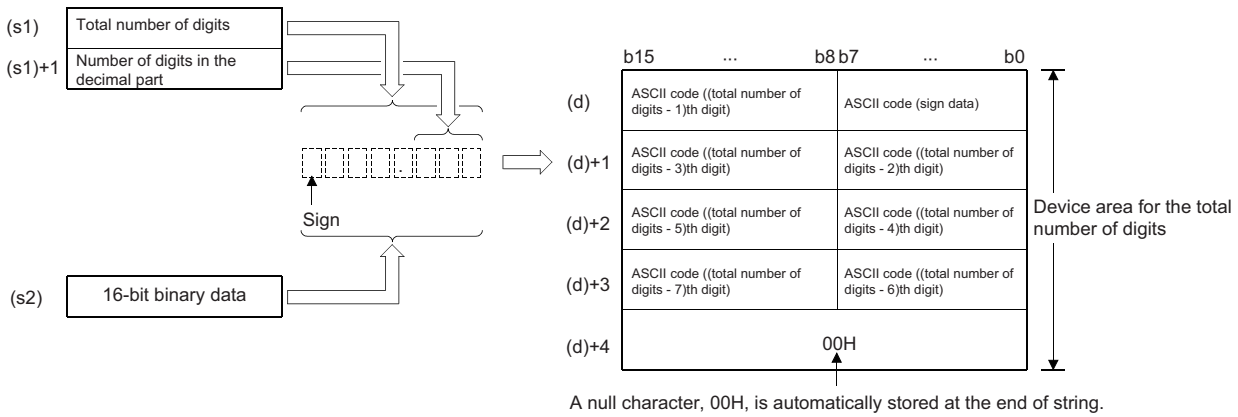
Operand		Description	Range	Data type	Data type (label)
(s1)	STR(P)	Start device where the number of digits of the conversion target data is stored	—	16-bit signed binary	ANY16_S_ARRAY (Number of elements: 2)
	STR(P)_U			16-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 2)
(s2)	STR(P)	Conversion 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 character 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

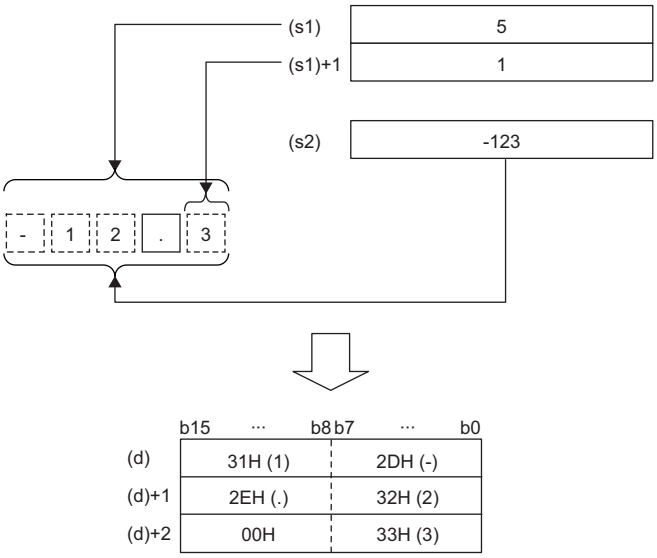
Processing details

- 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 character string data, and store the converted data in the device areas specified by (d) and later.



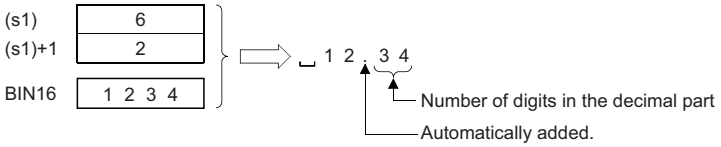
Ex.

When converting data "-123" in (s2) into a string assuming that the number of decimal places is one ("-12.3")

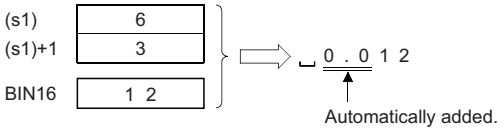


- The total number of digits that can be specified by (s1) is 2 to 8.
- The number of digits in the decimal part that can be specified by (s1)+1 is 0 to 5. Note that the number of digits in the decimal part must be smaller than the total number of digits minus 3.

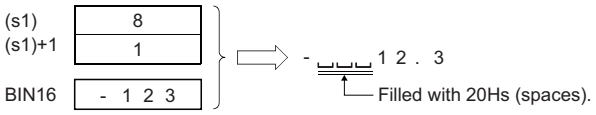
- The converted character string data are stored in the device areas specified by (d) and later as shown below.
  - As sign data, "20H" (space) is stored if the 16-bit binary data is positive, and "2DH" (-) is stored if the data is negative.
  - If the number of digits in the decimal part is set to other than 0, "2EH" (.) is automatically stored at the position before the specified number of digits. 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 character string.

Operation error

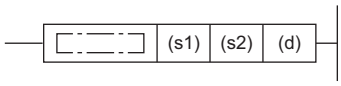
Error code (SD0)	Description
3401H	Invalid data that cannot be converted is input to (s1). • The specified total number of digits is out of the range, 2 to 8. • The specified number of digits in the decimal part is out of the range, 0 to 5. • The relationship between the total number of digits specified 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 (.)]

Converting 32-bit binary data to string data

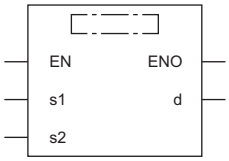
DSTR(P)(_U)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

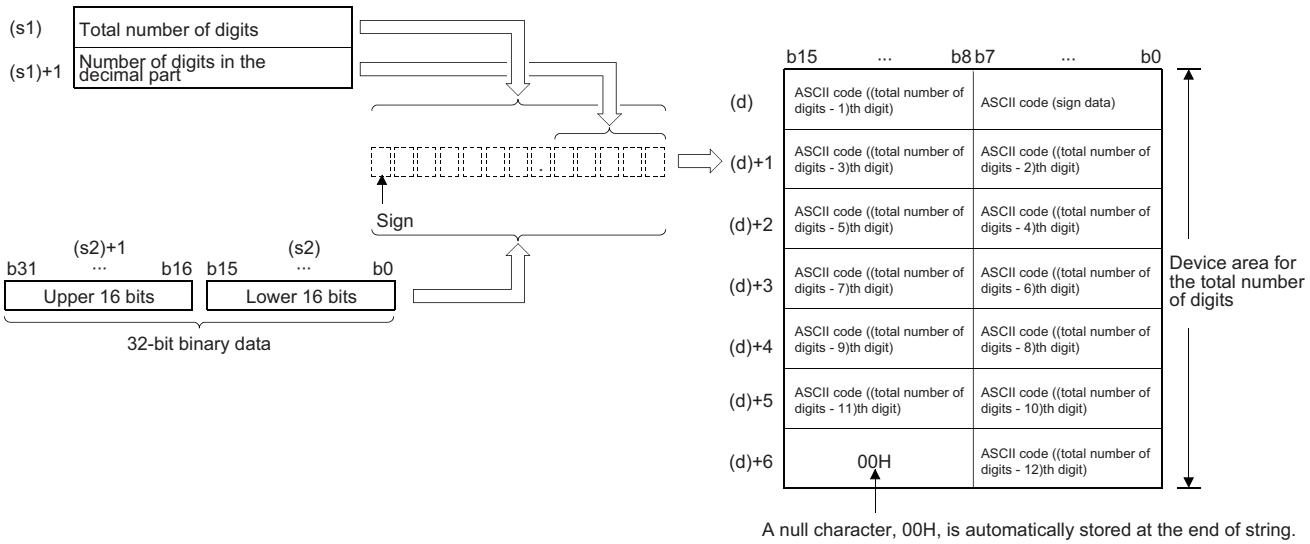
Operand		Description	Range	Data type	Data type (label)
(s1)	DSTR(P)	Start device where the number of digits of the conversion target data is stored	—	16-bit signed binary	ANY16_S_ARRAY (Number of elements: 2)
	DSTR(P)_U			16-bit unsigned binary	ANY16_U_ARRAY (Number of elements: 2)
(s2)	DSTR(P)	Conversion 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 character 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

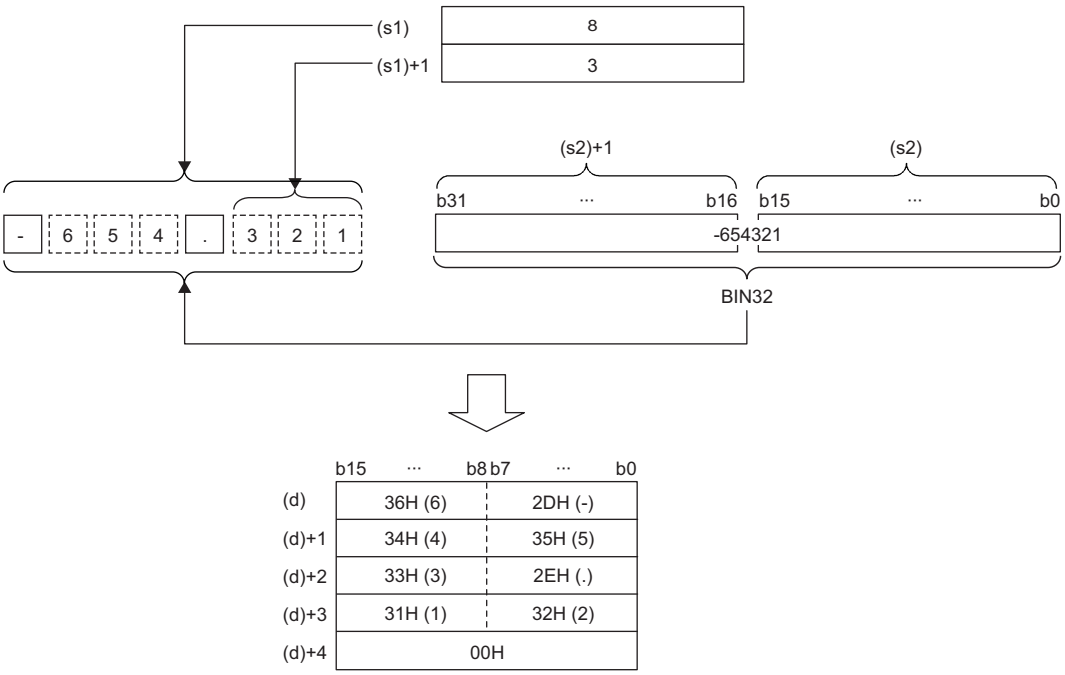
Processing details

- 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 character string data, and store the converted data in the device areas specified by (d) and later.



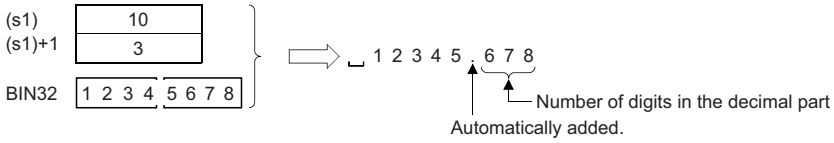
Ex.

When converting data "-654321" in (s2) into a string assuming that the number of decimal places is three ("-654.321")

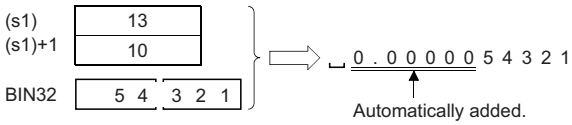


- The total number of digits that can be specified by (s1) is 2 to 13.
- The number of digits in the decimal part that can be specified by (s1)+1 is 0 to 10. Note that the number of digits in the decimal part must be smaller than the total number of digits minus 3.

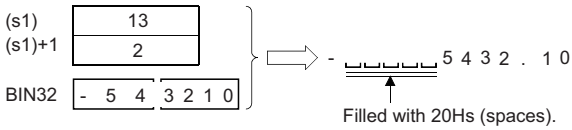
- The converted character string data are stored in the device areas specified by (d) and later as shown below.
  - As sign data, 20H (space) is stored if the 32-bit binary data is positive, and 2DH (-) is stored if the data is negative.
  - If the number of digits in the decimal part is set to other than 0, "2EH" (.) is automatically stored at the position before the specified number of digits. 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 character string.

Operation error

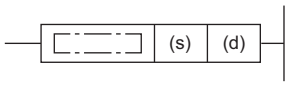
Error code (SD0)	Description
3401H	Invalid data that cannot be converted is input to (s1). • The specified total number of digits is out of the range, 2 to 13. • The specified number of digits in the decimal part is out of the range, 0 to 10. • The relationship between the total number of digits specified 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 (.)]

Converting BCD 4-digit data to decimal ASCII code

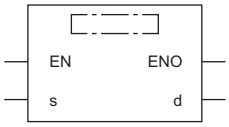
BCDDA(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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*1	Storage format of (d)	Reference
OFF	Data is stored in a fixed number of digits (4 digits)	Page 821 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 821 Operation of when SM705 (Number of conversion digits selection) is on

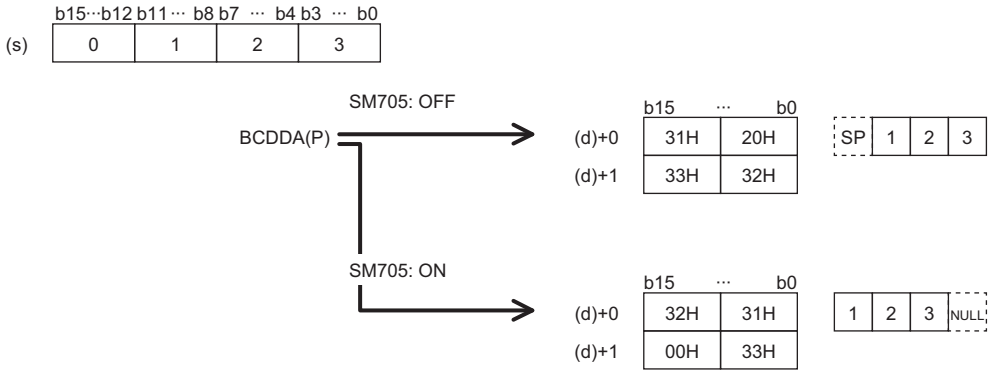
*1 For the firmware version of the CPU module supporting SM705, refer to the following.
MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

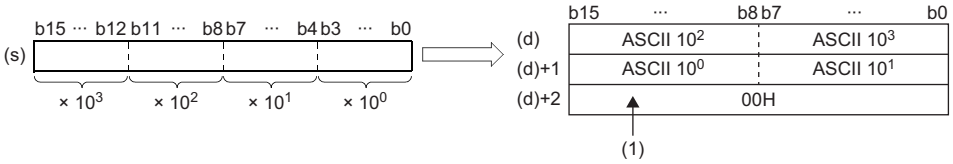
When the BCDDA(P) instruction is executed with BCD 4-digit data "0123" stored in (s)



- 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

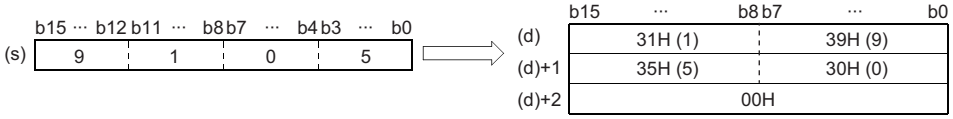
ASCII 10^0: Ones place of ASCII code

(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.

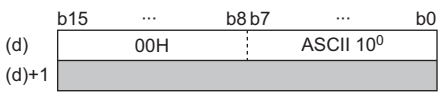
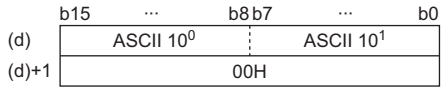
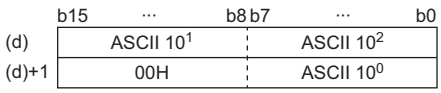
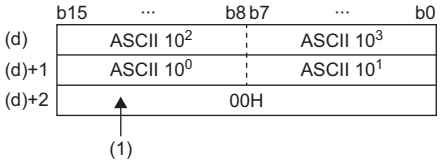
When 9105 is specified in (s)



■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.

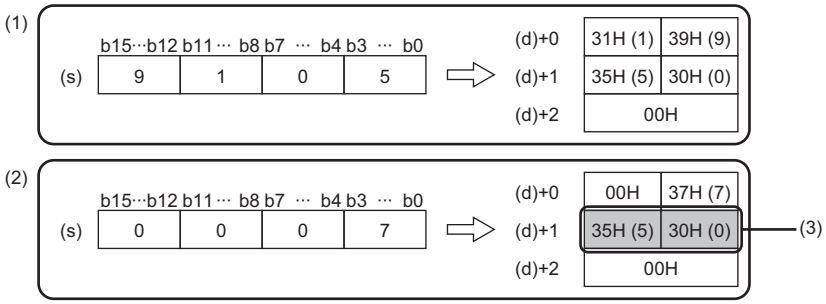
Operation error

Error code (SD0)	Description
3401H	Data in the device specified by (s) is out of the range, 0 to 9999.

Precautions

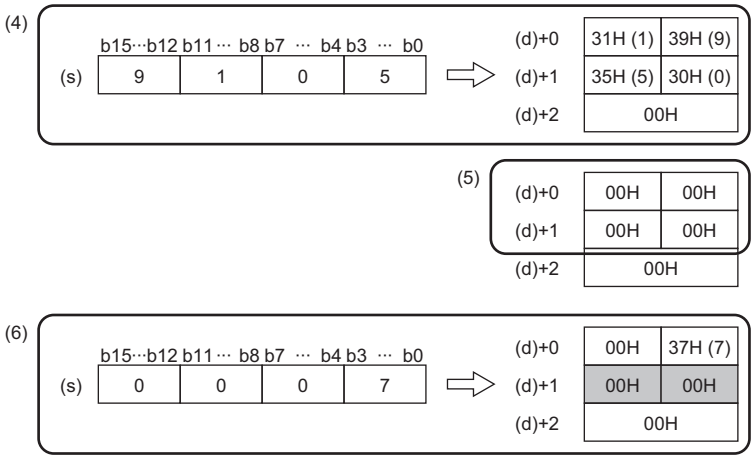
When SM705 (Number of conversion digits selection) is on, the operation result is stored in (d) for the effective number of digits. Therefore, when the BCDDA(P) instruction is executed successively and the operation result for each execution is stored in the same device, a part of the previous operation 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.



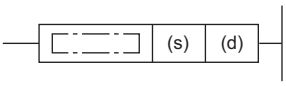
(4) "9105" is converted into a string.
(5) (d)+0 to (d)+1 are cleared.
(6) "7" is converted into a string.

Converting BCD 8-digit data to decimal ASCII code

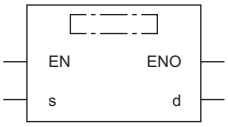
DBCDDA(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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*1	Storage format of (d)	Reference
OFF	Data is stored in a fixed number of digits (8 digits)	Page 825 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 826 Operation of when SM705 (Number of conversion digits selection) is on

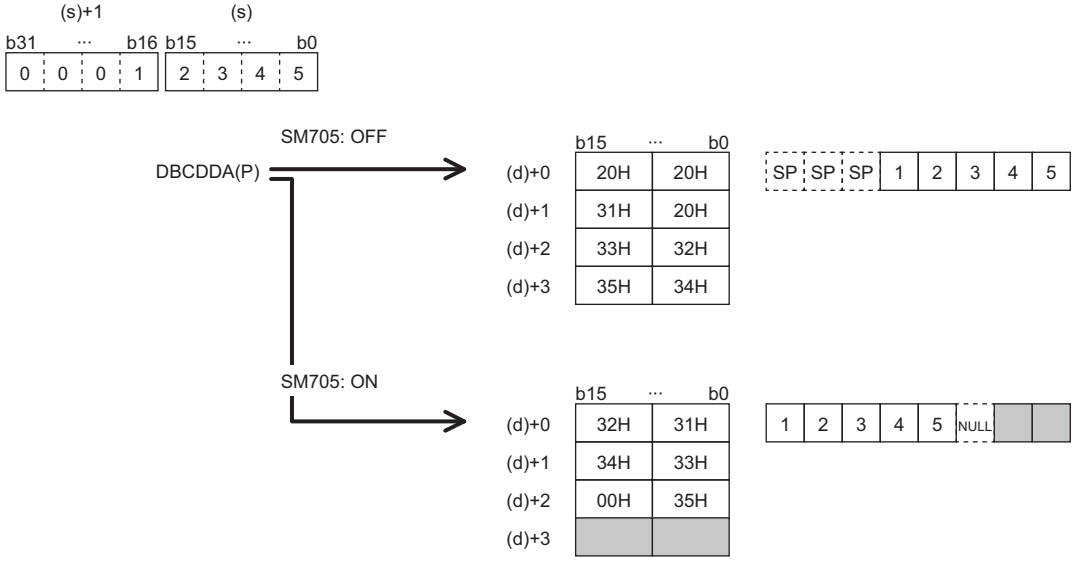
*1 For the firmware version of the CPU module supporting SM705, refer to the following.
MELSEC iQ-R CPU Module User's Manual (Application)
A CPU module which does not support SM705 operates in the same way as SM705 is off even if it is turned on.

■Operation overview

The following figure shows the operation when SM705 (Number of conversion digits selection) is off and on.

Ex.

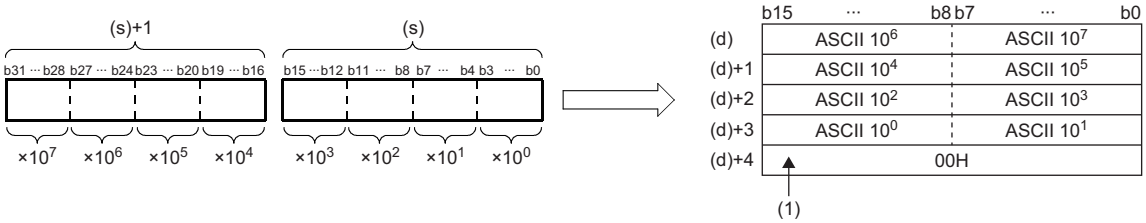
When the DBCDDA(P) instruction is executed with BCD 8-digit data "00012345" stored in (s)



- 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.



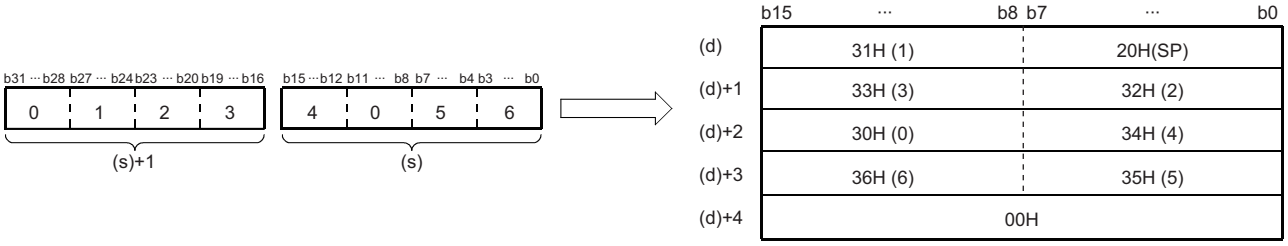
ASCII 10^7: Ten-millions place of ASCII code
ASCII 10^6: Millions place of ASCII code
⋮
ASCII 10^1: Tens place of ASCII code
ASCII 10^0: Ones place of ASCII code

(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).

Value of (s)	Data of (d) to (d)+3	Value of (s)	Data of (d) to (d)+3
• 0H to 9H	• The upper byte of (d) is filled with 00H. • (d)+1 and later remain unchanged. b15 ··· b8 b7 ··· b0 (d): 00H | ASCII 10^0 (d)+1: [ ] (d)+2: [ ] (d)+3: [ ]	• 10H to 99H	• (d)+1 is filled with 00H. • (d)+2 and later remain unchanged. b15 ··· b8 b7 ··· b0 (d): ASCII 10^0 | ASCII 10^1 (d)+1: 00H (d)+2: [ ] (d)+3: [ ]
⋮			
• 1000000H to 9999999H	• The upper byte of (d)+3 is filled with 00H. b15 ··· b8 b7 ··· b0 (d): ASCII 10^5 | ASCII 10^6 (d)+1: ASCII 10^3 | ASCII 10^4 (d)+2: ASCII 10^1 | ASCII 10^2 (d)+3: 00H | ASCII 10^0	• 10000000H to 99999999H	• (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. b15 ··· b8 b7 ··· b0 (d): ASCII 10^6 | ASCII 10^7 (d)+1: ASCII 10^4 | ASCII 10^5 (d)+2: ASCII 10^2 | ASCII 10^3 (d)+3: ASCII 10^0 | ASCII 10^1 (d)+4: 00H (1)

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.

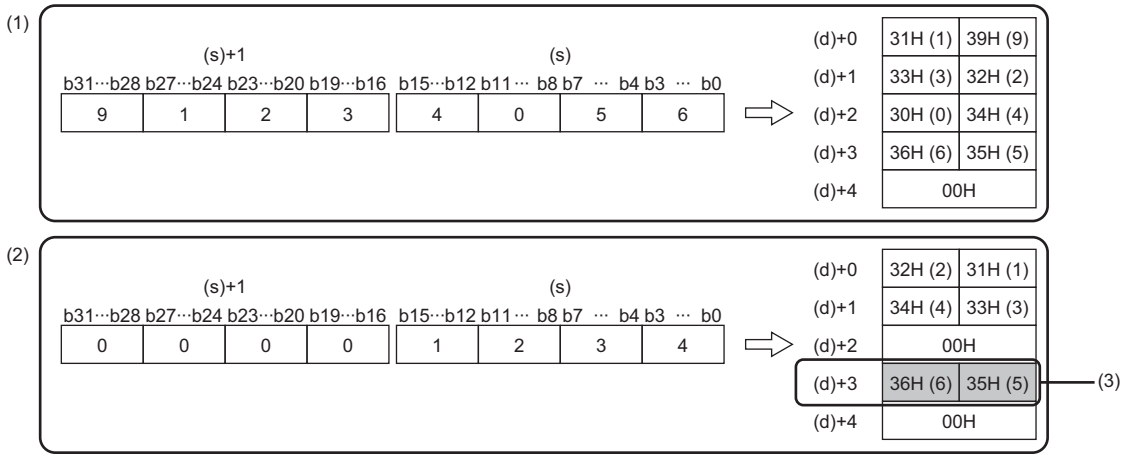
Operation error

Error code (SD0)	Description
3401H	Data in the device specified by (s) is out of the range, 0 to 99999999.

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"

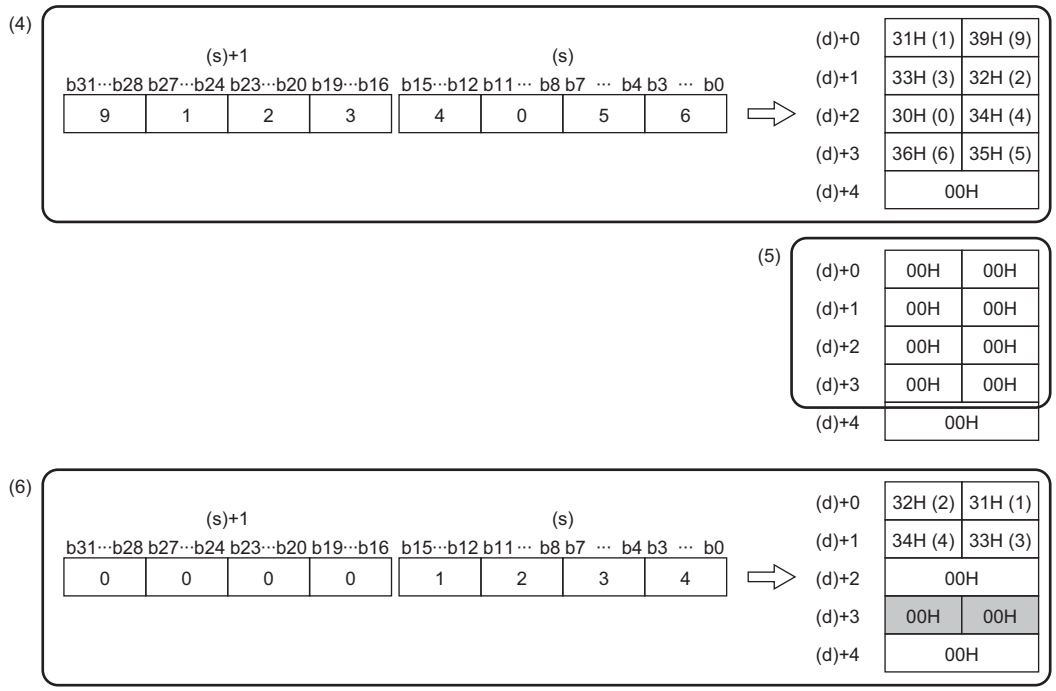


(1) "91234056" is converted into a string.

(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.

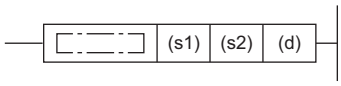
(6) "1234" is converted into a string.

Converting single-precision real number to string data

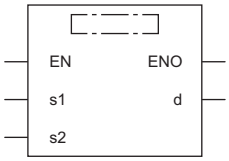
ESTR(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 character 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	○	○	○	○	○	—	○	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

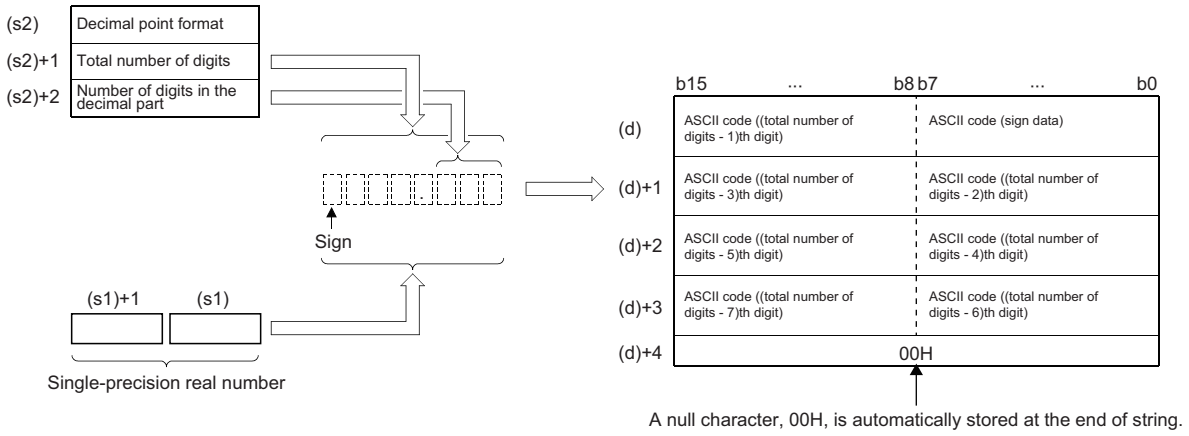
Processing details

- These instructions convert the single-precision real number data stored in the device specified by (s1) to a character 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

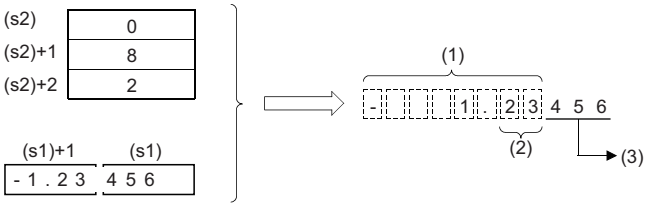
- When 0 is specified in (s2), the decimal point format is used.



- Total number of digits in the device specified by (s2)+1: When the number of digits in the decimal part is 0, total number of digits (maximum of 24)≥number of digits in the integral part[*1]+1. When it is a value other than 0, total number of digits (maximum of 24)≥(number of digits in the integral part[*1]+number of digits in the decimal part+2).

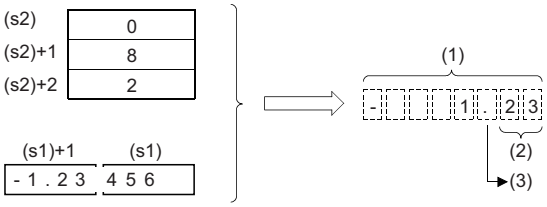
*1 Indicates the number of digits in the integral part of the 32-bit floating point real number data in the device specified by (s1).

- 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 smaller than the total number of digits minus 3.
- The converted character string data are stored in the device areas 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.



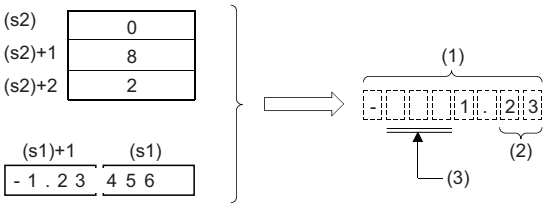
(1) Total number of digits
(2) Number of digits in the decimal part
(3) Rounded off.

- 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.



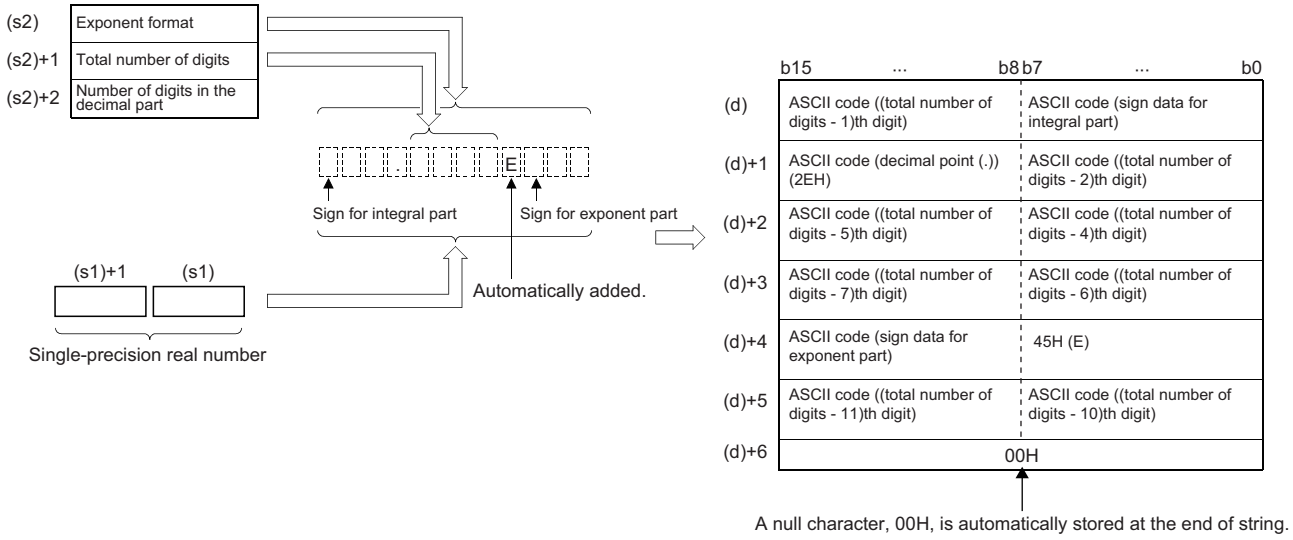
(1) Total number of digits
(2) Number of digits in the decimal part
(3) Filled with 20Hs (spaces).

- The value "00H" is automatically stored at the end of the converted character string.

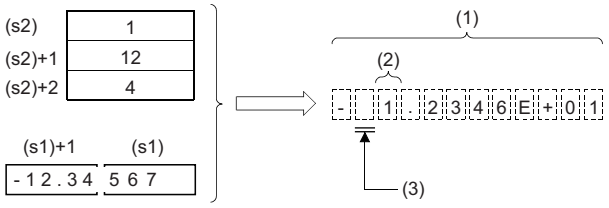
- The number of digits in the integral part of the 32-bit floating point real number data in the device specified by (s1) can be 1 to 16.

■Exponential format

- When 1 is specified in (s2), the exponential format is used.

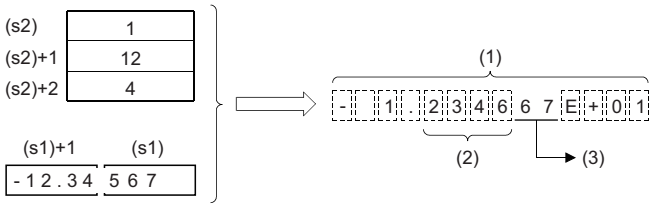


- Total number of digits in the device specified by (s2)+1: When the number of digits in the decimal part is 0, total number of digits (maximum of 24)≥2. When it is a value other than 0, total number of digits (maximum of 24)≥(number of digits in the decimal part+7).
- The number of digits in the decimal part that can be specified by (s2)+2 is 0 to 7. Note that the number of digits in the decimal part must be smaller than the total number of digits minus 7.
- The converted character string data are stored in the device areas 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.

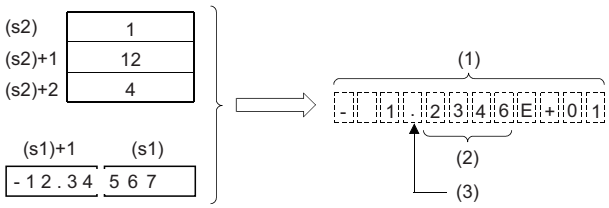


(1) Total number of digits

(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.

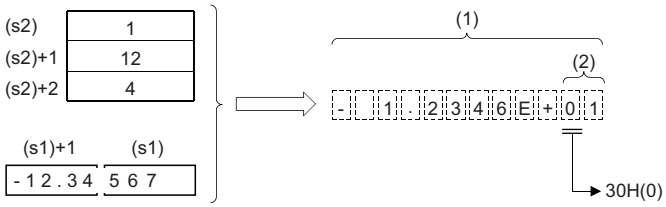


(1) Total number of digits

(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 character 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 48 Precautions

Operation error

Error code (SD0)	Description
3401H	The value in the device specified by (s1) is not in the following range. 0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$
	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 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.
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.

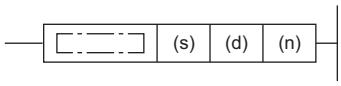
*1 Indicates the number of digits in the integral part of the 32-bit floating point real number data in the device specified by (s1).

Converting hexadecimal binary data to hexadecimal ASCII code

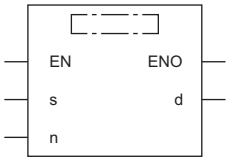
INT2ASC(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 exchanged to a character string	—	16-bit signed binary	ANY16*1
(d)	Start device for storing the converted character 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

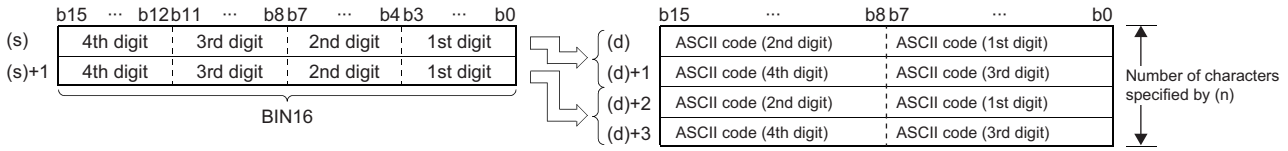
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■Applicable devices

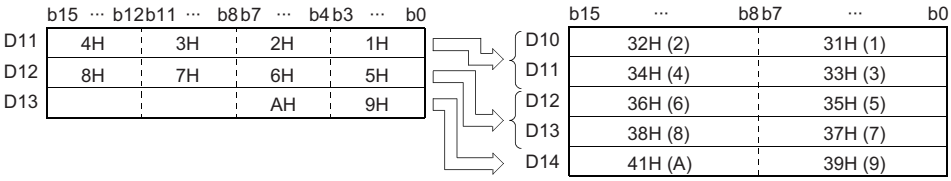
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

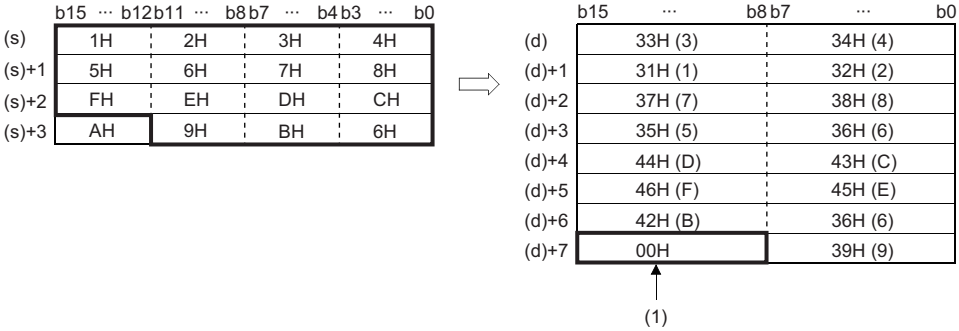
- Converts the 16-bit binary data in the device number specified by (s) and later to hexadecimal ASCII, and stores the converted data by the number of characters in the device specified by (n) in the device number specified by (d) and later.



- Setting the number of bytes 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 character 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 character 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

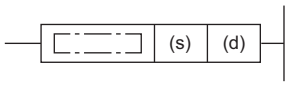
Error code (SD0)	Description
3405H	An out-of-range value is set to (n). • The specified number of characters is not between 0 and 16383.

Converting Unicode character string to Shift JIS character string

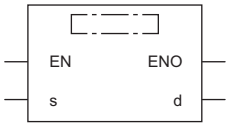
WS2SJIS(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert a Unicode character string to a Shift JIS character 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

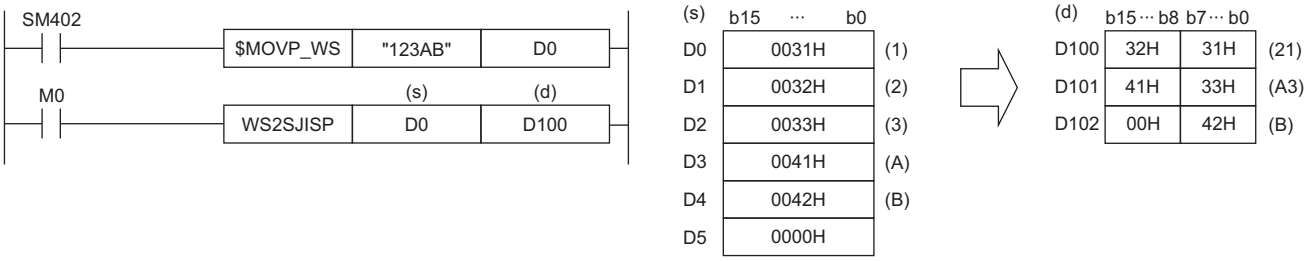
Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the character string to be converted	—	Unicode string	ANYSTRING_DOUBLE
(d)	Start device for storing the converted character string	—	String	ANYSTRING_SINGLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

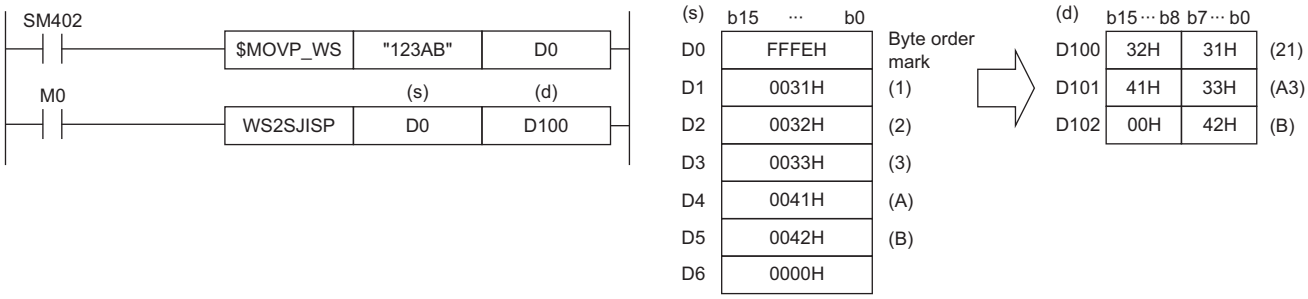
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instruction convert the Unicode character string in the device specified by (s) to the shift JIS character string, and stores the converted data in the device specified by (d).
- The Unicode character string in the device specified by (s) should be specified in little endian.
- 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.
3401H	Byte order mark FEFFH (big endian) is added to the character 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.
3405H	The character string in the device specified by (s) exceeds 16383 characters.*1

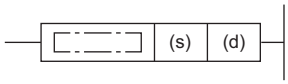
*1 A two-byte character such as a kanji character represented in shift JIS code should be counted 2.

Converting shift JIS character string to Unicode character string (without byte order mark)

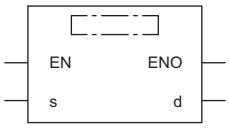
SJIS2WS(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert a Shift JIS character string to a Unicode character 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 character string to be converted	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the converted character string	—	Unicode string	ANYSTRING_DOUBLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

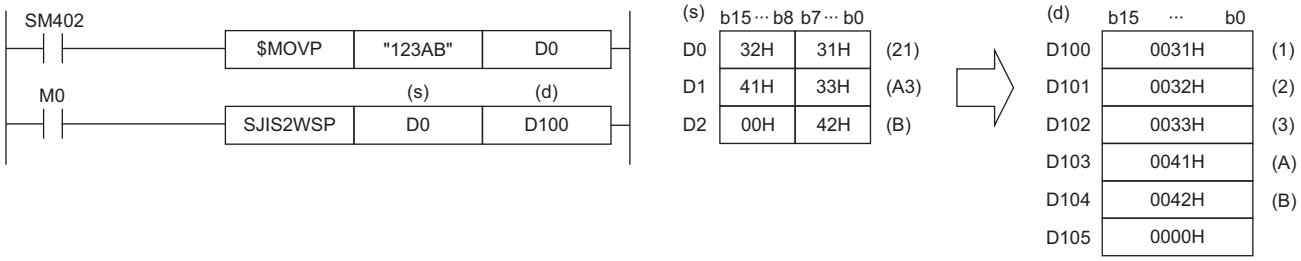
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the shift JIS character string in the device specified by (s) to a Unicode character string, and store the converted data in the device specified by (d).
- The shift JIS character string in the device specified by (s) should be specified in big endian.
- 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 838 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.
3401H	The range of data in the device specified by (s) includes a character code that cannot be converted.
3405H	The character string in the device specified by (s) exceeds 16383 characters.*1

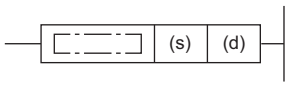
*1 A two-byte character such as a kanji character represented in shift JIS code should be counted 2.

Converting shift JIS character string to Unicode (with byte order mark)

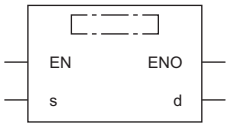
SJIS2WSB(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert a shift JIS character string to a Unicode character 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

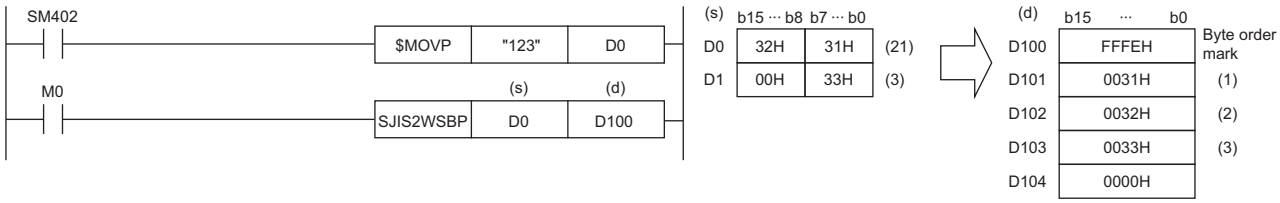
Operand	Description	Range	Data type	Data type (label)
(s)	Start device containing the character string to be converted	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the converted character string	—	Unicode string	ANYSTRING_DOUBLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions convert the shift JIS character string in the device specified by (s) to the Unicode character 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 character string in the device specified by (s) should be specified in big endian.
- 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.
3401H	The range of data in the device specified by (s) includes a character code that cannot be converted.
3405H	The character string in the device specified by (s) exceeds 16383 characters.*1

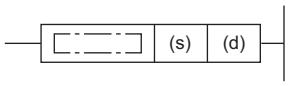
*1 A two-byte character such as a kanji character represented in shift JIS code should be counted 2.

Detecting a string length

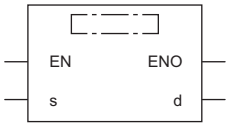
LEN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1494 LEN(_E)

■Execution condition

Instruction	Execution condition
LEN	
LENP	

Setting data

■Description, range, data type

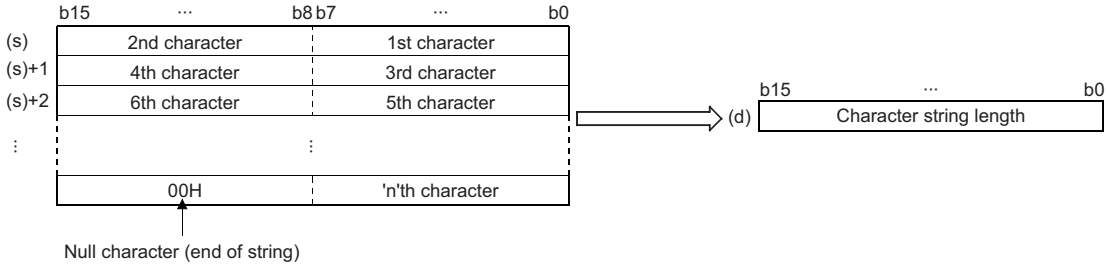
Operand	Description	Range	Data type	Data type (label)
(s)	Character string or the start device containing the character string	—	String	ANYSTRING_SINGLE
(d)	Number of the device for storing the length of the detected character string	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

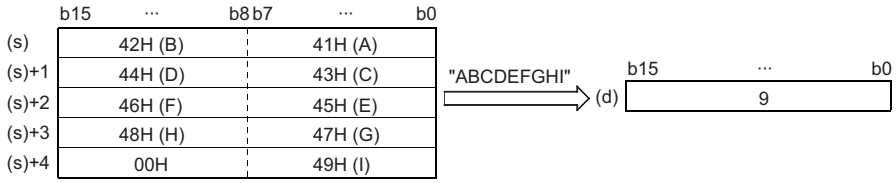
Processing details

- These instructions detect the length of the character 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 character string.



Ex.

When "ABCDEFGHI" is stored in the device specified by (s) and later



Operation error

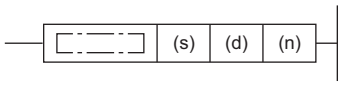
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s) and later.
3405H	The number of characters of the character string in the device specified by (s) exceeds 16383.

Extracting string data from the right

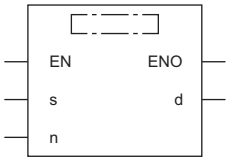
RIGHT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1496 LEFT(_E), RIGHT(_E)

■Execution condition

Instruction	Execution condition
RIGHT	
RIGHTP	

Setting data

■Descriptions, ranges, and data types

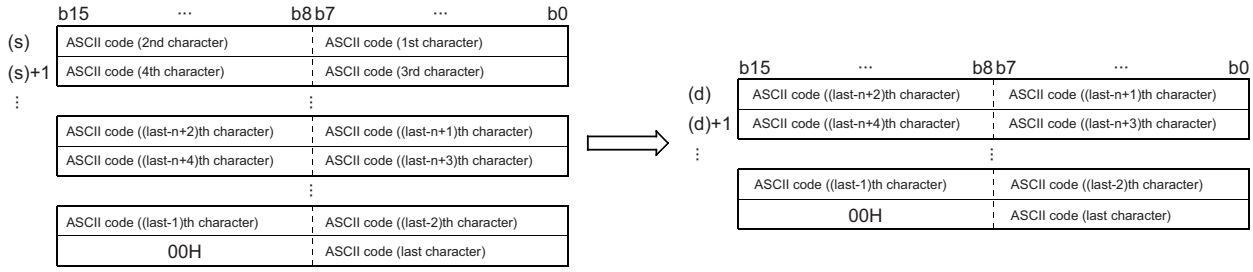
Operand	Description	Range	Data type	Data type (label)
(s)	Character string or the start device containing the character string	—	String	ANYSTRING_SINGLE
(d)	Start device for storing (n) characters of character string extracted from the right of the data in the device specified by (s)	—	String	ANYSTRING_SINGLE
(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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

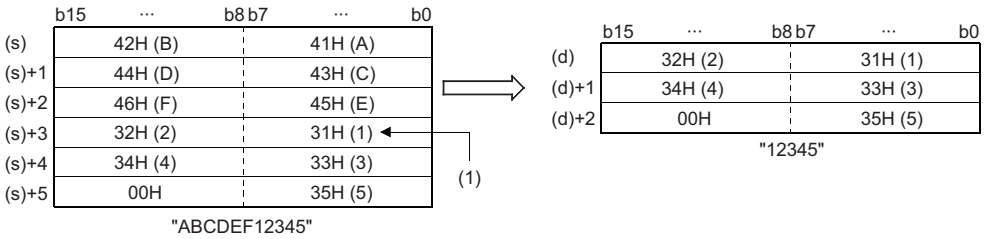
Processing details

- These instructions extract (n) characters of data from the right of the character string data (the end of the character 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 character string is automatically added to the end of the character 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

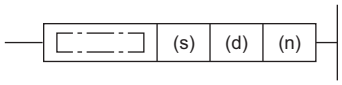
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s) and later.
3405H	An out-of-range value is set to (s). • The number of characters of the character string in the device specified by (s) exceeds 16383. • The number of characters of the character 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).

Extracting string data from the left

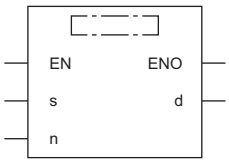
LEFT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1496 LEFT(_E), RIGHT(_E)

■Execution condition

Instruction	Execution condition
LEFT	
LEFTP	

Setting data

■Descriptions, ranges, and data types

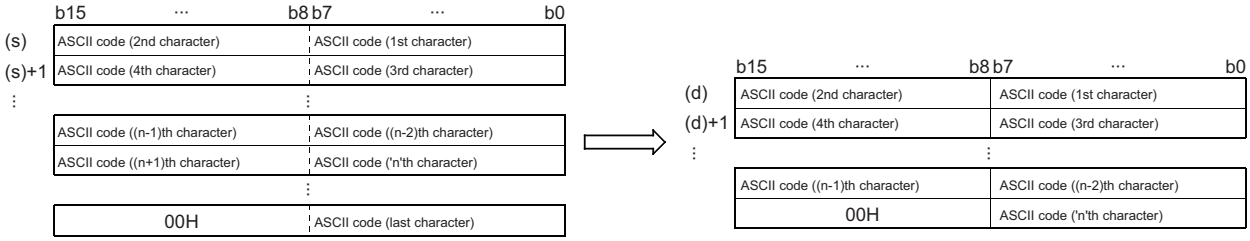
Operand	Description	Range	Data type	Data type (label)
(s)	Character string or the start device containing the character string	—	String	ANYSTRING_SINGLE
(d)	Start device for storing (n) characters of character string extracted from the left of the data in the device specified by (s)	—	String	ANYSTRING_SINGLE
(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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

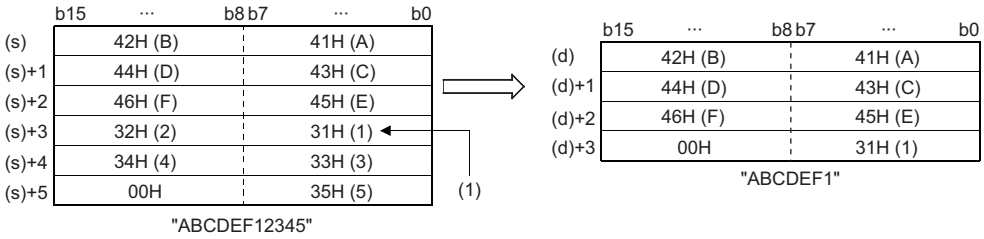
Processing details

- These instructions extract (n) characters of data from the left of the character string data (the start of the character 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 character string is automatically added to the end of the character 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

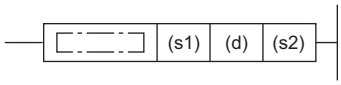
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s) and later.
3405H	An out-of-range value is set to (s). • The number of characters of the character string in the device specified by (s) exceeds 16383. • The number of characters of the character 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).

Extracting the specified string data

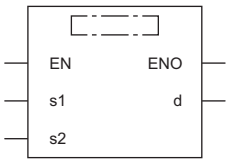
MIDR(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

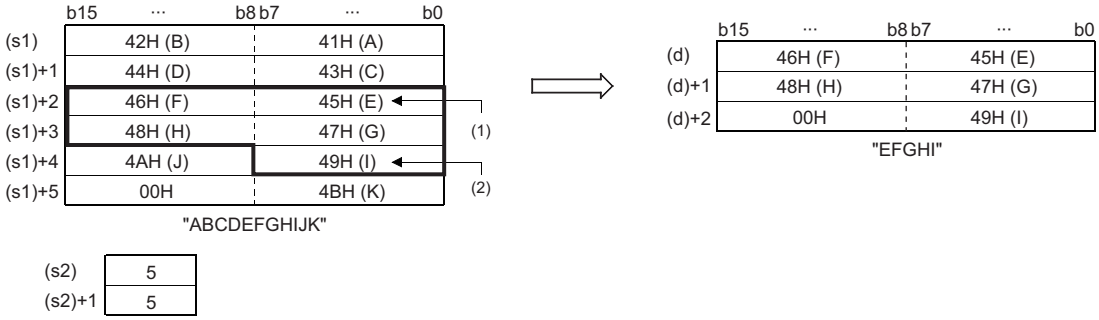
Operand	Description	Range	Data type	Data type (label)
(s1)	Character string or the start device containing the character string	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the character string data of the operation result	—	String	ANYSTRING_SINGLE
(s2)	Start device for storing the location 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

- These instructions extract the data by the number of characters specify by (s2)+1 from the location specified by (s2) in the character 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.

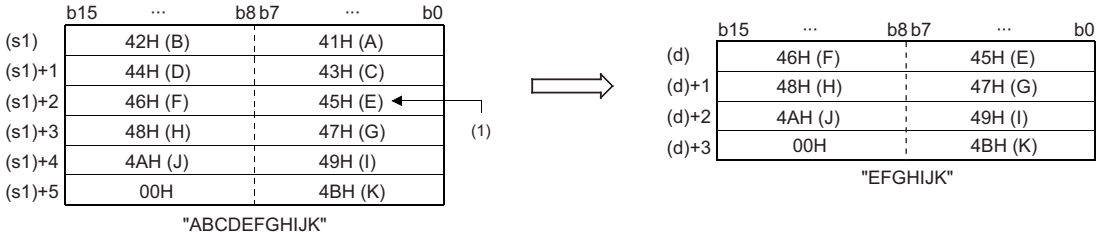


(s2)	5
(s2)+1	5

(1) Position specified by (s2): 5th character

(2) ASCII code ((s2)+1)th character from the specified position)

- The NULL code (00H) indicating the end of a character string is automatically added to the end of the character 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.



(s2)	5
(s2)+1	-1

(1) Position specified by (s2): 5th character

Operation error

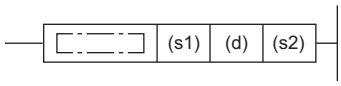
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s1) and later.
3405H	The number of characters of the character string in the device specified by (s1) exceeds 16383.
	An out-of-range value is set to (s2). • The 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 those in the devices specify by (s2) and (s2)+1 exceeds the number of characters in the device specified by (s1).

Replacing the specified string data

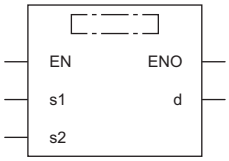
MIDW(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

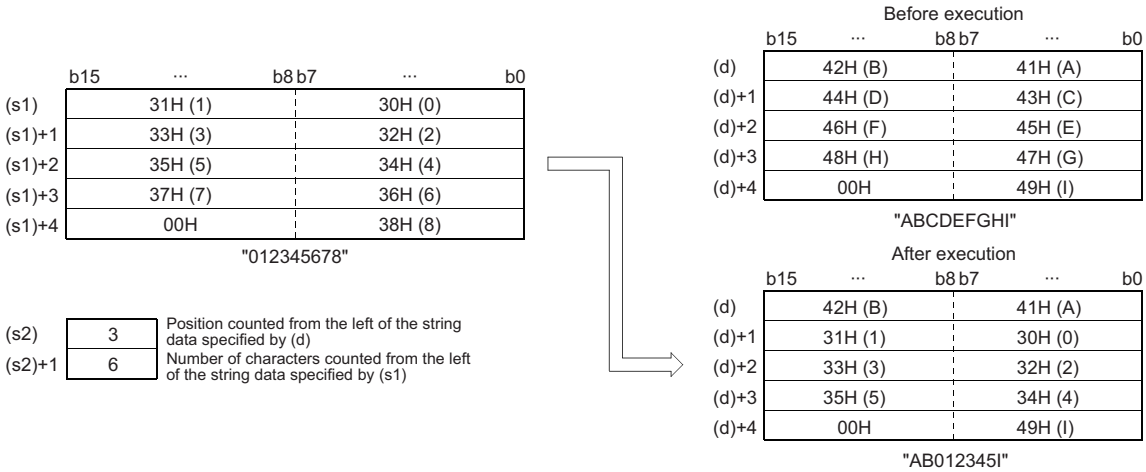
Operand	Description	Range	Data type	Data type (label)
(s1)	Character string or the start device containing the character string	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the character string data of the operation result	—	String	ANYSTRING_SINGLE
(s2)	Start device for storing the location 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

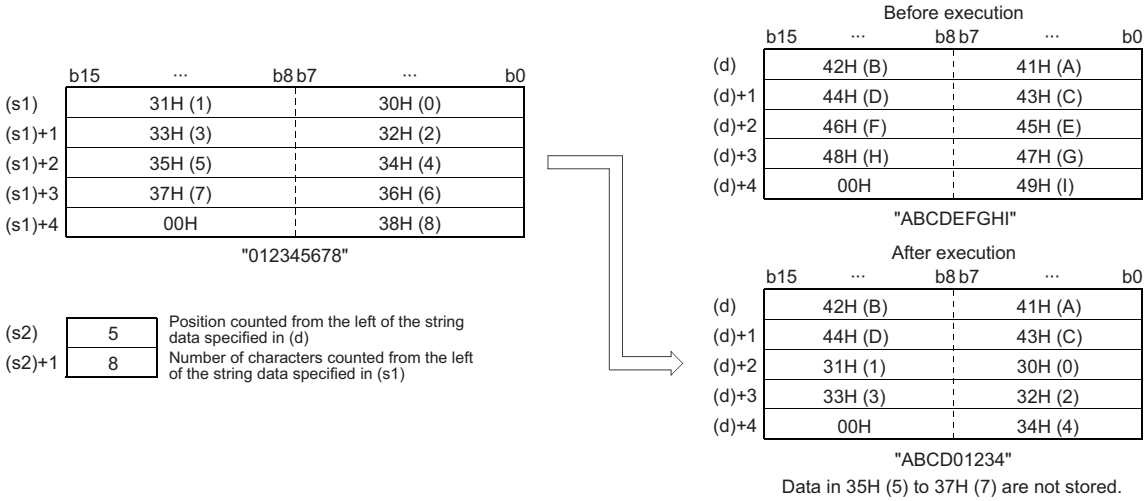
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	—	—	—	—

Processing details

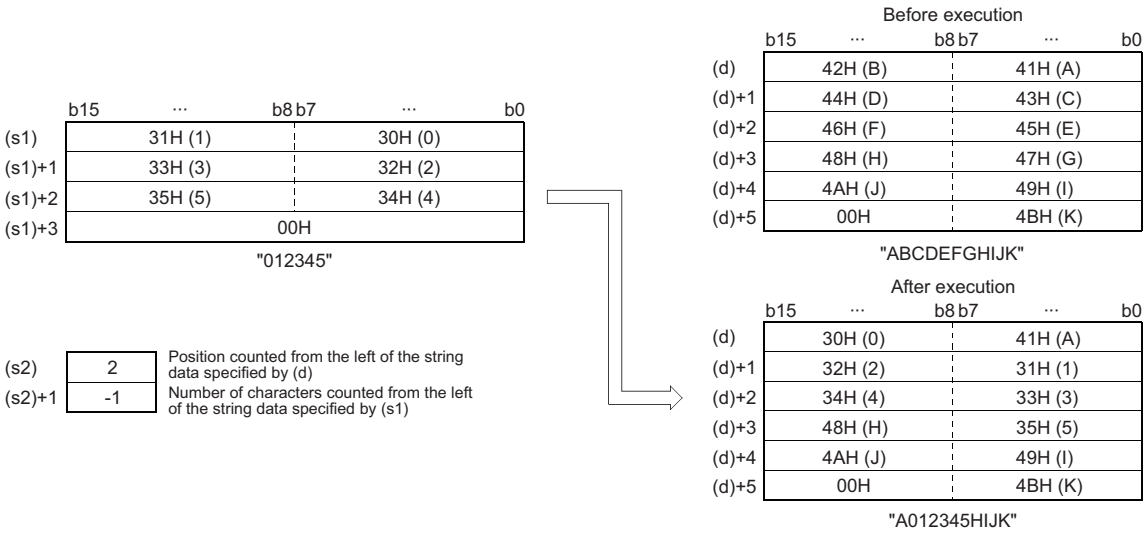
- These instructions read data by the number of characters stored in the device specified by (s2)+1 from the character string data stored in the device number specified by (s1) and later, and store the read data at the location in the device specified by (s2) and later in the character string stored in the device number specified by (d) and later.



- The NULL code (00H) indicating the end of a character string is automatically added to the end of the character 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 character 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

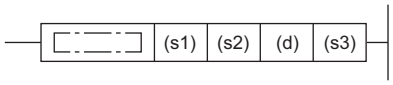
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s1) and later.
	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (d) and later.
3405H	The number of characters of the character string in the device specified by (s1) exceeds 16383.
	The number of characters of the character 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

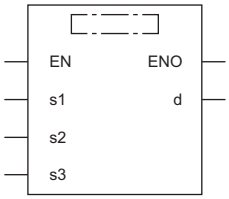
INSTR(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

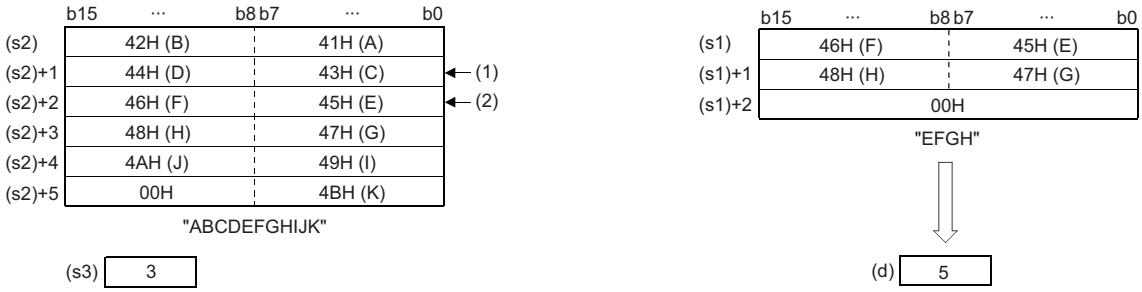
Operand	Description	Range	Data type	Data type (label)
(s1)	Character strings to be searched or the start device containing these character strings	—	String	ANYSTRING_SINGLE
(s2)	Character string to be searched for or the start device containing the character string to be searched for	—	String	ANYSTRING_SINGLE
(d)	Device for storing the search result	—	16-bit signed binary	ANY16
(s3)	Search start position	1 to 16383	16-bit 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—
(s3)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions search the character string data stored in the device number specified by (s2) and later starting from the (s3)th character from the left for the character 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 character 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 character string data is matching, 0 is stored in the device specified by (d).

Operation error

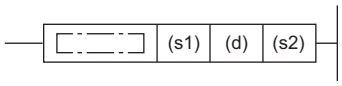
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s1) and later.
	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s2) and later.
3405H	An out-of-range value is set to (s1). • The number of characters of the character string that has been set is 0. • The number of characters of the character string that has been set exceeds 16383.
	The number of characters of the character 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

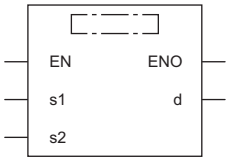
STRINS(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

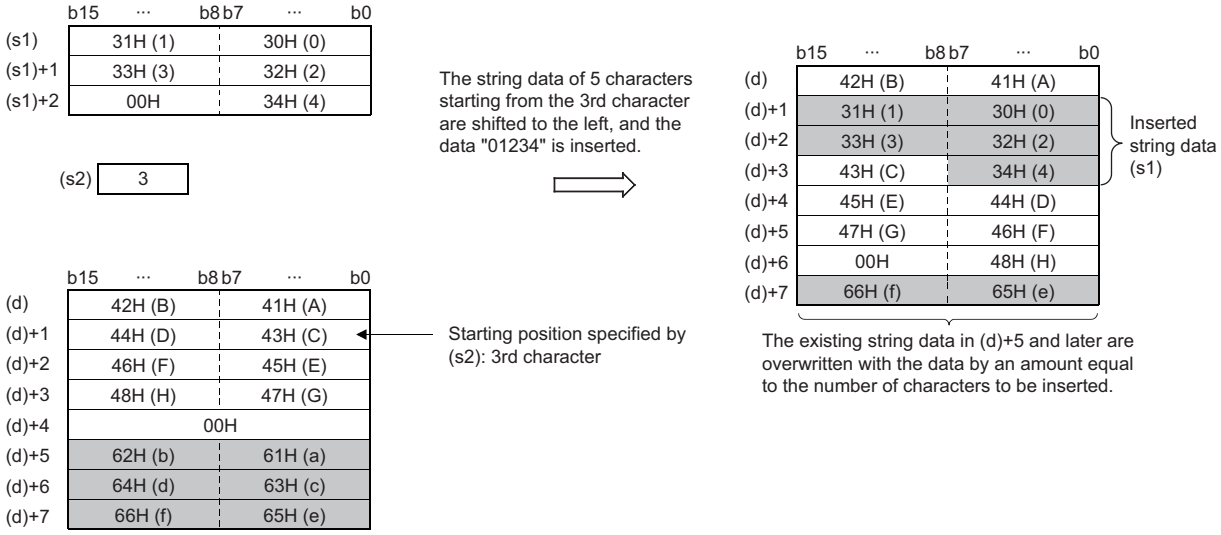
Operand	Description	Range	Data type	Data type (label)
(s1)	Character string to be inserted or the start device containing the character string to be inserted	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the insertion result character string	—	String	ANYSTRING_SINGLE
(s2)	Insertion position (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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- These instructions insert the character string data in the device specified by (s1) to the (s2)th character (insertion position) from the start of the character string data in the device specified by (d).



- If the character 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 character string.
- If the character 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 character string.
- When the number of characters in the device specified by (d) plus 1 is specified in (s2), the character string in the device specified by (s1) is concatenated to the end of the character string in the device specified by (d).

Operation error

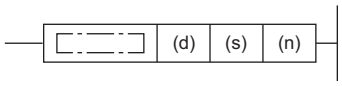
Error code (SD0)	Description
2820H	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (s1) and later.
	There is no NULL code (00H) in each setting area in the device/label memory in the device specified by (d) and later.
2821H	The devices specified by (s1) and (d), both containing character strings, are overlapping even partly.
	The device specified by (s1)+(d) containing the character string after insertion overlaps with the character string storage device specified by (s1).
3405H	The number of characters of the character 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 character 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 character string in the device specified by (d) exceeds 16383.
3406H	The character string after insertion stored in the device specified by (s1)+(d) becomes data outside the output enable range. • The number of characters of the character string after insertion in the device specified by (d) exceeds 16383. • The character string after insertion exceeds each setting area in the specified device/label memory.

Deleting string data

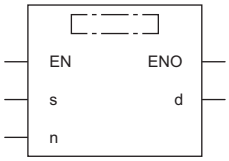
STRDEL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

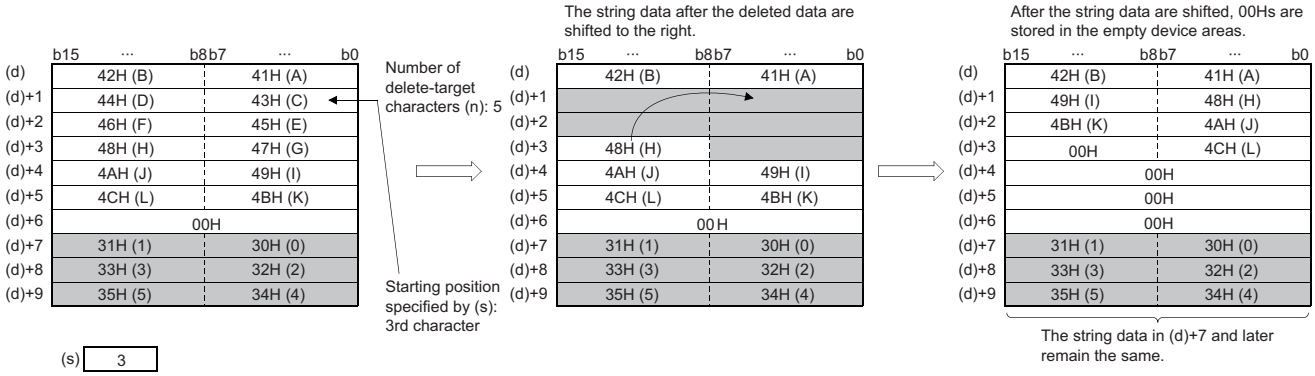
Operand	Description	Range	Data type	Data type (label)
(d)	Start device containing the character string to be deleted	—	String	ANYSTRING_SINGLE
(s)	Deletion start position	1 to 16383	16-bit signed binary	ANY16
(n)	Number of characters to be deleted	0 to 16384-(s)	16-bit signed binary	ANY16
EN	Execution 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□\□, U3E□\(H)G□	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 character string data in the device specified by (d).



- If the character 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 character string.
- If the character 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 character string.
- The character 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 each setting area in the device/label memory in the device specified by (d) and later.
3405H	The number of characters of the character 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 character 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 character string in the device specified by (d) to the last character. • The specified value is negative.

11 REAL VALUE PROCESSING

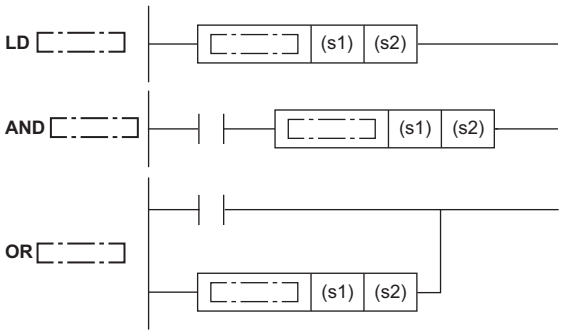
11.1 Floating-point instruction

Comparing single-precision real numbers

LDE□, ANDE□, ORE□

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions perform a comparison operation of a single-precision real number. (Devices are used as a normally open contact.)

Ladder	ST*1
 (□ 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.)*2

FBD/LD

(□ is replaced by a combination of LDE_, ANDE_, or ORE_ and EQ, NE, GT, LE, LT, or GE.)*2

*1 The engineering tool with version "1.035M" or later supports the ST.
*2 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□\□, U3E□\(H)G□	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.

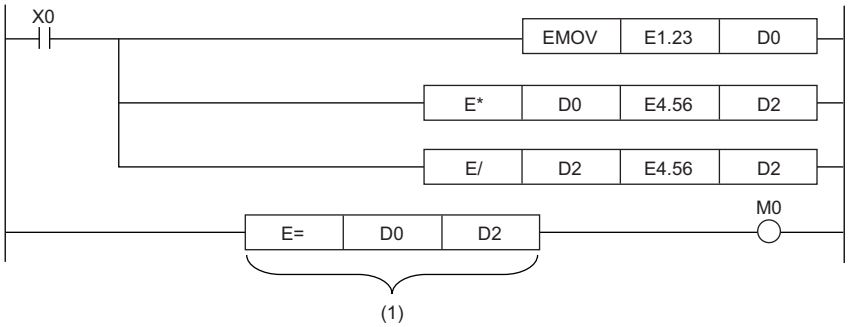
☞ Page 48 Precautions

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.



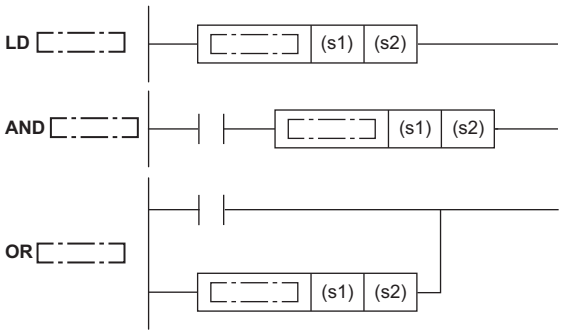
(1) Two values may not be equal.

Comparing double-precision real numbers

LDED□, ANDED□, ORED□

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions perform a comparison operation of a double-precision real number. (Devices are used as a normally open contact.)

Ladder	ST*1
 (□ is replaced by ED=, ED<>, ED>, ED<=, ED<, ED>=.)	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.)*2

FBD/LD

(□ is replaced by a combination of LDED_, ANDED_, or ORED_ and EQ, NE, GT, LE, LT, or GE.)*2

*1 The engineering tool with version "1.035M" or later supports the ST.
*2 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, $2^{-1022} \leq |(s1)| < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Comparison data or the start device where the comparison data is stored	0, $2^{-1022} \leq |(s2)| < 2^{1024}$	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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

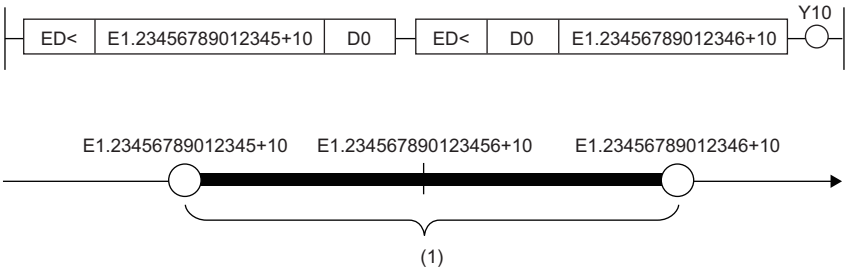
There is no operation error.

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.

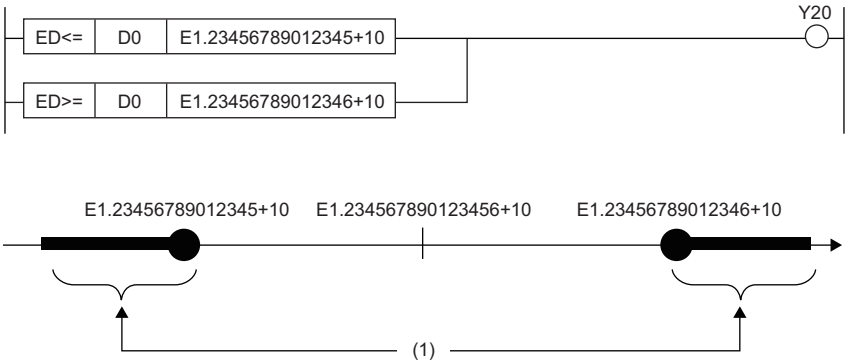
To determine the match between E1.234567890123456+10 (16 effective digits) and a double-precision real number



(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.)

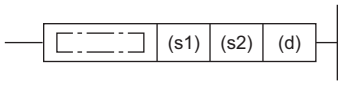
Outputting a comparison result of single-precision real numbers

ECMP(P)

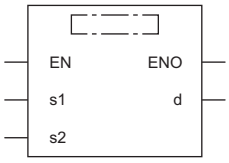
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions compare the single-precision real number data specified by (s1) with the single-precision real number data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST
	ENO:=ECMP(EN,s1,s2,d); ENO:=ECMPP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ECMP	
ECMPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Comparison data or the start device for storing the comparison data	0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Comparison data or the start device for storing the comparison data	0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(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

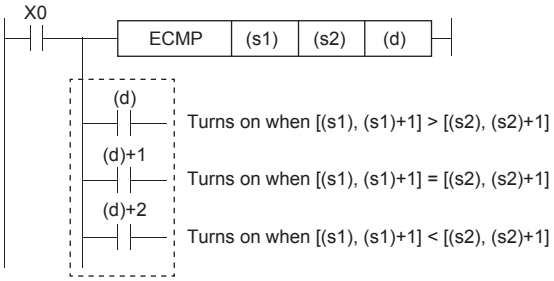
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	○	—	○	○	○	—	○	—	—
(s2)	—	—	○	○	—	○	○	○	—	○	—	—
(d)	○	—	○*1	—	—	—	—	○	—	—	—	—

*1 T, ST, and C cannot be used.

Processing details

- These instructions compare the single-precision real number data specified by (s1) with the single-precision real number data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.



Operation error

Error code (SD0)	Description
3402H	The value input to (s1) and (s2) is -0, a subnormal number, NaN (not a number), or ±∞.

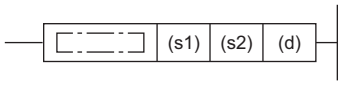
Outputting a comparison result of double-precision real numbers

EDCMP(P)

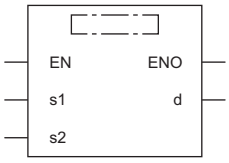
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions compare the double-precision real number data specified by (s1) with the double-precision real number data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST
	ENO:=EDCMP(EN,s1,s2,d); ENO:=EDCMPP(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
EDCMP	
EDCMPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Comparison data or the start device for storing the comparison data	0, $2^{-1022} \leq |(s1)| < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Comparison data or the start device for storing the comparison data	0, $2^{-1022} \leq |(s2)| < 2^{1024}$	Double-precision real number	ANYREAL_64
(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

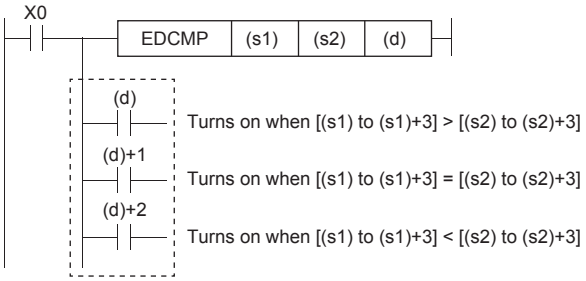
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○*1	—	—	—	—	○	—	—	—	—

*1 T, ST, and C cannot be used.

Processing details

- These instructions compare the double-precision real number data specified by (s1) with the double-precision real number data specified by (s2), and according to the result (small, equal, or large), (d), (d)+1, or (d)+2 is turned on.



Operation error

Error code (SD0)	Description
3402H	The value input to (s1) and (s2) is -0, a subnormal number, NaN (not a number), or ±∞.

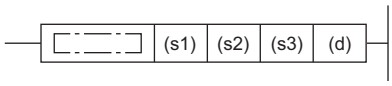
Outputting a band comparison result of single-precision real number

EZCP(P)

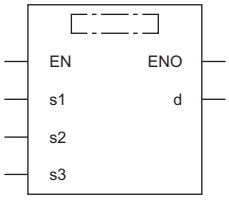
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions compare the band between the single-precision real number specified by lower limit value (s1) and the single-precision real number specified by upper limit value (s2) with the single-precision real number in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST
	ENO:= EZCP(EN, s1, s2, s3, d); ENO:= EZCPP(EN, s1, s2, s3, d);

FBD/LD



■Execution condition

Instruction	Execution condition
EZCP	
EZCPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Lower limit value or the start device for storing the lower limit value	0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Upper limit value or the start device for storing the upper limit value	0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s3)	Comparison data or the start device for storing the comparison data	0, $2^{-126} \leq \lvert(s3)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(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

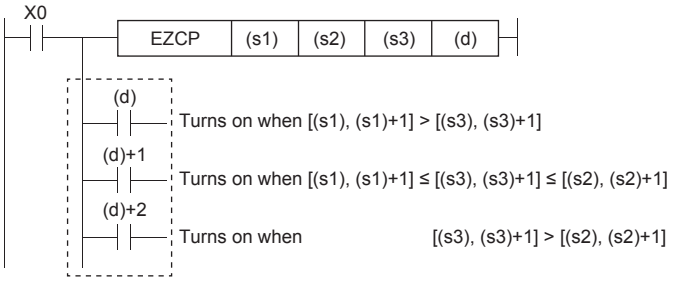
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	○	—	○	○	○	—	○	—	—
(s2)	—	—	○	○	—	○	○	○	—	○	—	—
(s3)	—	—	○	○	—	○	○	○	—	○	—	—
(d)	○	—	○*1	—	—	—	—	○	—	—	—	—

*1 T, ST, and C cannot be used.

11

Processing details

- These instructions compare the band between the single-precision real number specified by lower limit value (s1) and the single-precision real number specified by upper limit value (s2) with the single-precision real number in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.



Precautions

- Set (s1) to a value less than (s2). If (s1) is set to a value greater than (s2), (s2) is treated as the same value as (s1).

Operation error

Error code (SD0)	Description
3402H	The value input to (s1) to (s3) is -0, a subnormal number, NaN (not a number), or ±∞.

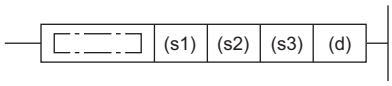
Outputting a band comparison result of double-precision real number

EDZCP(P)

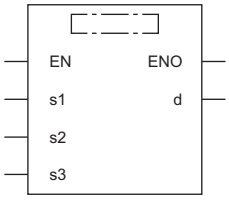
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions compare the band between the double-precision real number specified by lower limit value (s1) and the double-precision real number specified by upper limit value (s2) with the double-precision real number in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST
	ENO:= EDZCP(EN, s1, s2, s3, d); ENO:= EDZCPP(EN, s1, s2, s3, d);

FBD/LD



■Execution condition

Instruction	Execution condition
EDZCP	
EDZCPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Lower limit value or the start device for storing the lower limit value	0, $2^{-1022} \leq \lvert(s1)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Upper limit value or the start device for storing the upper limit value	0, $2^{-1022} \leq \lvert(s2)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(s3)	Comparison data or the start device for storing the comparison data	0, $2^{-1022} \leq \lvert(s3)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(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

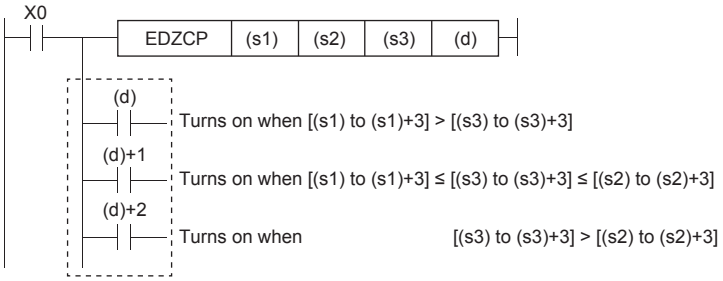
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	○	—	○	—	○	—	—
(s2)	—	—	○	—	—	○	—	○	—	○	—	—
(s3)	—	—	○	—	—	○	—	○	—	○	—	—
(d)	○	—	○*1	—	—	—	—	○	—	—	—	—

*1 T, ST, and C cannot be used.

Processing details

- These instructions compare the band between the double-precision real number specified by lower limit value (s1) and the double-precision real number specified by upper limit value (s2) with the double-precision real number in the device specified by comparison data (s3). According to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.



Precautions

- Set (s1) to a value less than (s2). If (s1) is set to a value greater than (s2), (s2) is treated as the same value as (s1).

Operation error

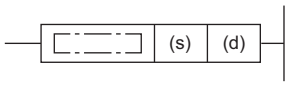
Error code (SD0)	Description
3402H	The value input to (s1) to (s3) is -0, a subnormal number, NaN (not a number), or ±∞.

Adding single-precision real numbers

E+(P) [when two operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions add single-precision real numbers.

Ladder	ST
	Not supported (☞ Page 872 E+(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 872 E+(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
E+	
E+P	

Setting data

■Description, range, data type

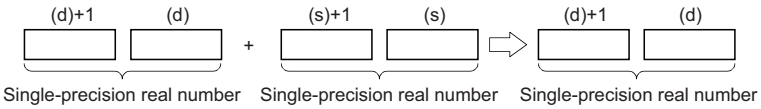
Operand	Description	Range	Data type	Data type (label)
(s)	Second addend data or the start device where the second addend data is stored	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device where the first addend data is stored	0, $2^{-126} \leq \lvert(d)\rvert < 2^{128}$	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□\□, U3E□\(H)G□	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 48 Precautions

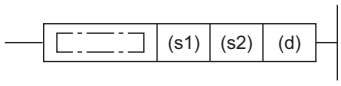
Operation error

Error code (SD0)	Description
3402H	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 ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

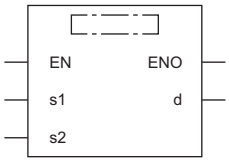
E+(P) [when three operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

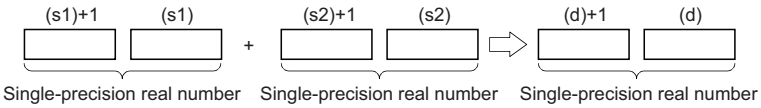
Operand	Description	Range	Data type	Data type (label)
(s1)	First addend data or the start device where the first addend data is stored	0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Second addend data or the start device where the second addend data is stored	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☐\☐, U3E☐\(H)G☐	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

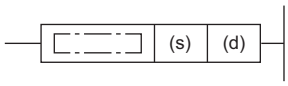
Error code (SD0)	Description
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

Subtracting single-precision real numbers

E-(P) [when two operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions perform subtraction between single-precision real numbers.

Ladder	ST
	Not supported (☞ Page 876 E-(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 876 E-(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
E-	
E-P	

Setting data

■Description, range, data type

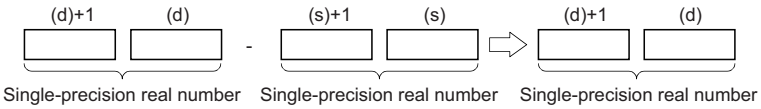
Operand	Description	Range	Data type	Data type (label)
(s)	Subtrahend data or the start device where subtrahend data is stored	0, $2^{-126} \leq |(s)| < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device where minuend data is stored	0, $2^{-126} \leq |(d)| < 2^{128}$	Single-precision real 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□\□, U3E□\(H)G□	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 48 Precautions

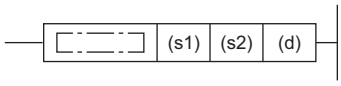
Operation error

Error code (SD0)	Description
3402H	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 ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

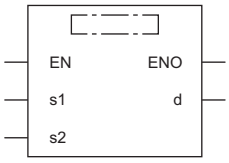
E-(P) [when three operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

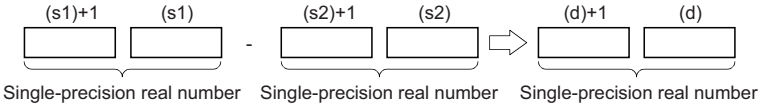
Operand	Description	Range	Data type	Data type (label)
(s1)	Minuend data or the start device where minuend data is stored	0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Subtrahend data or the start device where subtrahend data is stored	0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$	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☐\☐, U3E☐\(H)G☐	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 48 Precautions

Operation error

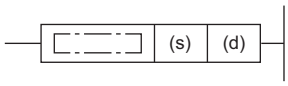
Error code (SD0)	Description
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert < 2^{128}$

Adding double-precision real numbers

ED+(P) [when two operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions add double-precision real numbers.

Ladder	ST
	Not supported (☞ Page 880 ED+(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 880 ED+(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
ED+	
ED+P	

Setting data

■Description, range, data type

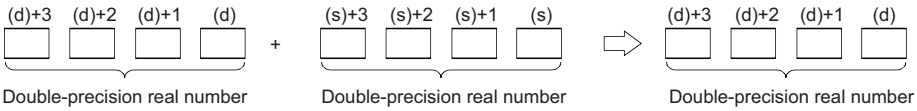
Operand	Description	Range	Data type	Data type (label)
(s)	Second addend data or the start device where the second addend data is stored	0, $2^{-1022} \leq |(s)| < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device where the first addend data is stored	0, $2^{-1022} \leq |(d)| < 2^{1024}$	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□\□, U3E□\(H)G□	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 48 Precautions

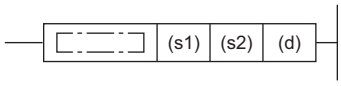
Operation error

Error code (SD0)	Description
3402H	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 ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

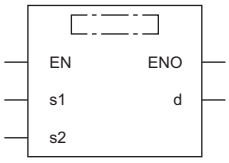
ED+(P) [when three operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

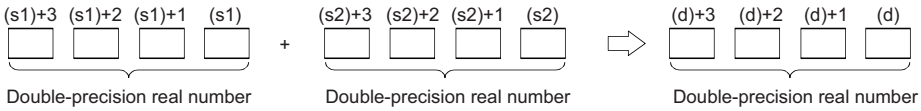
Operand	Description	Range	Data type	Data type (label)
(s1)	First addend data or the start device where the first addend data is stored	0, $2^{-1022} \leq \lvert(s1)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Second addend data or the start device where the second addend data is stored	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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

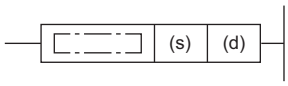
Error code (SD0)	Description
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert < 2^{1024}$

Subtracting double-precision real numbers

ED-(P) [when two operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions perform subtraction between double-precision real numbers.

Ladder	ST
	Not supported (☞ Page 884 ED-(P) [when three operands are set])

FBD/LD
Not supported (☞ Page 884 ED-(P) [when three operands are set])

■Execution condition

Instruction	Execution condition
ED-	
ED-P	

Setting data

■Description, range, data type

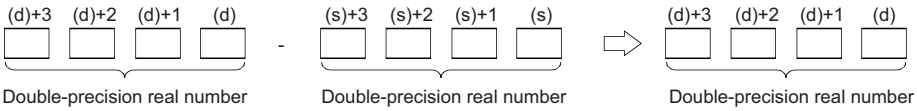
Operand	Description	Range	Data type	Data type (label)
(s)	Subtrahend data or the start device where subtrahend data is stored	0, $2^{-1022} \leq |(s)| < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device where minuend data is stored	0, $2^{-1022} \leq |(d)| < 2^{1024}$	Double-precision 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□\□, U3E□\(H)G□	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 48 Precautions

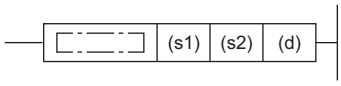
Operation error

Error code (SD0)	Description
3402H	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 ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

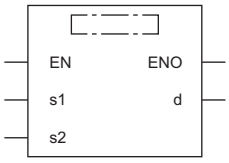
ED-(P) [when three operands are set]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

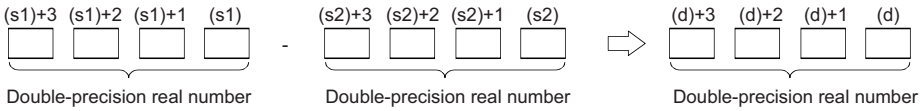
Operand	Description	Range	Data type	Data type (label)
(s1)	Minuend data or the start device where minuend data is stored	0, $2^{-1022} \leq \lvert(s1)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Subtrahend data or the start device where subtrahend data is stored	0, $2^{-1022} \leq \lvert(s2)\rvert < 2^{1024}$	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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

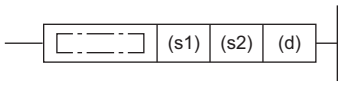
Error code (SD0)	Description
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert < 2^{1024}$

Multiplying single-precision real numbers

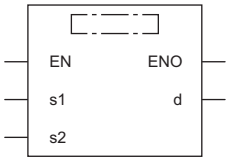
E*(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions multiply single-precision real numbers.

Ladder	ST*1
	ENO:=EMULTI(EN,s1,s2,d); ENO:=EMULTIP(EN,s1,s2,d);

FBD/LD



(☐ is replaced by either of the following: EMULTI, EMULTIP.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
E*	
E*P	

Setting data

■Description, range, data type

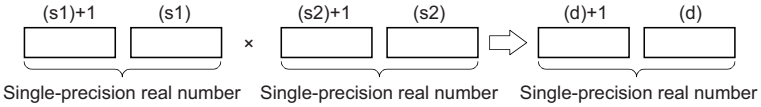
Operand	Description	Range	Data type	Data type (label)
(s1)	Multiplicand data or the start device where multiplicand data is stored	0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Multiplier data or the start device where multiplier data is stored	0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$	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☐\☐, U3E☐\(H)G☐	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 48 Precautions

Operation error

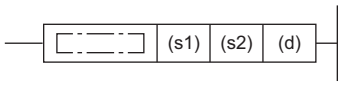
Error code (SD0)	Description
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert < 2^{128}$

Dividing single-precision real numbers

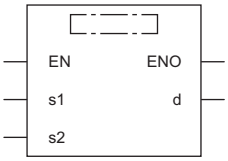
E/(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions perform division between single-precision real numbers.

Ladder	ST*1
	ENO:=EDIVISION(EN,s1,s2,d); ENO:=EDIVISIONP(EN,s1,s2,d);

FBD/LD



(☐ is replaced by either of the following: EDIVISION, EDIVISIONP.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
E/	
E/P	

Setting data

■Description, range, data type

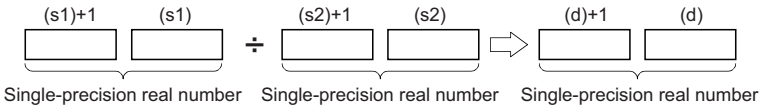
Operand	Description	Range	Data type	Data type (label)
(s1)	Dividend data or the start device where dividend data is stored	0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Divisor data or the start device where divisor data is stored	0, $2^{-126} \leq \lvert(s2)\rvert < 2^{128}$	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☐\☐, U3E☐\(H)G☐	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 48 Precautions

Operation error

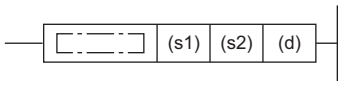
Error code (SD0)	Description
3400H	The data (divisor) in the device specified by (s2) is 0.
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data in the device specified by (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert < 2^{128}$

Multiplying double-precision real numbers

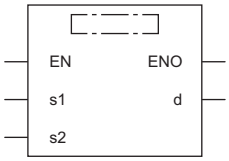
ED*(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions multiply double-precision real numbers.

Ladder	ST*1
	ENO:=EDMULTI(EN,s1,s2,d); ENO:=EDMULTIP(EN,s1,s2,d);

FBD/LD



(☐ is replaced by either of the following: EDMULTI, EDMULTIP.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
ED*	
ED*P	

Setting data

■Description, range, data type

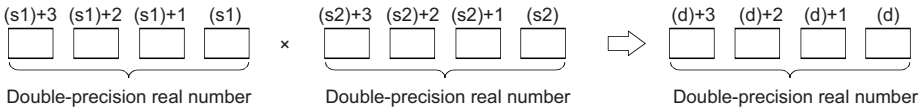
Operand	Description	Range	Data type	Data type (label)
(s1)	Multiplicand data or the start device where multiplicand data is stored	0, $2^{-1022} \leq \lvert(s1)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Multiplier data or the start device where multiplier data is stored	0, $2^{-1022} \leq \lvert(s2)\rvert < 2^{1024}$	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☐\☐, U3E☐\(H)G☐	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 48 Precautions

Operation error

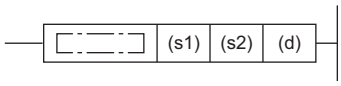
Error code (SD0)	Description
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert < 2^{1024}$

Dividing double-precision real numbers

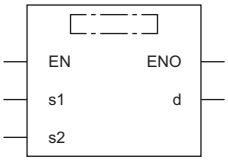
ED/(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions perform division between double-precision real numbers.

Ladder	ST*1
	ENO:=EDDIVISION(EN,s1,s2,d); ENO:=EDDIVISIONP(EN,s1,s2,d);

FBD/LD



(☐ is replaced by either of the following: EDDIVISION, EDDIVISIONP.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
ED/	
ED/P	

Setting data

■Description, range, data type

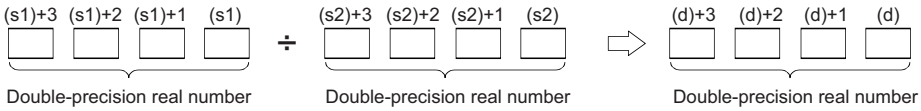
Operand	Description	Range	Data type	Data type (label)
(s1)	Dividend data or the start device where dividend data is stored	0, $2^{-1022} \leq \lvert(s1)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Divisor data or the start device where divisor data is stored	0, $2^{-1022} \leq \lvert(s2)\rvert < 2^{1024}$	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☐\☐, U3E☐\(H)G☐	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 $2^{-1022} \leq |\text{specified value (stored value)}| < 2^{1024}$ 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 48 Precautions

Operation error

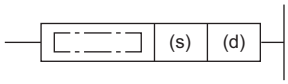
Error code (SD0)	Description
3400H	The data (divisor) in the device specified by (s2) is 0.
3402H	The value input to (s1) is -0, a subnormal number, NaN (not a number), or ±∞.
	The value input to (s2) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

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

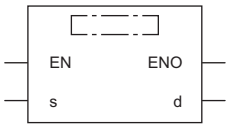
INT2FLT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert 16-bit signed binary data to a single-precision real number.

Ladder	ST*1
	ENO:=INT2FLT(EN,s,d); ENO:=INT2FLTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
INT2FLT	
INT2FLTP	

Setting data

■Description, range, data type

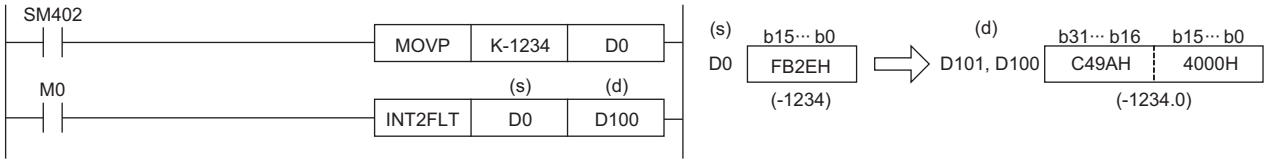
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the device containing integral data	-32768 to 32767	16-bit signed binary	ANY16_S
(d)	Start device for storing the converted single-precision real number	—	Single-precision real 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□\□, U3E□\(H)G□	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

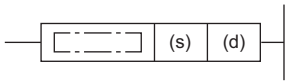
There is no operation error.

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

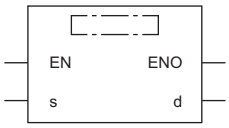
UINT2FLT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert 16-bit unsigned binary data to a single-precision real number.

Ladder	ST*1
	ENO:=UINT2FLT(EN,s,d); ENO:=UINT2FLTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
UINT2FLT	
UINT2FLTP	

Setting data

■Description, range, data type

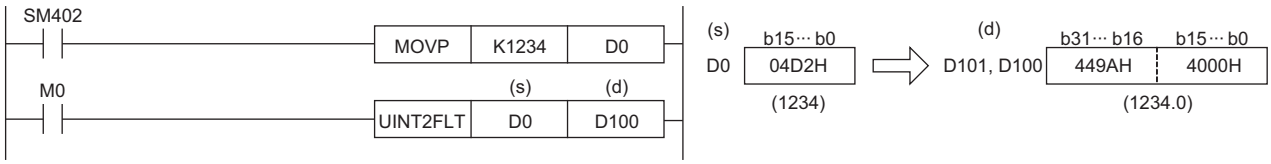
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the device containing integral data	0 to 65535	16-bit unsigned binary	ANY16_U
(d)	Start device for storing the converted single-precision real number	—	Single-precision real number	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□\□, U3E□\(H)G□	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

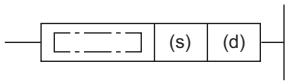
There is no operation error.

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

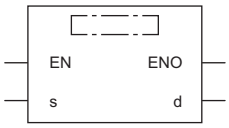
DINT2FLT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert 32-bit signed binary data to a single-precision real number.

Ladder	ST*1
	ENO:=DINT2FLT(EN,s,d); ENO:=DINT2FLTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DINT2FLT	
DINT2FLTP	

Setting data

■Description, range, data type

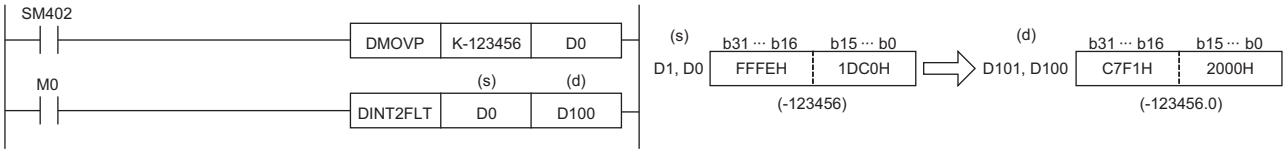
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the start device containing integral data	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
(d)	Start device for storing the converted single-precision real number	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

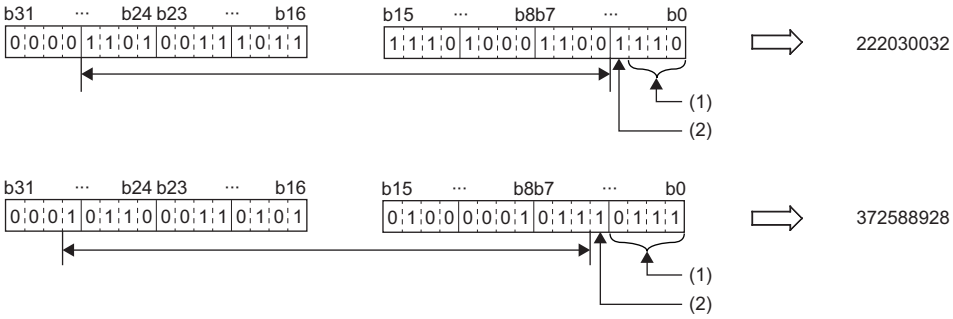
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

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

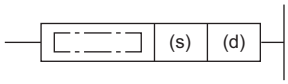
There is no operation error.

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

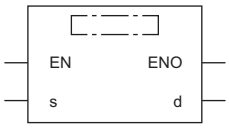
UDINT2FLT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert 32-bit unsigned binary data to a single-precision real number.

Ladder	ST*1
	ENO:=UDINT2FLT(EN,s,d); ENO:=UDINT2FLTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
UDINT2FLT	
UDINT2FLTP	

Setting data

■Description, range, data type

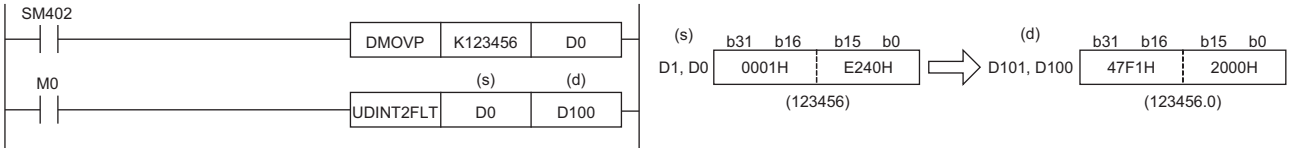
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the start device containing integral data	0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	Start device for storing the converted single-precision real number	—	Single-precision real number	ANYREAL_32

■Applicable devices

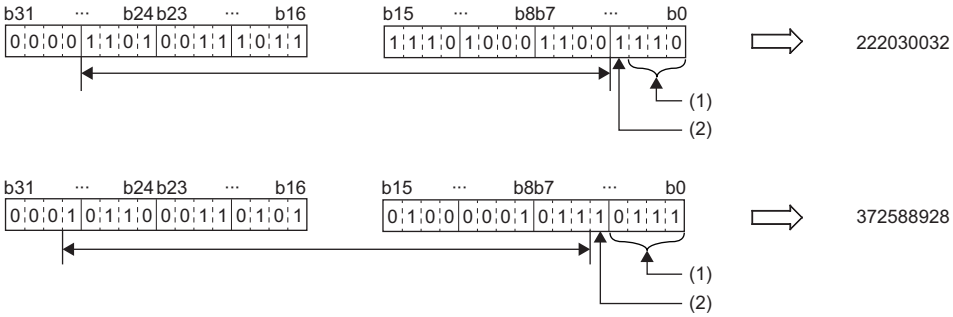
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

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

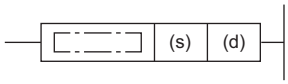
There is no operation error.

Converting double-precision real number to single-precision real number

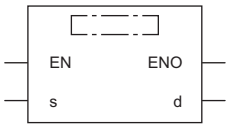
DBL2FLT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert a double-precision real number to a single-precision real number.

Ladder	ST*1
	ENO:=DBL2FLT(EN,s,d); ENO:=DBL2FLTP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DBL2FLT	
DBL2FLTP	

Setting data

■Description, range, data type

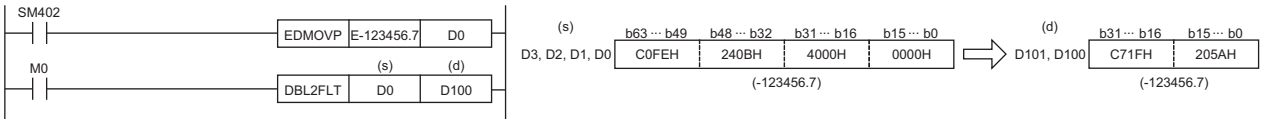
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the start device containing integral data	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the converted single-precision 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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

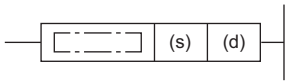
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

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

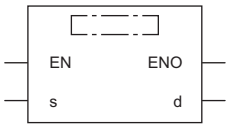
INT2DBL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert 16-bit signed binary data to a double-precision real number.

Ladder	ST*1
	ENO:=INT2DBL(EN,s,d); ENO:=INT2DBLP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
INT2DBL	
INT2DBLP	

Setting data

■Description, range, data type

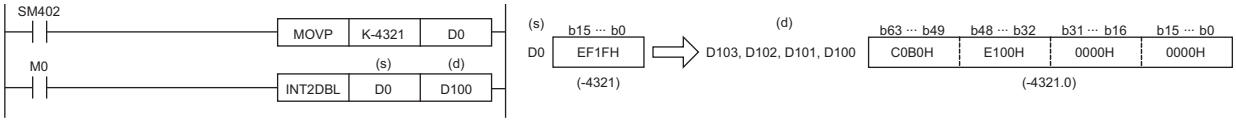
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a double-precision real number, or the device containing integral data	-32768 to 32767	16-bit signed binary	ANY16_S
(d)	Start device for storing the converted double-precision real number	—	Double-precision real 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□\□, U3E□\(H)G□	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

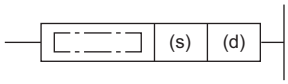
There is no operation error.

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

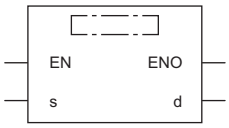
UINT2DBL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert 16-bit unsigned binary data to a double-precision real number.

Ladder	ST*1
	ENO:=UINT2DBL(EN,s,d); ENO:=UINT2DBLP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
UINT2DBL	
UINT2DBLP	

Setting data

■Description, range, data type

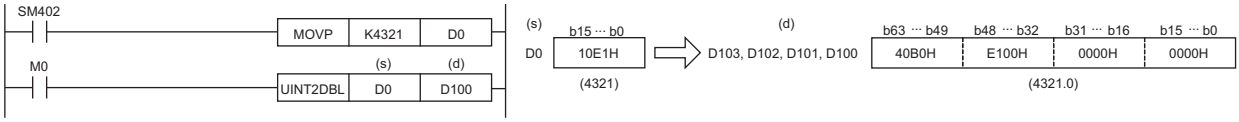
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a double-precision real number, or the device containing integral data	0 to 65535	16-bit unsigned binary	ANY16_U
(d)	Start device for storing the converted double-precision real number	—	Double-precision real number	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□\□, U3E□\(H)G□	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

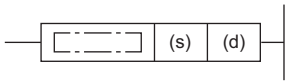
There is no operation error.

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

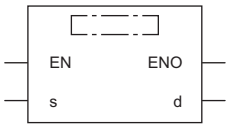
DINT2DBL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert 32-bit signed binary data to a double-precision real number.

Ladder	ST*1
	ENO:=DINT2DBL(EN,s,d); ENO:=DINT2DBLP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
DINT2DBL	
DINT2DBLP	

Setting data

■Description, range, data type

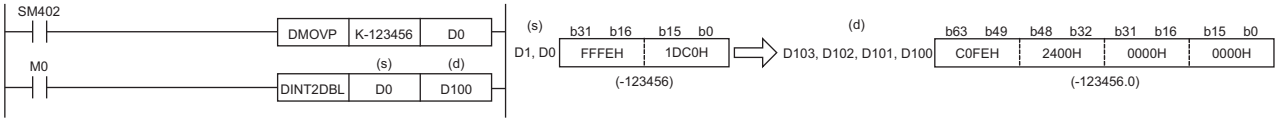
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a double-precision real number, or the start device containing integral data	-2147483648 to 2147483647	32-bit signed binary	ANY32_S
(d)	Start device for storing the converted double-precision real number	—	Double-precision 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□\□, U3E□\(H)G□	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

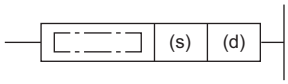
There is no operation error.

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

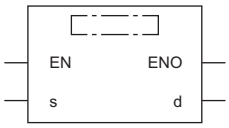
UDINT2DBL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert 32-bit unsigned binary data to a double-precision real number.

Ladder	ST*1
	ENO:=UDINT2DBL(EN,s,d); ENO:=UDINT2DBLP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
UDINT2DBL	
UDINT2DBLP	

Setting data

■Description, range, data type

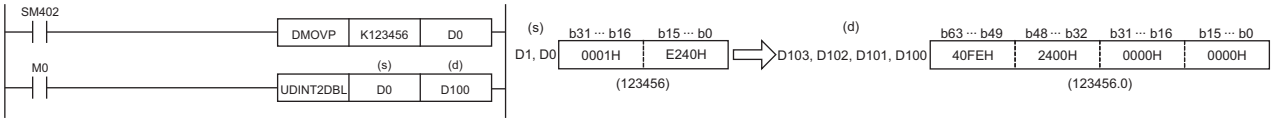
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a double-precision real number, or the start device containing integral data	0 to 4294967295	32-bit unsigned binary	ANY32_U
(d)	Start device for storing the converted double-precision real number	—	Double-precision 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□\□, U3E□\(H)G□	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

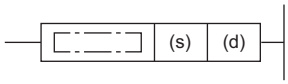
There is no operation error.

Converting single-precision real number to double-precision real number

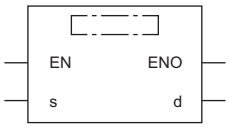
FLT2DBL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions convert a single-precision real number to a double-precision real number.

Ladder	ST*1
	ENO:=FLT2DBL(EN,s,d); ENO:=FLT2DBLP(EN,s,d);

FBD/LD



*1 The engineering tool with version "1.035M" or later supports the ST.

■Execution condition

Instruction	Execution condition
FLT2DBL	
FLT2DBLP	

Setting data

■Description, range, data type

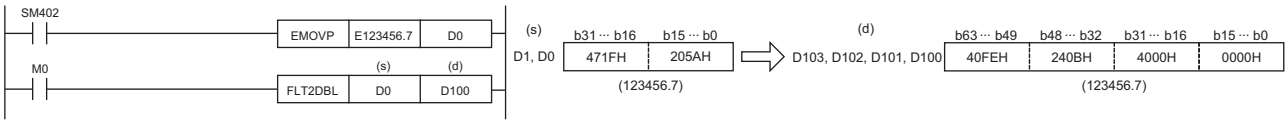
Operand	Description	Range	Data type	Data type (label)
(s)	Integral data to be converted to a single-precision real number, or the start device containing integral data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the converted double-precision 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□\□, U3E□\(H)G□	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

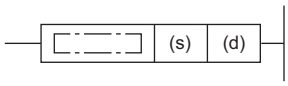
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.

Converting string data to single-precision real number

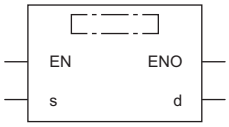
EVAL(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

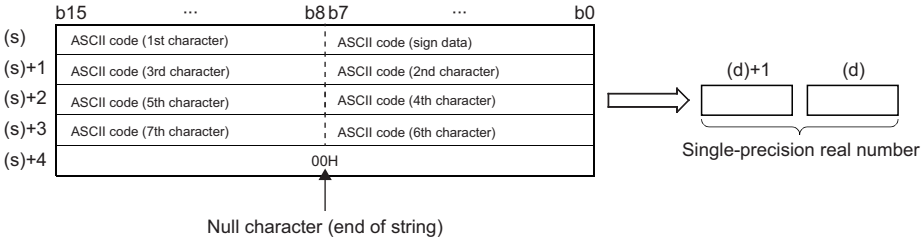
Operand	Description	Range	Data type	Data type (label)
(s)	Character string data to be converted into single-precision real number data, or the start device containing the character string data	—	String	ANYSTRING_SINGLE
(d)	Start device for storing the converted single-precision real number data	—	Single-precision 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	—	—	○	○	○	○	○	○	—	—	—	—

Processing details

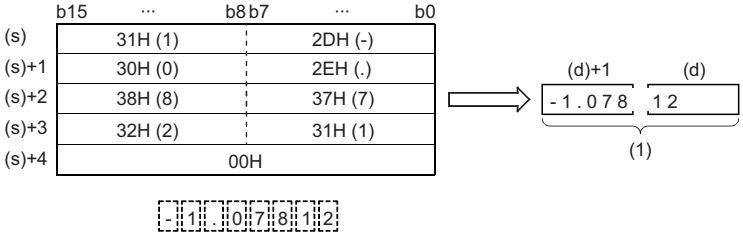
- These instructions convert the character string stored in the device number specified by (s) and later to single-precision real number data, and store the converted real number in the device specified by (d).
- The specified character string can be converted to a single-precision real number in either decimal point or exponential format.



- "20H" (space) that exists in the middle of the data is ignored.
- The character string can consist of up to 24 characters. "20H" (space) and "30H" (0) are counted as one character as well.

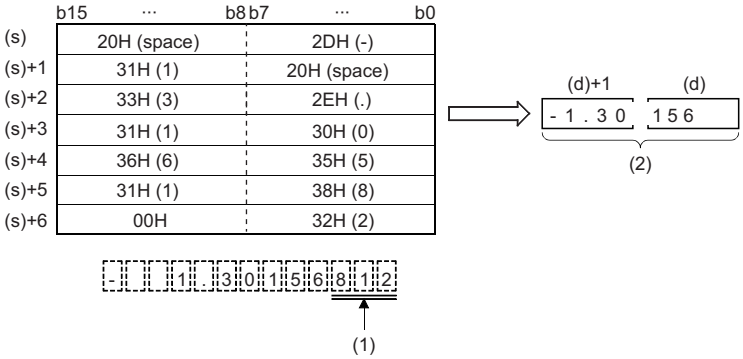
■Decimal point format

- When the character string in the device specified by (s) is in decimal point format, conversion is performed as shown below.



(1) Single-precision real number

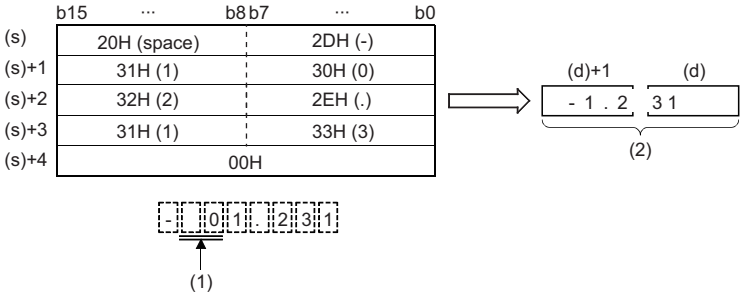
- The character string in the device specified by (s) to be converted to a single-precision real number is converted by assuming that the six digits excluding the sign, decimal point, and exponent are effective and the seventh and subsequent digits are discarded.



(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 character string in the device specified by (s), the instruction performs conversion by ignoring 20H and 30H.

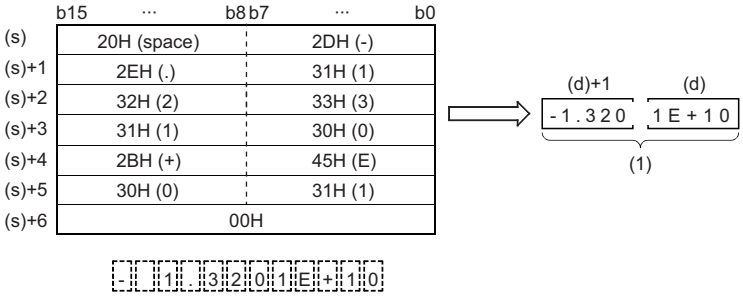


(1) Ignored.

(2) Single-precision real number

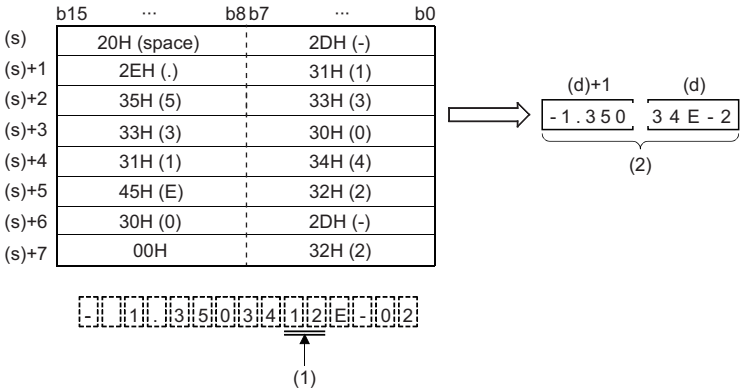
■Exponential format

- When the character string in the device specified by (s) is in exponential format, conversion is performed as shown below.



(1) Single-precision real number

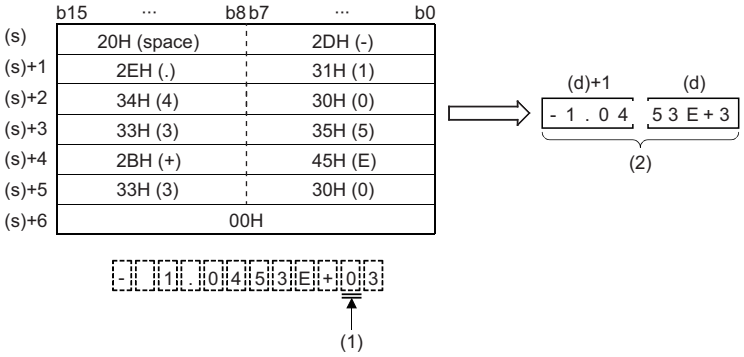
- The character string in the device specified by (s) to be converted to a single-precision real number is converted by assuming that the six digits excluding the sign, decimal point, and exponent are effective and the seventh and subsequent digits are discarded.



(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 character 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 character string in exponential format, the instruction performs conversion by ignoring 30H.



(1) Ignored.

(2) Single-precision real number

Operation error

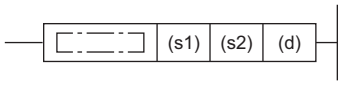
Error code (SD0)	Description
2820H	00H does not exist within the range of the relevant device specified by (s).
3401H	Invalid data that cannot be converted are set to (s). • The integral part or decimal part contains a character other than 30H (0) to 39H (9). • The specified string contains two or more 2EH (.) • 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. • The number of characters in the device specified by (s) and later is 0 or exceeds 24.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

Converting BCD format data to single-precision real number

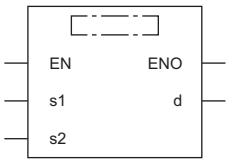
EREXP(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

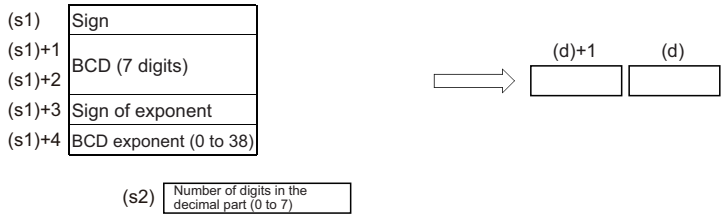
Operand	Description	Range	Data type	Data type (label)
(s1)	Start device containing BCD floating point format data	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 5)
(s2)	Number of digits in the decimal part	0 to 7	16-bit signed binary	ANY16
(d)	Start device for storing a single-precision real 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□\□, U3E□\(H)G□	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.



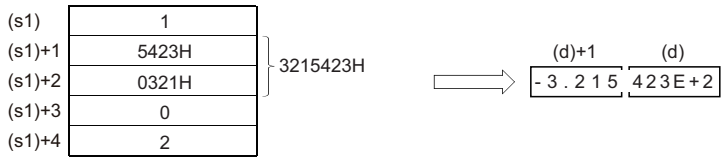
(s2) Number of digits in the decimal part (0 to 7)

(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)



(s2) 6

Operation error

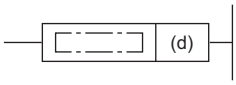
Error code (SD0)	Description
3401H	The value set to (s1) as sign data is not 0 or 1.
	A value other than 0 to 9 exists at any digit of data set to (s1)+1 and (s1)+2.
	The value set to (s1)+3 as sign data of exponent is not 0 or 1.
	The exponent data set to (s1)+4 is out of the range, 0 to 38.
	The value set to (s2) as the number of digits in the decimal part is out of the range, 0 to 7.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

Inverting the sign of single-precision real number

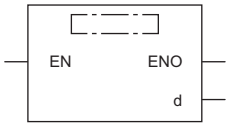
ENEG(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

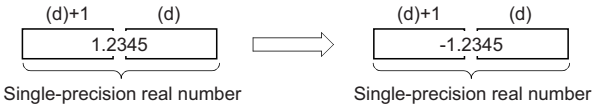
Operand	Description	Range	Data type	Data type (label)
(d)	Start device containing the single-precision real number data subject to sign inversion	—	Single-precision real number	ANYREAL_32
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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

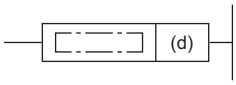
Error code (SD0)	Description
3402H	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.

Inverting the sign of double-precision real number

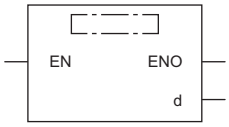
EDNEG(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

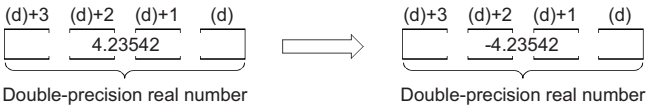
Operand	Description	Range	Data type	Data type (label)
(d)	Start device containing the double-precision real number subject to sign inversion	—	Double-precision real number	ANYREAL_64
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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

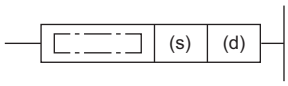
Error code (SD0)	Description
3402H	The value input to (d) is -0, a subnormal number, NaN (not a number), or ±∞.

Transferring single-precision real number

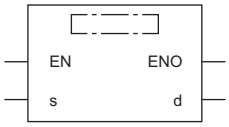
EMOV(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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

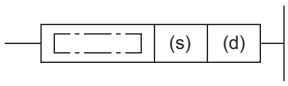
There is no operation error.

Transferring double-precision real number

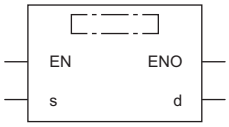
EDMOV(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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

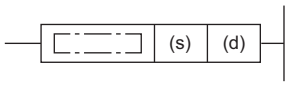
There is no operation error.

Calculating the sine of single-precision real number

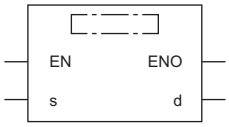
SIN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1448 SIN(_E)

■Execution condition

Instruction	Execution condition
SIN	
SINP	

Setting data

■Description, range, data type

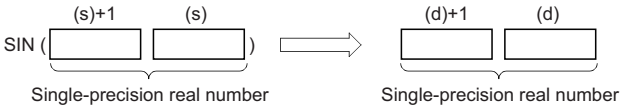
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for sine calculation, or the start device containing the angle data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real 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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

Error code (SD0)	Description
3402H	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.

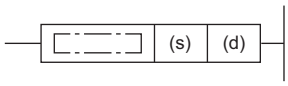
☞ Page 960 RAD(P)
☞ Page 962 DEG(P)

Calculating the cosine of single-precision real number

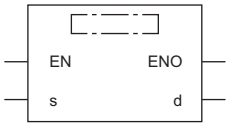
COS(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1449 COS(_E)

■Execution condition

Instruction	Execution condition
COS	
COSP	

Setting data

■Description, range, data type

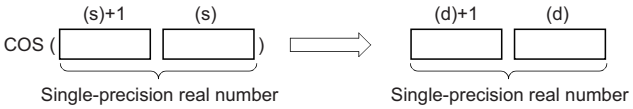
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for cosine calculation, or the start device containing the angle data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real 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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

Error code (SD0)	Description
3402H	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.

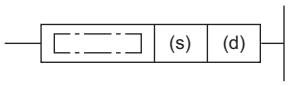
☞ Page 960 RAD(P)
☞ Page 962 DEG(P)

Calculating the tangent of single-precision real number

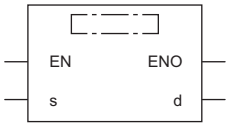
TAN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1450 TAN(_E)

■Execution condition

Instruction	Execution condition
TAN	
TANP	

Setting data

■Description, range, data type

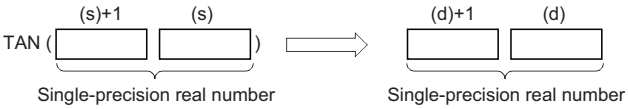
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for tangent calculation, or the start device containing the angle data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real 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□\□, U3E□\(H)G□	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 48 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
3402H	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.

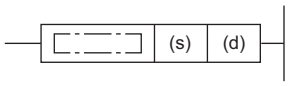
☞ Page 960 RAD(P)
☞ Page 962 DEG(P)

Calculating the arc sine of single-precision real number

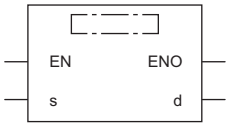
ASIN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1451 ASIN(_E)

■Execution condition

Instruction	Execution condition
ASIN	
ASINP	

Setting data

■Description, range, data type

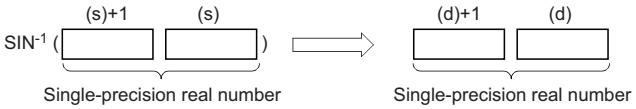
Operand	Description	Range	Data type	Data type (label)
(s)	Sine data used for arc sine calculation, or the start device containing the sine data	-1.0 to 1.0	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution 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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	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.

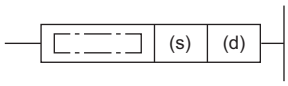
☞ Page 960 RAD(P)
☞ Page 962 DEG(P)

Calculating the arc cosine of single-precision real number

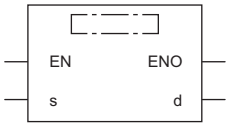
ACOS(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1452 ACOS(_E)

■Execution condition

Instruction	Execution condition
ACOS	
ACOSP	

Setting data

■Description, range, data type

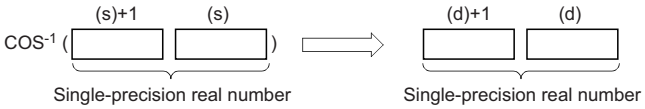
Operand	Description	Range	Data type	Data type (label)
(s)	Cosine data used for arc cosine calculation, or the start device containing the cosine data	-1.0 to 1.0	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	ANYREAL_32
EN	Execution 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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	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.

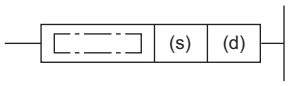
☞ Page 960 RAD(P)
☞ Page 962 DEG(P)

Calculating the arc tangent of single-precision real number

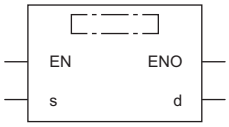
ATAN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1453 ATAN(_E)

■Execution condition

Instruction	Execution condition
ATAN	
ATANP	

Setting data

■Description, range, data type

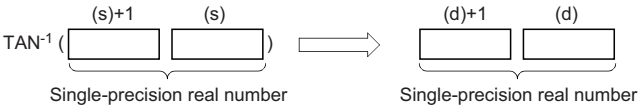
Operand	Description	Range	Data type	Data type (label)
(s)	Tangent data used for arc tangent calculation, or the start device containing the tangent data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision 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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

Error code (SD0)	Description
3402H	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.

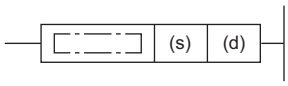
☞ Page 960 RAD(P)
☞ Page 962 DEG(P)

Calculating the sine of double-precision real number

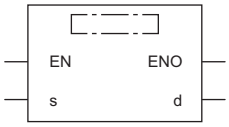
SIND(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1448 SIN(_E)

■Execution condition

Instruction	Execution condition
SIND	
SINDP	

Setting data

■Description, range, data type

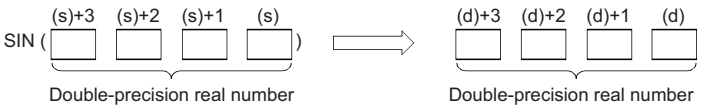
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for sine calculation, or the start device containing the angle data	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real 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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

Error code (SD0)	Description
3402H	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.

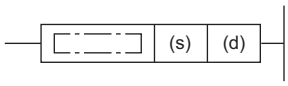
☞ Page 964 RADD(P)
☞ Page 966 DEGD(P)

Calculating the cosine of double-precision real number

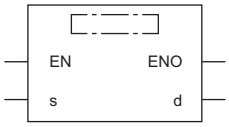
COSD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1449 COS(_E)

■Execution condition

Instruction	Execution condition
COSD	
COSDP	

Setting data

■Descriptions, ranges, and data types

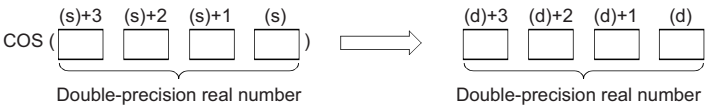
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for cosine calculation, or the start device containing the angle data	0, $2^{-1022} \leq (s) \mid < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

Error code (SD0)	Description
3402H	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.

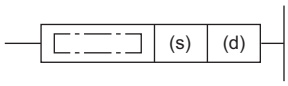
☞ Page 964 RADD(P)
☞ Page 966 DEGD(P)

Calculating the tangent of double-precision real number

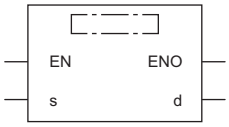
TAND(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1450 TAN(_E)

■Execution condition

Instruction	Execution condition
TAND	
TANDP	

Setting data

■Description, range, data type

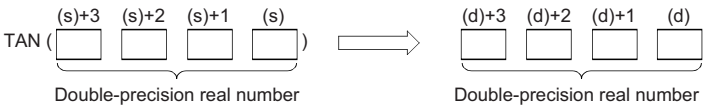
Operand	Description	Range	Data type	Data type (label)
(s)	Angle data used for tangent calculation, or the start device containing the angle data	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision 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□\□, U3E□\(H)G□	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 48 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
3402H	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.

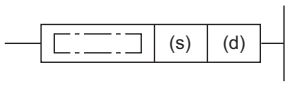
☞ Page 964 RADD(P)
☞ Page 966 DEGD(P)

Calculating the arc sine of double-precision real number

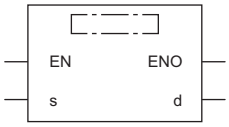
ASIND(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1451 ASIN(_E)

■Execution condition

Instruction	Execution condition
ASIND	
ASINDP	

Setting data

■Description, range, data type

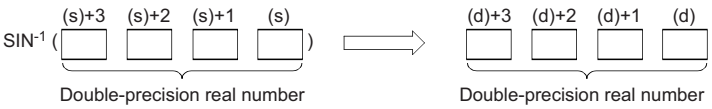
Operand	Description	Range	Data type	Data type (label)
(s)	Sine data used for arc sine calculation, or the start device containing the sine data	-1.0 to 1.0	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution 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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	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.

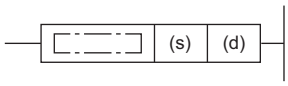
☞ Page 964 RADD(P)
☞ Page 966 DEGD(P)

Calculating the arc cosine of double-precision real number

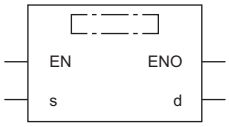
ACOSD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1452 ACOS(_E)

■Execution condition

Instruction	Execution condition
ACOSD	
ACOSDP	

Setting data

■Description, range, data type

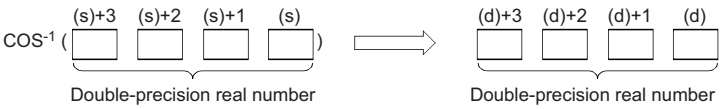
Operand	Description	Range	Data type	Data type (label)
(s)	Cosine data used for arc cosine calculation, or the start device containing the cosine data	-1.0 to 1.0	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	ANYREAL_64
EN	Execution 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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	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.

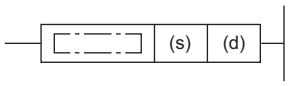
☞ Page 964 RADD(P)
☞ Page 966 DEGD(P)

Calculating the arc tangent of double-precision real number

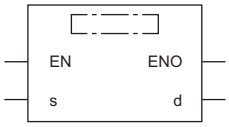
ATAND(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1453 ATAN(_E)

■Execution condition

Instruction	Execution condition
ATAND	
ATANDP	

Setting data

■Description, range, data type

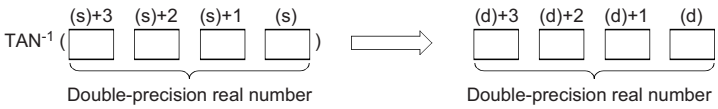
Operand	Description	Range	Data type	Data type (label)
(s)	Tangent data used for arc tangent calculation, or the start device containing the tangent data	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision 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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

Error code (SD0)	Description
3402H	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.

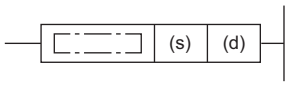
☞ Page 964 RADD(P)
☞ Page 966 DEGD(P)

Calculating the sine of BCD data

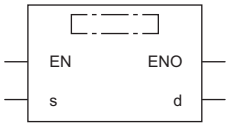
BSIN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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)	(d)+1		(d)+2
SIN (s) =			.	

(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

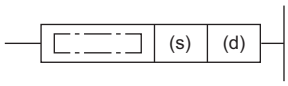
Error code (SD0)	Description
3405H	An out-of-range value is set to (s). • The specified data is not a BCD value. • The specified data is out of the range, 0 to 360.

Calculating the cosine of BCD data

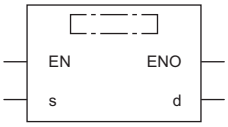
BCOS(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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)	(d)+1		(d)+2
COS (s) =			.	

(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

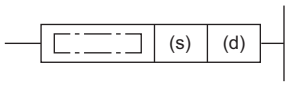
Error code (SD0)	Description
3405H	An out-of-range value is set to (s). • The specified data is not a BCD value. • The specified data is out of the range, 0 to 360.

Calculating the tangent of BCD data

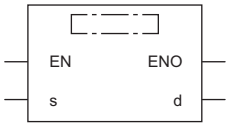
BTAN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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)	(d)+1		(d)+2
TAN (s) =			.	

(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

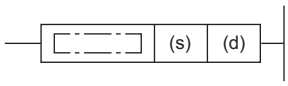
Error code (SD0)	Description
3405H	An out-of-range value is set to (s). • The specified data is not a BCD value. • The specified data is out of the range, 0 to 360. • The specified data is 90° or 270°.

Calculating the arc sine of BCD data

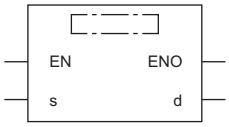
BASIN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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)	(s)+1		(s)+2	
SIN^{-1} (	[ ]	[ ]	.	[ ]	) = (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

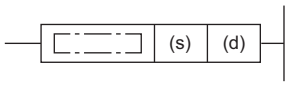
Error code (SD0)	Description
3405H	An out-of-range value is set to (s). • The specified data is not a BCD value. • The specified data is out of the range, -1.0000 to 1.0000.

Calculating the arc cosine of BCD data

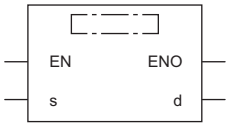
BACOS(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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) (s)+1 (s)+2

COS^{-1} ([ ] [ ] . [ ]) = (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

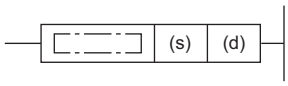
Error code (SD0)	Description
3405H	An out-of-range value is set to (s). • The specified data is not a BCD value. • The specified data is out of the range, -1.0000 to 1.0000.

Calculating the arc tangent of BCD data

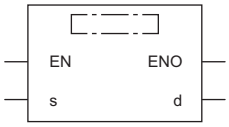
BATAN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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) (s)+1 (s)+2

TAN^{-1} (Sign Integral part . Decimal part) = (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

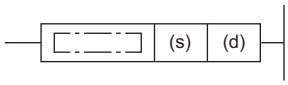
Error code (SD0)	Description
3405H	The data in the device specified by (s) is not a BCD value.

Converting single-precision real number angle to radian

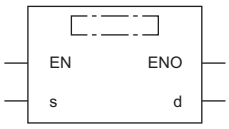
RAD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

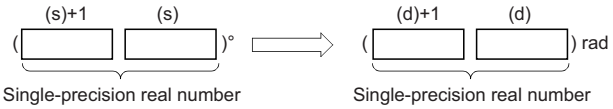
Operand	Description	Range	Data type	Data type (label)
(s)	Angle for which the unit is to be changed to radian, or the start device containing the angle	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the angle in radians	—	Single-precision 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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

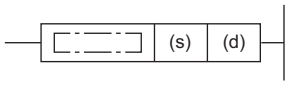
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

Converting single-precision real number radian to angle

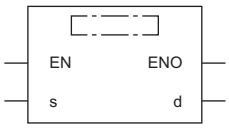
DEG(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

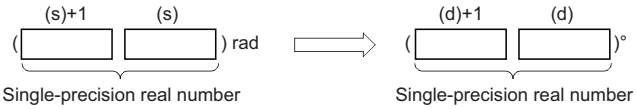
Operand	Description	Range	Data type	Data type (label)
(s)	Angle in radians for which the unit is to be changed to the degree, or the start device containing the angle in radians	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the value 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□\□, U3E□\(H)G□	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.

$$\text{Degree} = \text{Radian} \times \frac{\pi}{180}$$

- 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 48 Precautions

Operation error

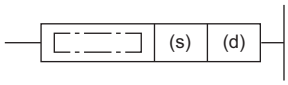
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

Converting double-precision real number angle to radian

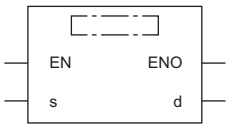
RADD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

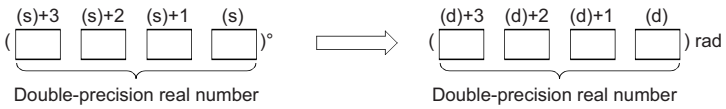
Operand	Description	Range	Data type	Data type (label)
(s)	Angle for which the unit is to be changed to radian, or the start device containing the angle	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the angle in radians	—	Double-precision 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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

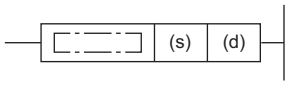
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

Converting double-precision real number radian to angle

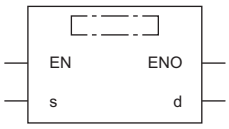
DEGD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

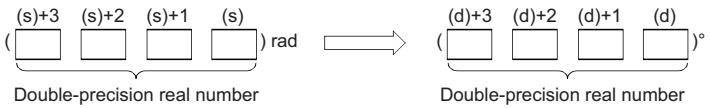
Operand	Description	Range	Data type	Data type (label)
(s)	Angle in radians for which the unit is to be changed to the degree, or the start device containing the angle in radians	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the value 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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

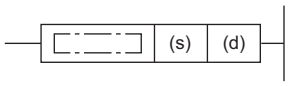
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

Calculating the square root of single-precision real number

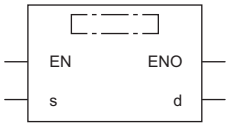
ESQRT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

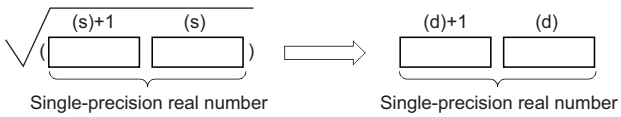
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for square root operation, or the start device containing the data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

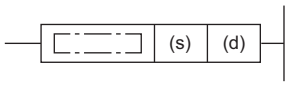
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	The value in the device specified by (s) is a negative number.

Calculating the square root of double-precision real number

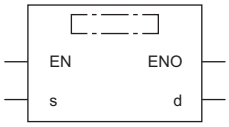
EDSQRT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

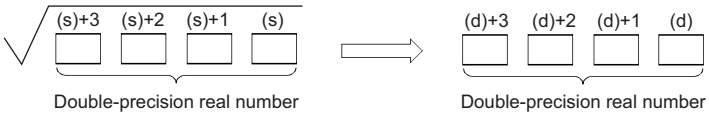
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for square root operation, or the start device containing the data	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

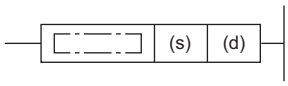
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	The value in the device specified by (s) is a negative number.

Calculating the exponent of single-precision real number

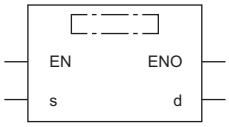
EXP(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1447 EXP(_E)

■Execution condition

Instruction	Execution condition
EXP	
EXPP	

Setting data

■Description, range, data type

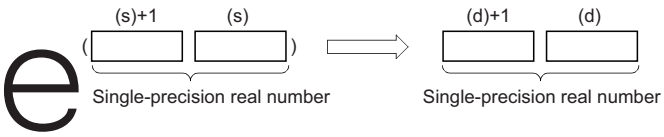
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for exponential operation, or the start device containing the data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

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).

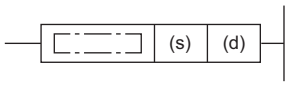
$$10^{X} = e^{\frac{X}{0.43429}}$$

Calculating the exponent of double-precision real number

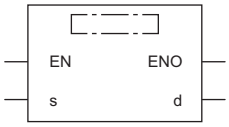
EXPD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1447 EXP(_E)

■Execution condition

Instruction	Execution condition
EXPD	
EXPDP	

Setting data

■Description, range, data type

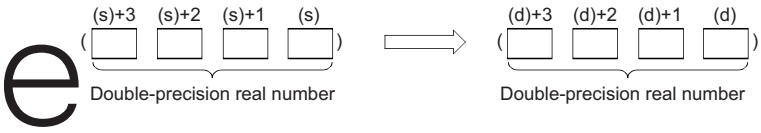
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for exponential operation, or the start device containing the data	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real number	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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

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).

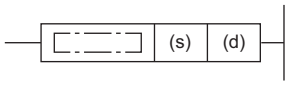
$$10^X = e^{\frac{X}{0.43429}}$$

Calculating the natural logarithm of single-precision real number

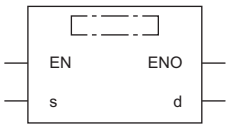
LOG(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1445 LOG(_E)

■Execution condition

Instruction	Execution condition
LOG	
LOGP	

Setting data

■Description, range, data type

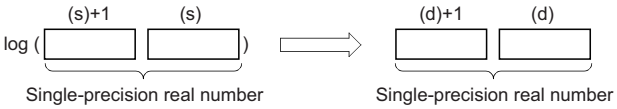
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for natural logarithm operation, or the start device containing the data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

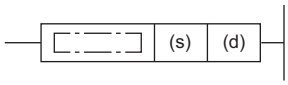
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	An out-of-range value is set to (s). • The specified value is a negative number. • The specified value is 0.

Calculating the natural logarithm of double-precision real number

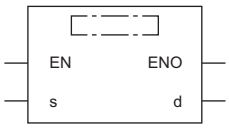
LOGD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1445 LOG(_E)

■Execution condition

Instruction	Execution condition
LOGD	
LOGDP	

Setting data

■Description, range, data type

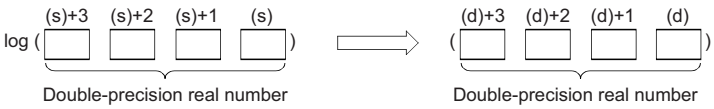
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for natural logarithm operation, or the start device containing the data	0, $2^{-1022} \leq \lvert(s)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(d)	Start device for storing the operation result	—	Double-precision real 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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

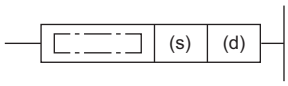
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	An out-of-range value is set to (s). • The specified value is a negative number. • The specified value is 0.

Calculating the square root of BCD 4-digit data

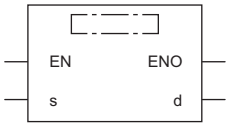
BSQRT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

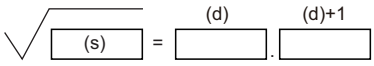
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for square root operation, or the device containing the data	0 to 9999	BCD 4-digit	ANY16
(d)	Start device for storing the operation result	—	BCD 4-digit	ANY16_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
ENO	Execution 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□\□, U3E□\(H)G□	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

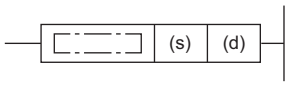
Error code (SD0)	Description
3405H	The data in the device specified by (s) is not a BCD value.

Calculating the square root of BCD 8-digit data

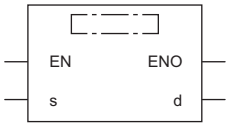
BDSQRT(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

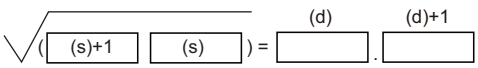
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for square root operation, or the start device containing the data	0 to 99999999	BCD 8-digit	ANY32
(d)	Start device for storing the operation result	—	BCD 4-digit	ANY16_ARRAY (Number of elements: 2)
EN	Execution condition	—	Bit	BOOL
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□\□, U3E□\(H)G□	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

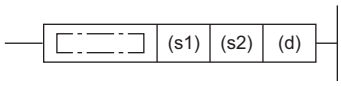
Error code (SD0)	Description
3405H	The data in the device specified by (s) is not a BCD value.

Calculating the exponentiation of single-precision real number

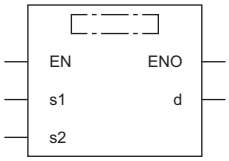
POW(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

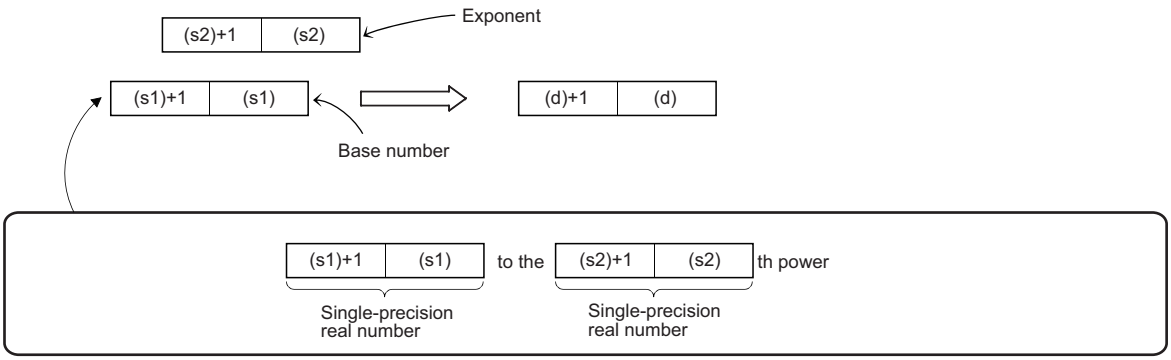
Operand	Description	Range	Data type	Data type (label)
(s1)	Exponentiation recipient data or the start device containing the exponentiation recipient data	0, $2^{-126} \leq \lvert(s1)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(s2)	Exponentiation data or the start device containing the data	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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

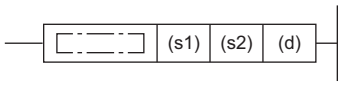
Error code (SD0)	Description
3402H	The value input to (s1) or (s2) is -0, a subnormal number, NaN (not a number), or ±∞.

Calculating the exponentiation of double-precision real number

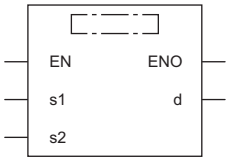
POWD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

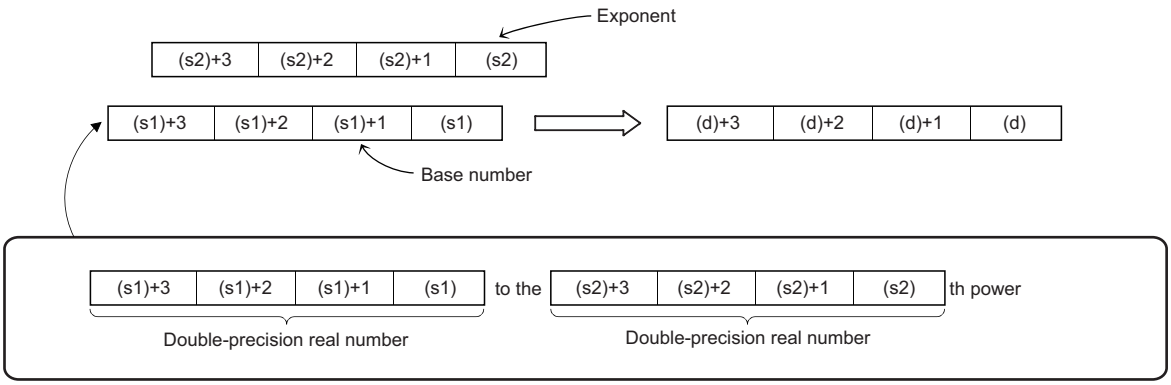
Operand	Description	Range	Data type	Data type (label)
(s1)	Exponentiation recipient data or the start device containing the exponentiation recipient data	0, $2^{-1022} \leq \lvert(s1)\rvert < 2^{1024}$	Double-precision real number	ANYREAL_64
(s2)	Exponentiation data or the start device containing the 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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

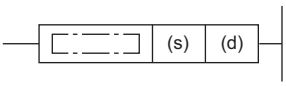
Error code (SD0)	Description
3402H	The value input to (s1) or (s2) is -0, a subnormal number, NaN (not a number), or ±∞.

Calculating the common logarithm of single-precision real number

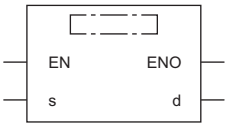
LOG10(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

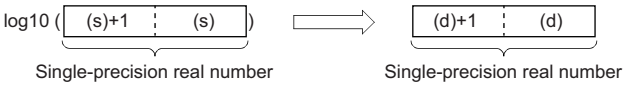
Operand	Description	Range	Data type	Data type (label)
(s)	Data used for common logarithm operation, or the start device containing the data	0, $2^{-126} \leq \lvert(s)\rvert < 2^{128}$	Single-precision real number	ANYREAL_32
(d)	Start device for storing the operation result	—	Single-precision real number	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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

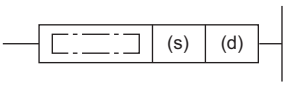
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	An out-of-range value is set to (s). • The specified value is a negative number. • The specified value is 0.

Calculating the common logarithm of double-precision real number

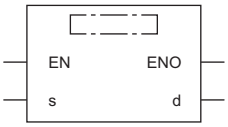
LOG10D(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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 48 Precautions

Operation error

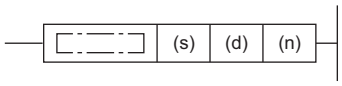
Error code (SD0)	Description
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	An out-of-range value is set to (s). • The specified value is a negative number. • The specified value is 0.

Searching the maximum value of single-precision real number

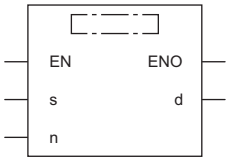
EMAX(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1483 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

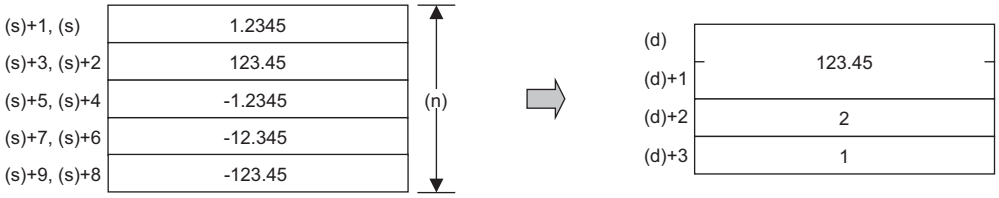
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.
*2 Specify a device regardless of the programming language used. Do not specify labels.

■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□\□, U3E□\(H)G□	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: Location

(d)+3: Number of maximum values

- When (n) is 0, the processing is not performed.

Operation error

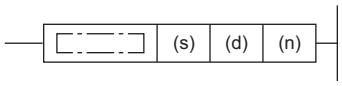
Error code (SD0)	Description
3402H	The block data in the device specified by (s) includes a value other than single-precision real number.

Searching the maximum value of double-precision real number

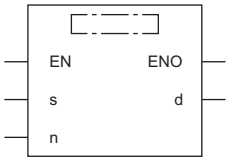
EDMAX(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1483 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

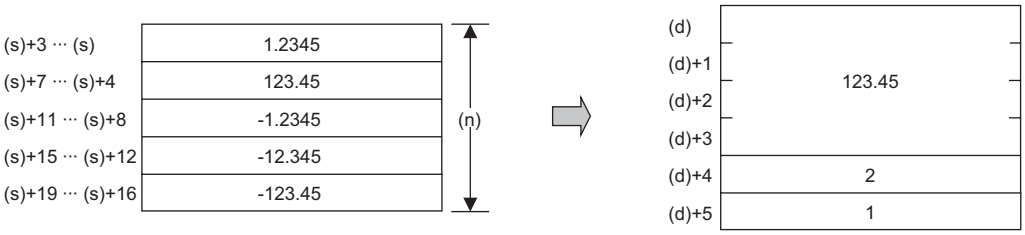
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.
*2 Specify a device regardless of the programming language used. Do not specify labels.

■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□\□, U3E□\(H)G□	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

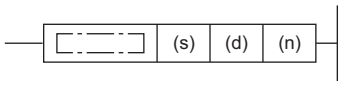
Error code (SD0)	Description
3402H	The block data in the device specified by (s) includes a value other than double-precision real number.

Searching the minimum value of single-precision real number

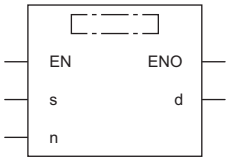
EMIN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1483 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

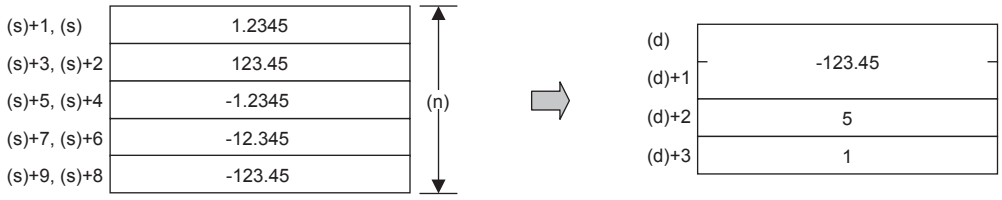
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.
*2 Specify a device regardless of the programming language used. Do not specify labels.

■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□\□, U3E□\(H)G□	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: Location

(d)+3: Number of minimum values

- When (n) is 0, the processing is not performed.

Operation error

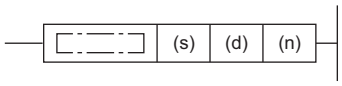
Error code (SD0)	Description
3402H	The block data in the device specified by (s) includes a value other than single-precision real number.

Searching the minimum value of double-precision real number

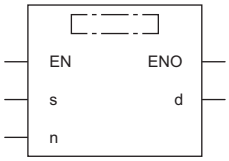
EDMIN(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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 1483 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

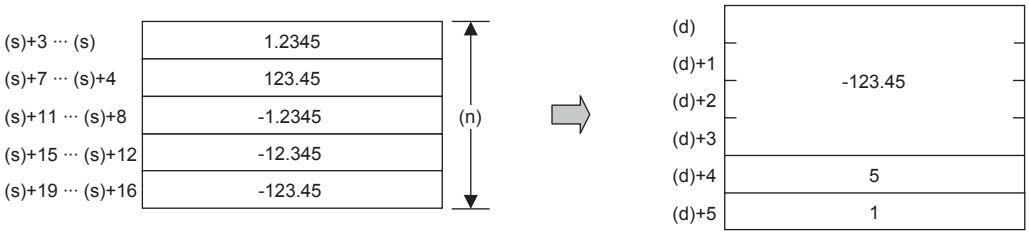
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.
*2 Specify a device regardless of the programming language used. Do not specify labels.

■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□\□, U3E□\(H)G□	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
3402H	The block data in the device specified by (s) includes a value other than double-precision real number.

12 RANDOM NUMBER

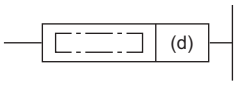
12.1 Random Number Instructions

Generating random number

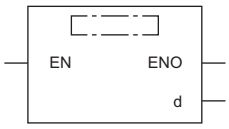
RND(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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

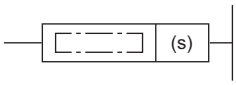
There is no operation error.

Changing random sequence

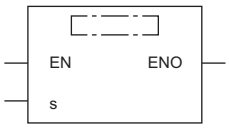
SRND(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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.

13 DEVICE OPERATION

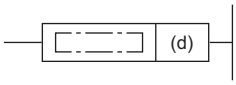
13.1 Index Register Instructions

Saving all data of the index register

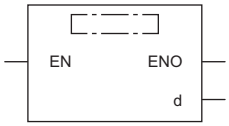
ZPUSH(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

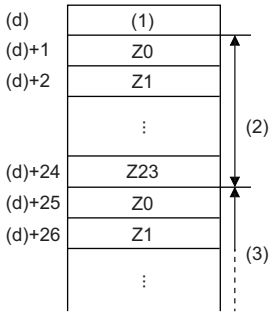
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions save the content of the index register to the device specified by (d) or the array label element and later.
- 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 and long index register, 24 words of data are saved. Accordingly, when 0 point is assigned to the index register, the long index register is saved by 12 points.
- 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 1004 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.
- The following figure shows the configuration of the areas used after (d).



(1) Number of saves
(2) 1st nesting (24 words)
(3) 2nd nesting

Operation error

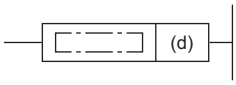
Error code (SD0)	Description
3405H	The number of saves in (d)+0 is FFFF.

Returning all data of the index register

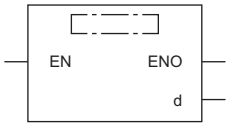
ZPOP(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	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.
- When the content of the index register is read, the number of saves (d) is decremented by 1.
- Refer to the following for the configuration of the areas used after (d).

☞ Page 1002 ZPUSH(P)

Operation error

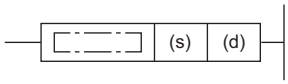
Error code (SD0)	Description
3405H	The number of saves in (d)+0 is 0.

Saving the selected data of the index register and long index register

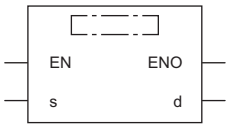
ZPUSH(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

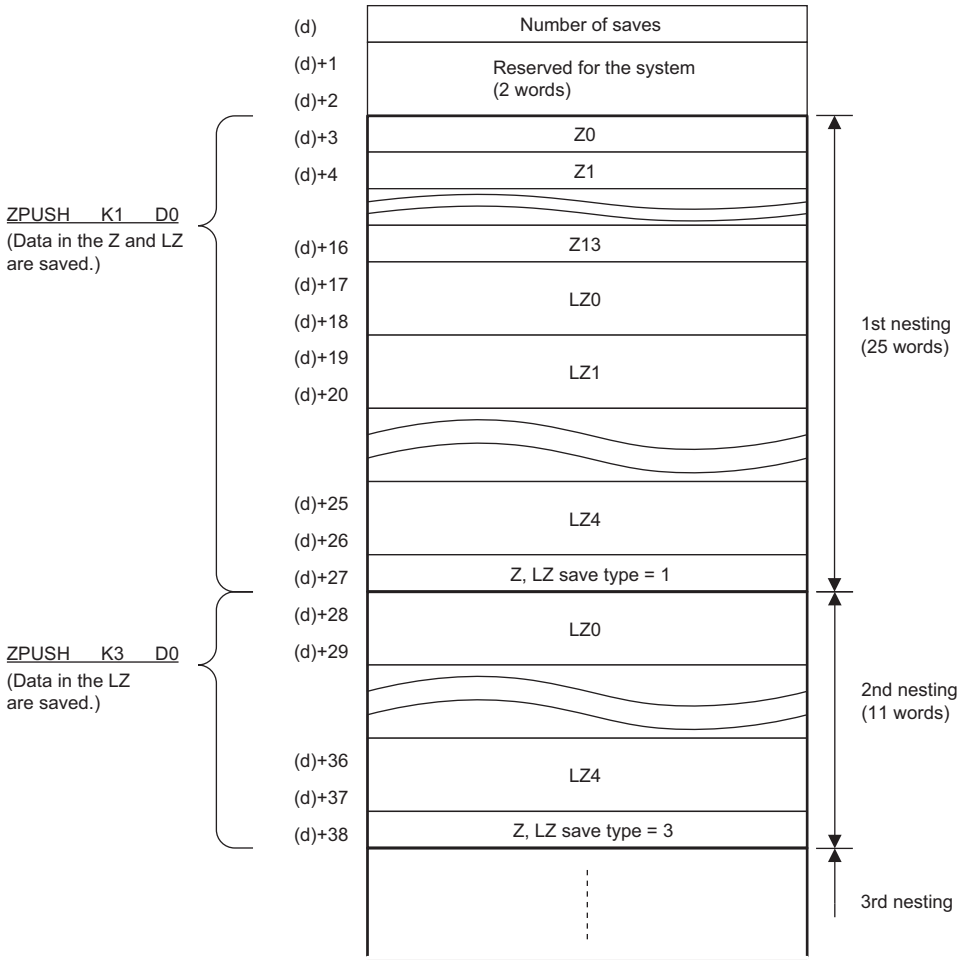
- These instructions save the contents of the index register and long index register in the device specified by (s) to the device specified by (d) or array label element and later. The instructions also save the type of the saved index register and long index register to the end of the saved data.
- When the contents of the index register and long index register are saved, the number of saves (d) is incremented by 1.
- The following table lists the values specified by (s) and the index registers and long index registers 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 1008 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.
- The following figure shows the configuration of the areas used after (d). (Z0 to Z23 and LZ0 to LZ4)



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

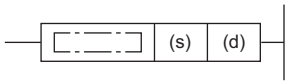
Error code (SD0)	Description
3405H	An out-of-range value is set to (s). • The specified value is other than 1 to 3. • When the number of index register points is 0, 2 is specified. • When the number of long index register points is 0, 3 is specified.
	The value stored in the system-reserved area in the area specified by (d) has been changed.

Returning the selected data of the index register and long index register

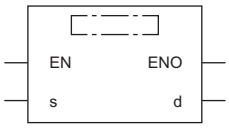
ZPOP(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	○	—	—	○	○	—	—	—
(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 or 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 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 1005 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
3405H	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.

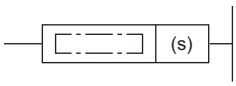
13.2 File Register Operation Instructions

Switching the file register block number

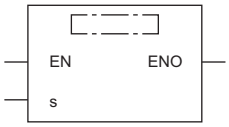
RSET(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

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

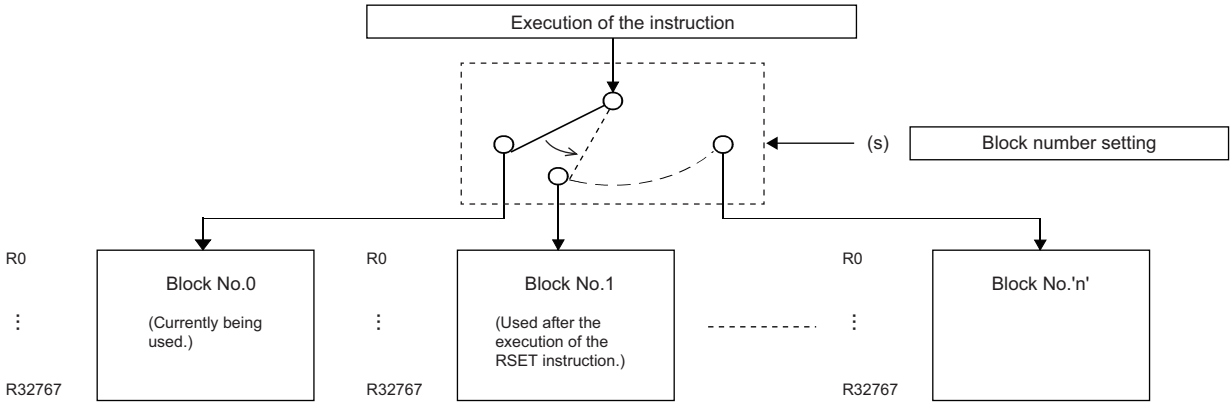
Operand	Description	Range	Data type	Data type (label)
(s)	Block number data to be changed or the device number where the block number data is stored	0 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■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□\□, U3E□\(H)G□	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

For the restrictions of the file register, refer to the following.

☞ Page 64 Restrictions on using file registers

Operation error

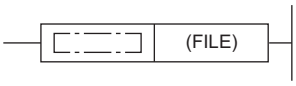
Error code (SD0)	Description
2820H	The specified file register does not exist.
3405H	The block number in the device specified by (s) does not exist.

Changing the file register file name

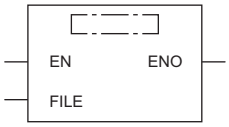
QDRSET(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions change the file name of the file register used in the program to that stored in the device specified by (file name).

Ladder	ST
 FILE: File name	ENO:=QDRSET(EN,filename); ENO:=QDRSETP(EN,filename);

FBD/LD



FILE: File name

■Execution condition

Instruction	Execution condition
QDRSET	
QDRSETP	

Setting data

■Description, range, data type

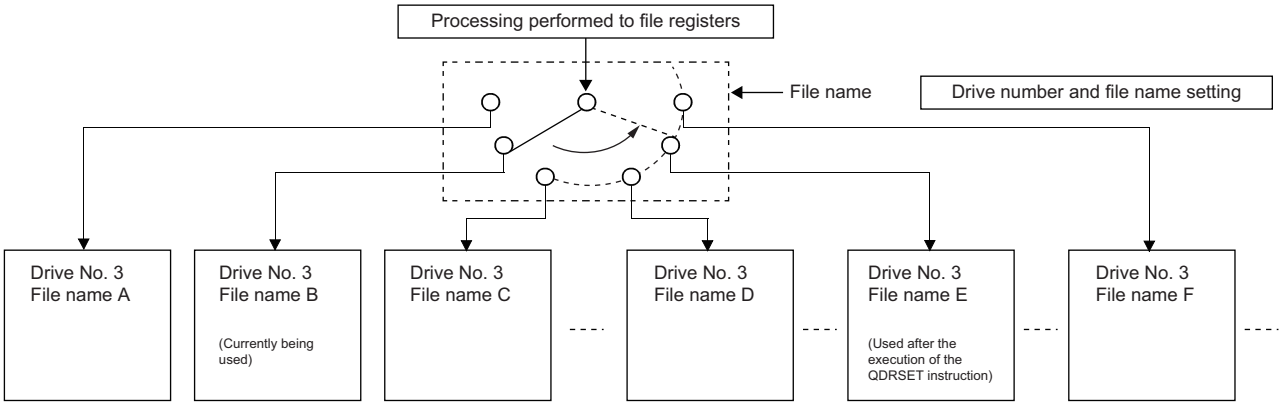
Operand	Description	Range	Data type	Data type (label)
(File name)	String data of drive number file name of the file register to be set, or the start device where the string data is stored Example: "1: ABC"	—	Unicode string	ANYSTRING_DOUBLE
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(File name)	—	—	○	—	—	—	—	○	—	—	○	—

Processing details

- These instructions change the file name of the file register used in the program to that stored in the device specified by (file name). After the file name is changed, all file registers processed by the program are those linked to block number 0 of the new file name. The RSET(P) instruction is used to change the block number.



- For the drive number, 1 or 3 can be specified.
- When drive 1 is specified, the drive works as drive 3. The use status of the drive is reflected to SD614. It is not reflected to SD604.
- Extension ".QDR" need not be specified for the file name.
- Even if the drive number file name is specified by a parameter, the file name specified by the QDRSET(P) instruction takes precedence.

Point

- If the file name is changed by the QDRSET(P) instruction, operating the CPU module switch from STOP to RUN restores the file name specified by the parameter. To continue to use the file name specified by the QDRSET(P) instruction even when the CPU module switch is changed from STOP to RUN, execute the QDRSET(P) instruction using SM402 that triggers one scan when the switch is changed from STOP to RUN.
- When a file register is specified for the refresh device, do not use the QDRSET(P) instruction to change the file name of the file register.

Precautions

- Even when the NULL code (0000H) is specified for the file name, the file name setting is not cleared and no processing is performed.
- For the restrictions of the file register, refer to the following.

☞ Page 64 Restrictions on using file registers

Operation error

Error code (SD0)	Description
2840H	The drive number/file name specified by (file name) does not exist.
3405H	Out-of-range data is set to (file name). • A drive number other than 1 and 3 is specified. • Only the drive number is specified.

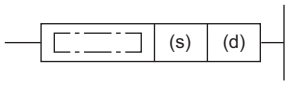
13.3 File Register Read/Write Instructions

Reading 1-byte data from the file register

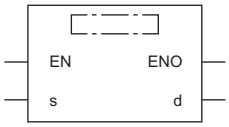
ZRRDB(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

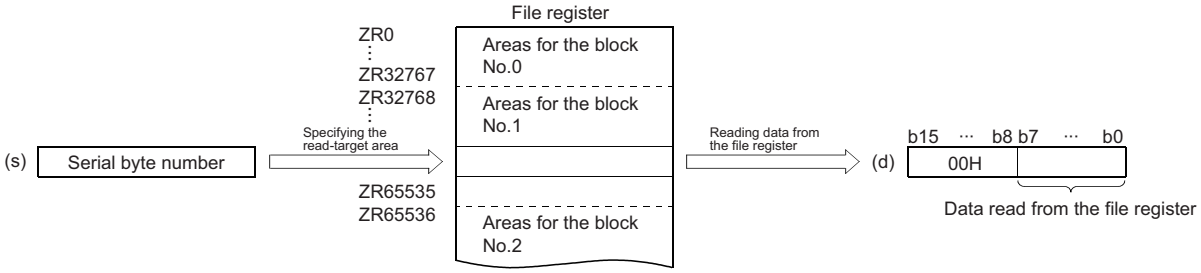
Operand	Description	Range	Data type	Data type (label)
(s)	Serial byte number of the file register to be read	0 to 4294967295	32-bit unsigned binary	ANY32
(d)	Start number of the device for storing the data that has been read	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

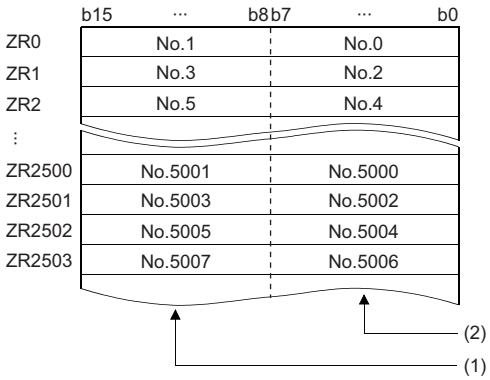
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	○	○	○	○	—	—	—
(d)	○	○	○	○	○	—	—	○	—	—	—	—

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.

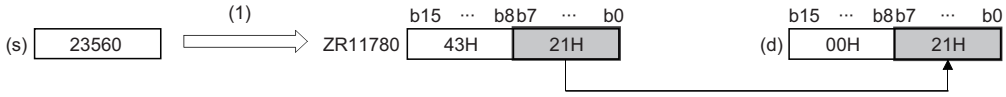


(1) Data areas when an odd number is specified

(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

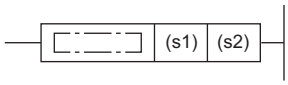
Error code (SD0)	Description
2820H	The specified device number (serial byte number) is out of range.

Writing 1-byte data to the file register

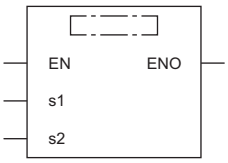
ZRWRB(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

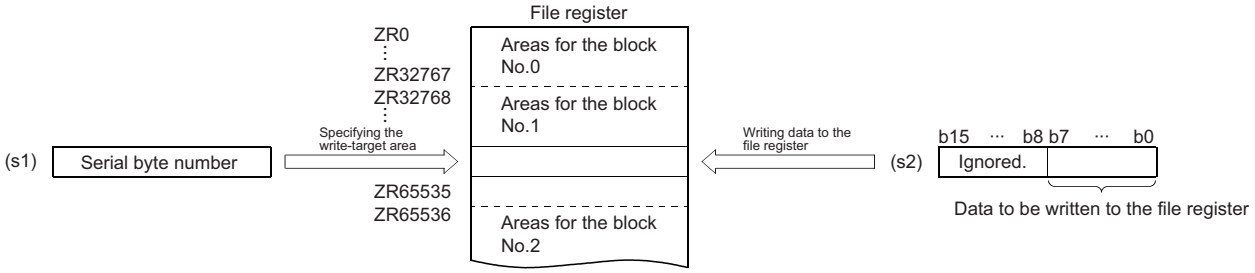
Operand	Description	Range	Data type	Data type (label)
(s1)	Serial byte number of the file register to be written	0 to 4294967295	32-bit unsigned binary	ANY32
(s2)	Device number where the write data is stored	-32768 to 32767	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

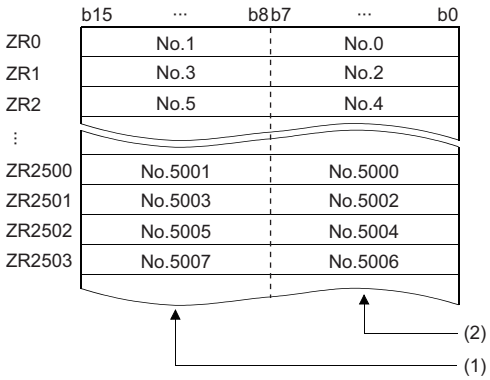
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	○	○	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—

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.

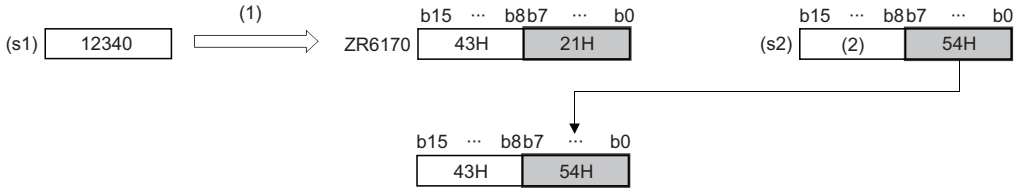


(1) Data areas when an odd number is specified

(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.

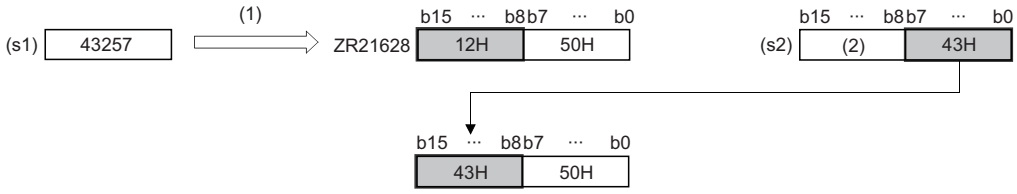


(1) Specifying the write-target area

(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.

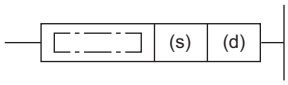
13.4 Indirect Address Read Instructions

Reading the indirect address

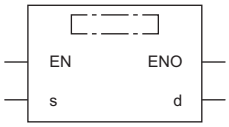
ADRSET(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions read the indirect address of the specified device.

Ladder	ST
	ENO:=ADRSET(EN,s,d); ENO:=ADRSETP(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ADRSET	
ADRSETP	

Setting data

■Descriptions, ranges, and data types

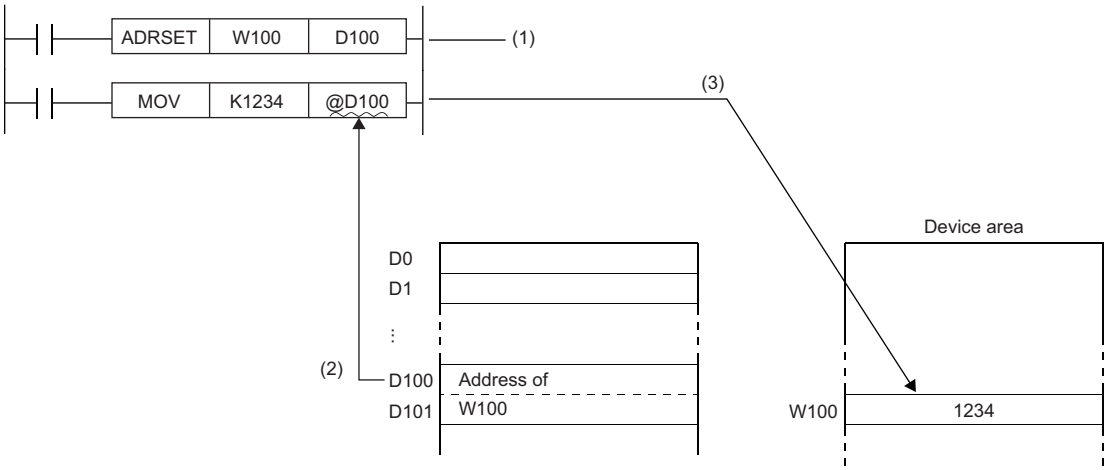
Operand	Description	Range	Data type	Data type (label)
(s)	Number of the device from which the indirect address is to be read	—	Device name	ANY_ELEMENTARY
(d)	Start number of the device for storing the indirect address of the device specified by (s)	—	32-bit signed binary	ANY32
EN	Execution condition	—	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□\□, U3E□\(H)G□	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.

MELSEC iQ-R CPU Module User's Manual (Application)

- 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.

14 TIMER, COUNTER

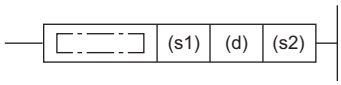
14.1 Special Counter Instructions

Counting up or down the current value (1-phase input)

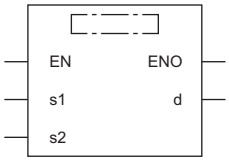
UDCNT1

RnCPU | RnENCPU | RnPCPU (Process) | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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	—	Bit	ANYBIT_ARRAY*1 (Number of elements: 2)
	(s1)+1: Count up/down flag Off indicates count-up (counting up the current value). On indicates count-down (counting down the current value).			
(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

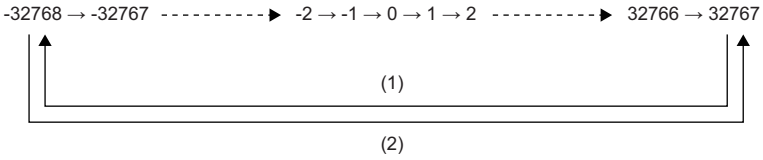
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○*1	—	—	—	—	—	—	—	—	—	—	—
(d)	—	—	○*2	—	—	—	—	—	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—

*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.



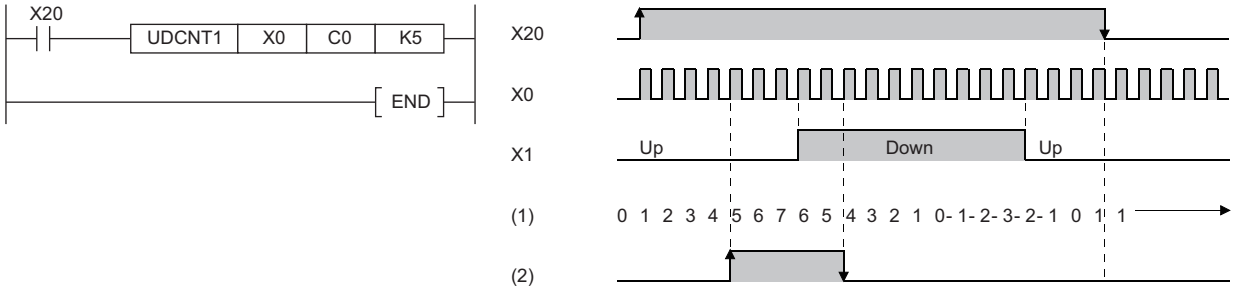
(1) When counting up

(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 X0 turns off and on after X20 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 CPU module, and performs the actual count operation using system interrupts. (The device data stored in the work area of the CPU module 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 CPU module. The interrupt interval of the CPU module 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

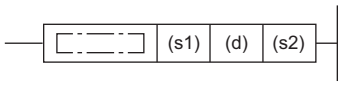
There is no operation error.

Counting up or down the current value (2-phase input)

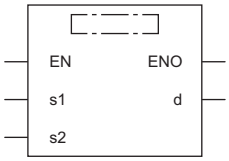
UDCNT2

RnCPU | RnENCPU | RnPCPU (Process) | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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)	—	Bit	ANYBIT_ARRAY*1 (Number of elements: 2)
	(s1)+1: Count input number (phase B pulse)			
(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.

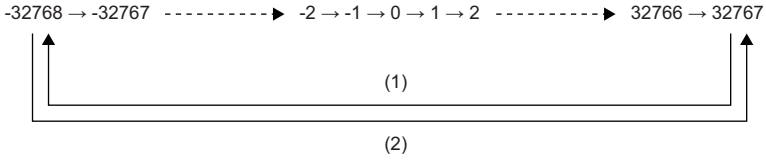
■Applicable devices

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□\□, U3E□\(H)G□	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.



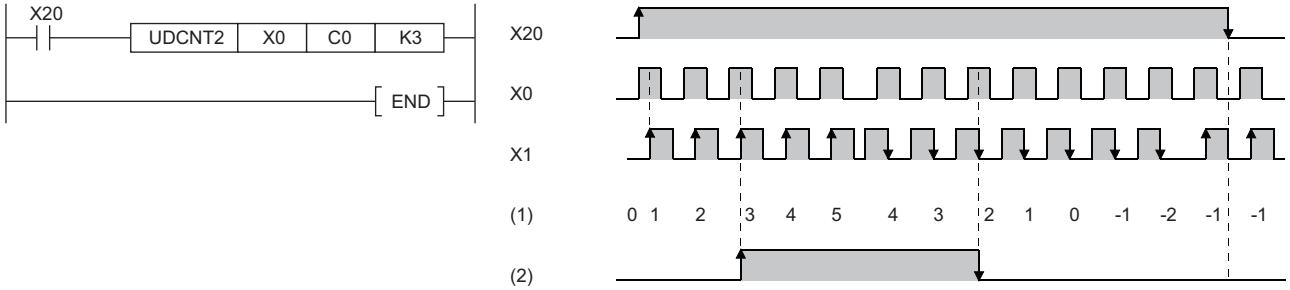
(1) When counting up

(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 X0 and X1 after X20 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 CPU module, and performs the actual count operation using system interrupts. (The device data stored in the work area of the CPU module 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 CPU module. The interrupt interval of the CPU module 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.

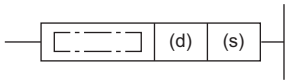
14.2 Special Timer Instructions

Teaching timer

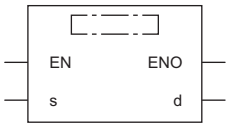
TTMR

RnCPU | RnENCPU | RnPCPU (Process) | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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: Device for the system of CPU module			
(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□\□, U3E□\(H)G□	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	Device for the system of CPU module	—	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 CPU module. 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.

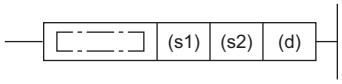
Special function timer

STMR

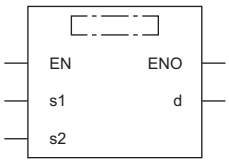
RnCPU | RnENCPU | RnPCPU (Process) | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○*1	—	—	—	—	—	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	—	—	—	—	—	—	—	—	—	—

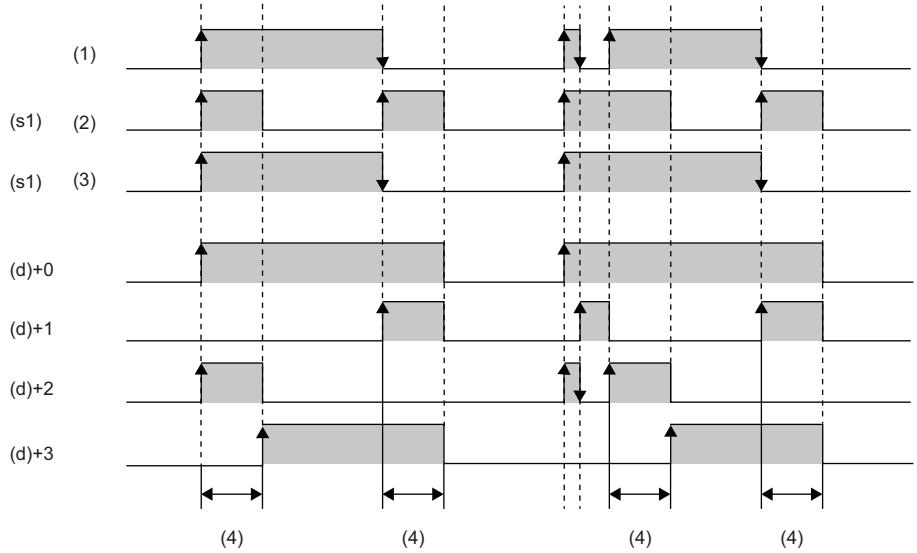
*1 Only T can be used.

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.
📖 MELSEC iQ-R CPU Module User's Manual (Application)

Operation error

There is no operation error.

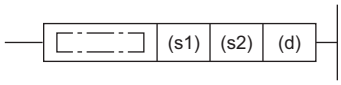
14.3 Pulse Related Instructions

Measuring the density of pulses

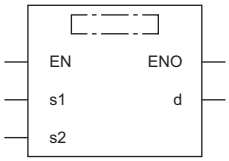
SPD

RnCPU | RnENCPU | RnPCPU (Process) | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This instruction counts the device input only for the specified time.

Ladder	ST
	ENO:=SPD(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SPD	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Pulse input device number	—	Bit	ANY_BOOL*1
(s2)	Measurement time, or the device number of the device containing the measurement time (unit: ms)	-32768 to 32767	16-bit signed binary	ANY16
(d)	Device for storing the measurement result	—	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.

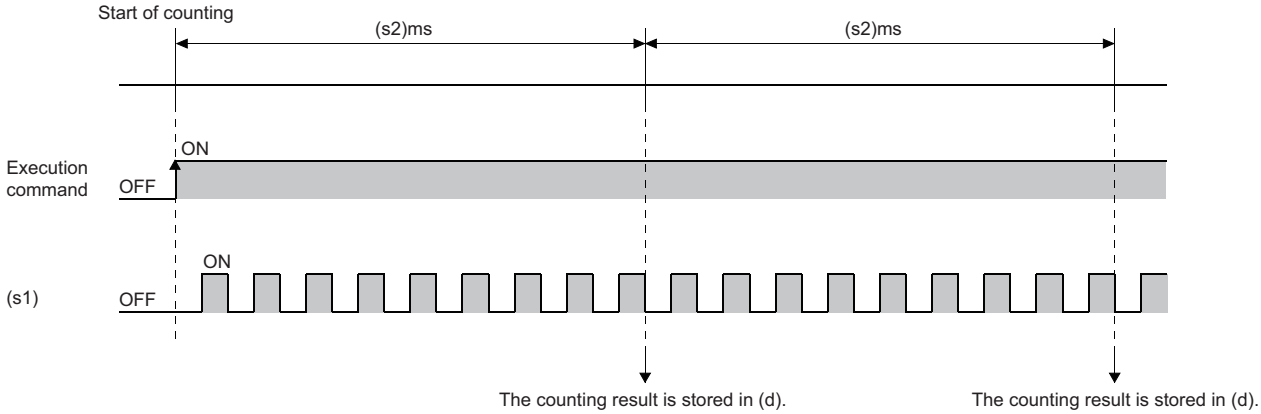
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, 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)	○*1	—	—	—	—	—	—	—	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

*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).

Processing details

- This instruction counts the number of times the input of the device specified by (s1) turns on for the duration specified by (s2), and stores the count result in the device specified by (d).



- Upon completion of measurement, the SPD instruction starts measurement from 0 again. To stop measurement by the SPD instruction, turn off the execution command.
- If the value specified in (s2) is 0, no processing is performed.

Operation error

There is no operation error.

Point

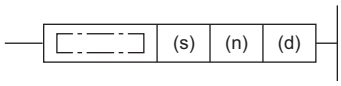
- The SPD instruction stores the data of the argument device in the work area of the CPU module, and performs the actual count operation using system interrupts. (The device data stored in the work area of the CPU module is cleared by turning off the execution command or setting it to STOP then RUN.) For this reason, the pulses that can be counted must have longer on/off time than the interval of the CPU module. The interrupt interval of the CPU module is 1ms.
- The SPD instruction can be used a maximum of six times in all running programs. The seventh or subsequent UDCNT1 instruction, if issued, causes no processing.
- The set value cannot be changed during measurement by the SPD instruction (while the command input is on). To change the set value, turn off the command input in advance.

Outputting pulses at regular intervals

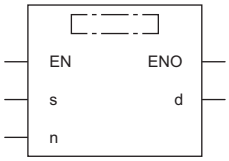
PLSY

RnCPU | RnENCPU | RnPCPU (Process) | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This instruction outputs the pulses of the specified frequency to the output module.

Ladder	ST
	ENO:=PLSY(EN,s,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
PLSY	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Frequency, or the start number of the device containing the frequency	1 to 100	16-bit signed binary	ANY16
(n)	Number of outputs, or the start number of the device containing the number of outputs	0 to 65535	16-bit unsigned binary	ANY16
(d)	Device used for pulse output	—	Bit	ANY_BOOL*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to device (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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○*1	—	—	—	—	—	—	—	—	—	—	—

*1 Only Y 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).

Processing details

- This instruction outputs the pulses at the frequency specified by (s), by the number of times specified by (n), to the output module with the output number (Y) in the device specified by (d).
- A frequency from 1Hz to 100Hz can be specified in (s). If the specified value in (s) is not in the range from 1 to 100, no processing is performed.
- A value from 0 to 65535 (0000H to FFFFH) can be specified for the number of outputs in (n). If 0 is specified in (n), pulses are output continuously.
- Only the output number (Y) corresponding to the output module can be specified for the pulse output in (d).
- The PLSY instruction starts pulse output on the rising edge of the command. When the command turns off, the PLSY instruction stops pulse output.

Operation error

There is no operation error.

Point

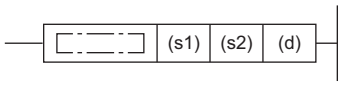
- The PLSY instruction stores the device data of the argument in the work area of the CPU module, and performs the actual output operation using system interrupts. (The device data stored in the work area of the CPU module is cleared by turning off the execution command or setting it to STOP then RUN.) For this reason, the pulses that can be out must have longer on/off time than the interval of the CPU module. The interrupt interval of the CPU module is 1ms.
- Do not change the argument of the PLSY instruction during pulse output by the instruction (the execution command is on). To change the argument, turn off the execution command in advance.
- The PLSY instruction can be used only once in all programs running in the CPU module. The second or subsequent PLSY instruction, if issued, causes no processing.

Performing the pulse width modulation

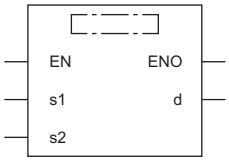
PWM

RnCPU | RnENCPU | RnPCPU (Process) | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

When on continues for the specified time, this instruction outputs the pulse of the period to the output module.

Ladder	ST
	ENO:=PWM(EN,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
PWM	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	On time, or the start number of the device containing the on time (Unit: ms)	1 to 65535	16-bit unsigned binary	ANY16
(s2)	Period, or the start number of the device containing the period (Unit: ms)	1 to 65535	16-bit unsigned binary	ANY16
(d)	Pulse output device number	—	Bit	ANY_BOOL*1
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 Only labels assigned to device (Y) can be used.

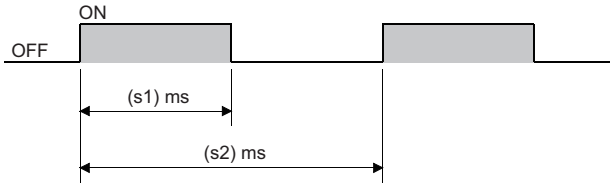
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	○	○	○	○	—	—	○	○	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○*1	—	—	—	—	—	—	—	—	—	—	—

*1 Only Y 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).

Processing details

- This instruction outputs the on time specified by (s1) and the pulse of the period specified by (s2) to the output module specified by (d).



- A value in the range from 1 to 65535 (0001H to FFFFH) can be specified in (s1) and (s2). (The value specified in (s1) must be less than the value specified in (s2).)
- No processing is performed in the following cases.
  - (s1) and (s2) are 0.
  - (s1)≥(s2)
  - The PWM instruction is executed more than once.

Operation error

There is no operation error.

Point

- The PWM instruction stores the device data of the argument in the work area of the CPU module, and performs the actual output operation using system interrupts. (The device data stored in the work area of the CPU module is cleared by turning off the execution command or setting it to STOP then RUN.) The interrupt interval of the CPU module is 1ms. The PWM instruction can be used only once in all programs running in the CPU module.
- Do not change the argument of the PWM instruction during pulse output by the instruction (the execution command is on). To change the argument, turn off the execution command in advance.

15 SHORTCUT CONTROL

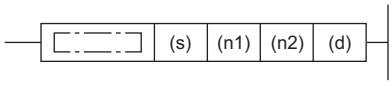
15.1 Shortcut Control Instruction

Rotary table shortest direction control

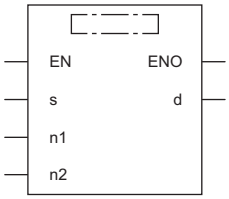
ROTC

RnCPU | RnENCPU | RnPCPU (Process) | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This instruction controls shortcut rotation on the rotary table divided equally by the specified value.

Ladder	ST
	ENO:=ROTC(EN,s,n1,n2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
ROTC	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	(s)+0: Device for measuring the number of table rotations (reserved for the system)	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
	(s)+1: Call counter number		16-bit unsigned binary	
	(s)+2: Call item number		16-bit unsigned binary	
(n1)	Number of table divisions	2 to 65535	16-bit unsigned binary	ANY16
(n2)	Number of low-speed sections	0 to less than (n1)	16-bit unsigned binary	ANY16
(d)	(d)+0: Phase A input signal	—	Bit	ANYBIT_ARRAY (Number of elements: 8)
	(d)+1: Phase B input signal			
	(d)+2: 0-point detection input signal			
	(d)+3: High-speed forward rotation output signal (reserved for the system)			
	(d)+4: Low-speed forward rotation output signal (reserved for the system)			
	(d)+5: Stop output signal (reserved for the system)			
	(d)+6: Low-speed reverse rotation output signal (reserved for the system)			
	(d)+7: High-speed reverse rotation output signal (reserved for the system)			
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	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(n1)	○	○	○	○	○	—	—	○	○	—	—	—
(n2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	—	—	—	—	—	—	—	—	—	—

Processing details

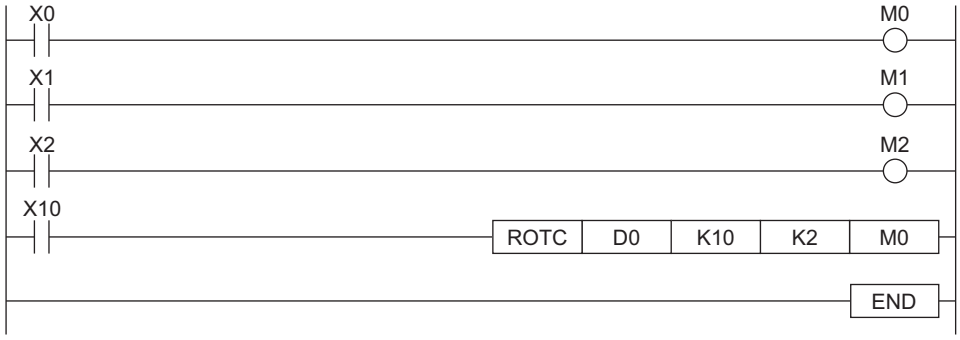
- This instruction controls the rotation of the rotary table divided equally by the value specified by (n1) so that it rotates at short cut to the position of the counter number specified by (s)+1 to get in and out the item of the number specified by (s)+2.
- It performs control by assuming that the item numbers and counter numbers are assigned counterclockwise.
- (s)+0 is the counter used for the system to count items to determine which item is in the 0th counter. Do not rewrite the data with the program. Otherwise, accurate control cannot be performed.
- The value specified in (n2) must be less than the number of table divisions specified by (n1).
- (d)+0 and (d)+1 are the phase A input signal and phase B input signal used to detect the forward and reverse rotations of the rotary table. The direction of rotation is determined by whether phase B is on the rising or falling edge when phase A is on.
  - Phase B is on the rising edge: Forward rotation (clockwise)
  - Phase B is on the falling edge: Reverse rotation (counterclockwise)
- (d)+2 is the 0-point detection signal that turns on when the 0th item reaches the 0th counter. When the device specified by (d)+2 turns on during execution of the ROTC instruction, the device specified by (s)+0 is cleared. Start shortcut control with the ROTC instruction after performing this clearing operation.
- (d)+3 to (d)+7 are output signals for controlling table operations. One of the output signals in (d)+3 to (d)+7 is turned on according to the execution result of the ROTC instruction.
- When the command of the ROTC instruction is off, shortcut control is not performed and (d)+3 to (d)+7 are all turned off.
- The ROTC instruction can be used only once in all running programs. If it is used more than once, normal operation cannot be performed.
- If the value in (s)+0 to (s)+2 or (n2) is greater than (n1), no processing is performed.

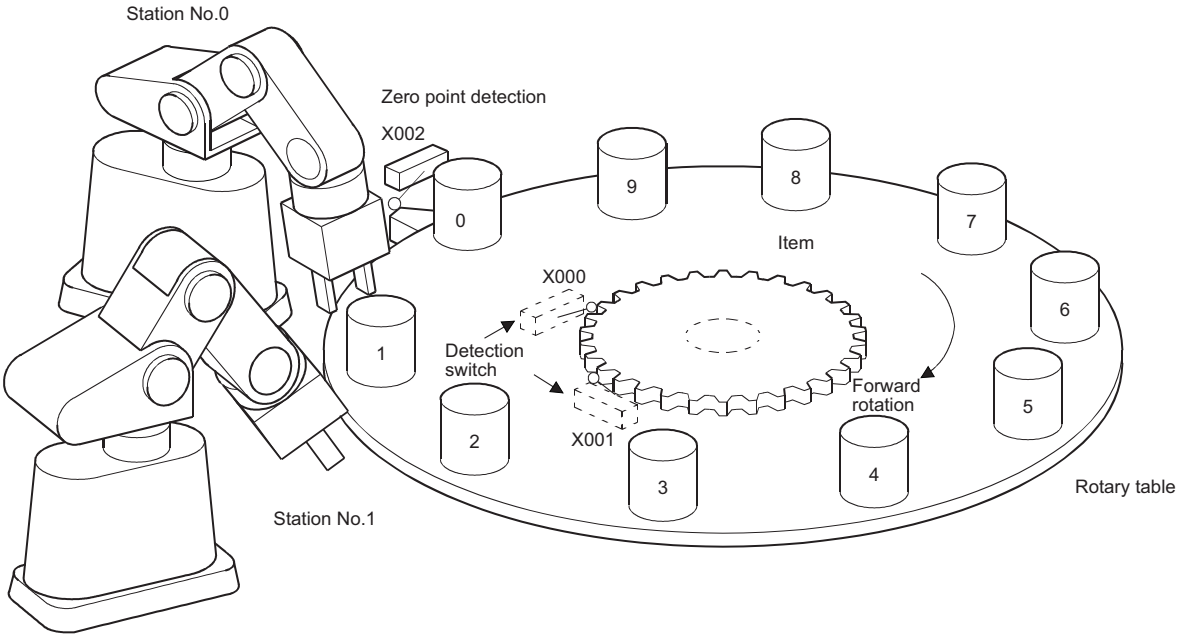
■Program example

When getting in and out at counters, the items placed on the rotary table divided into 10 sections, this program controls the rotary table so that the items rotate at short cut.

The item number is specified by D2, and the counter number is specified by D1.

The table is rotated at low speeds in two front and rear sections.





Operation error

There is no operation error.

16 RAMP SIGNAL

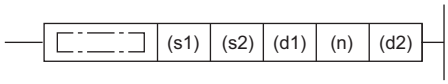
16.1 Ramp Signal Instruction

Ramp signal

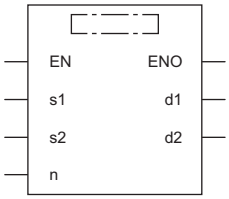
RAMPQ

RnCPU | RnENCPU | RnPCPU (Process) | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This instruction shifts from a specified value to another specified value in (n) times.

Ladder	ST
	ENO:=RAMPQ(EN,s1,s2,n,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
RAMPQ	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Initial value	-32768 to 32767	16-bit signed binary	ANY16
(s2)	Last value	-32768 to 32767	16-bit signed binary	ANY16
(d1)	(d1)+0: Current value	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
	(d1)+1: Number of executions			
(n)	Number of shifts	1 to 32767	16-bit signed binary	ANY16
(d2)	(d2)+0: Completion device	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
	(d2)+1: Bit for selecting data retention at completion			
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)	○	○	○	○	○	—	—	○	○	—	—	—
(d1)	○	○	○	○	○	—	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—
(d2)	○	—	—	—	—	—	—	—	—	—	—	—

Processing details

- When the execution command is on, this instruction performs processing as follows.
  - Shifting from the value specified by (s1) to the value specified by (s2) in the number of times specified by (n).
  - For (n), specify the number of scans (number of shifts) to be performed to shift from (s1) to (s2). If the value specified in (n) is out of the range between 0 and 32768, no processing is performed.
  - (d1)+1 is used for the system to store the number of times the RAMPQ instruction has been executed.
  - The change value per scan is calculated by the following equation.

$$Cv = \frac{(s2) - (s1)}{(n)}$$

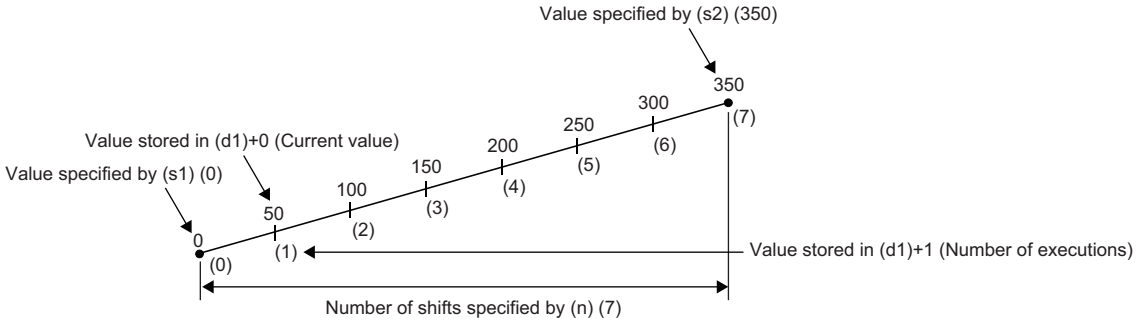
Cv: Amount of change in 1 scan

(s2): Value specified by (s2)

(s1): Value specified by (s1)

(n): Value specified by (n)

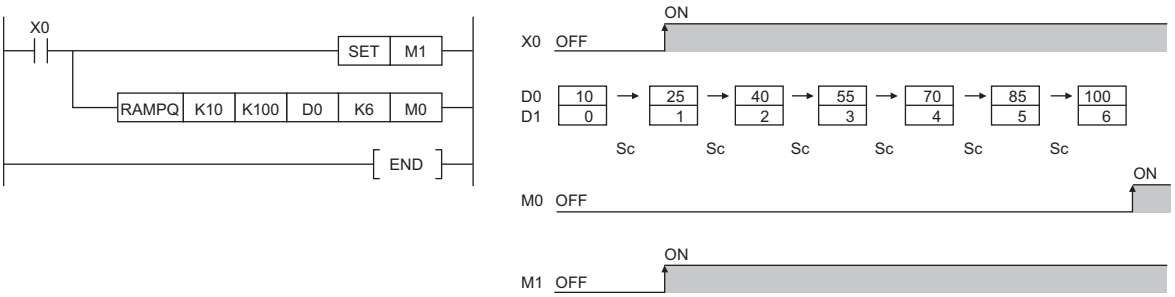
- The following figure shows how to change values from 0 to 350 in seven scans.



- If the change value in one scan is indivisible, correct it so that it becomes the value specified by (s2) in the number of shifts specified by (n). For this reason, a linear ramp may not be created.
- The following figure shows the operation of processing using the RAMPQ instruction.

Ex.

Program which, when X0 turns on, changes the content of D0 from 10 to 100 in six scans and holds the content of D0 when the change is completed



Sc: 1 scan

- After scanning is performed the number of shifts specified by (n), the completion device specified by (d2)+0 turns on. The on/off status of the completion device and the data in (d1)+0 are determined by on/off of the device specified by (d2)+1. When (d2)+1 is off, the RAMPQ instruction turns off (d2)+0 in the next scan and restarts shifting from the initial value. When (d2)+1 is on, (d2)+0 is kept on and the data in (d1)+0 remains unchanged.
- If the command turns off during execution of the RAMPQ instruction, the data in (d1)+0 will not change thereafter. When the command turns on again, the RAMPQ instruction restarts shifting from the initial value.
- Do not change the values in (s1) and (s2) before the completion device specified by (d2)+0 turns on. The value to be stored in (d1)+1 is calculated using the same calculation formula every scan, and therefore changing the values in (s1) and (s2) may result in a sudden change.
- When making the digit specification using a bit device in (d1), specify the digit in K4 format.

Precautions

When the digit specification is made using a bit device in (d1), it is acceptable only when the digit is specified in K4.

Operation error

There is no operation error.

17 MATRIX INPUT

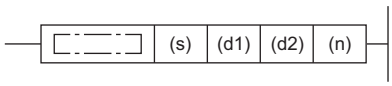
17.1 Matrix Input Instruction

Matrix input

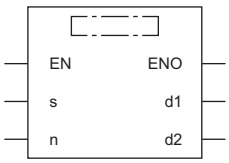
MTR

RnCPU | RnENCPU | RnPCPU (Process) | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This instruction sequentially reads the input of 16 points×n columns connected to the specified input number and after.

Ladder	ST
	ENO:=MTR(EN,s,n,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
MTR	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Start device of input	—	Bit	ANY_BOOL*1*3
(d1)	Start device of output	—	Bit	ANY_BOOL*2*3
(d2)	Start device for storing the matrix input data	—	Bit	ANY_BOOL*3
(n)	Number of input columns	2 to 8	16-bit unsigned 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 (Y) can be used.
*3 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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	—	—	—	—	—	—	—	—	—	—	—
(d1)	○*2	—	—	—	—	—	—	—	—	—	—	—
(d2)	○	—	—	—	—	—	—	—	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

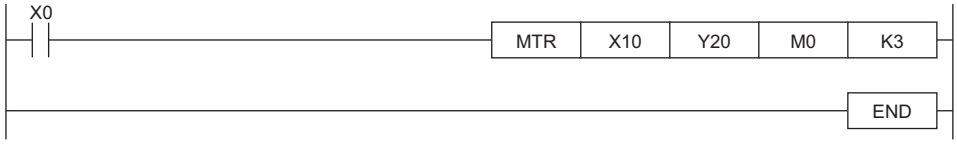
*1 Only X can be used.
*2 Only Y can be used.

Processing details

- This instruction sequentially reads the input of 16 points×n columns connected to the (s) specified input number and after, and stores the input data that has been read in the device specified by (d2) and later.
- One scan reads one column (16 points) of data.
- The instruction sequentially repeats the reading of data from column 1 to column (n).
- In the device specified by (d2) and later, the data in column 1 is stored in the 16 points from the start and the data in column 2 is stored in the next 16 points. For this reason, the MTR instruction occupies 16×(n) points from the device specified by (d2).
- (d1) is the output for selecting the column to be read and is turned on and off automatically by the system. The (n) points from the device specified by (d1) is used.
- Only a device number which is a multiple of 16 can be specified in (s), (d1), and (d2).
- A value from 2 to 8 can be specified in (n).
- No processing is performed in the following cases.
  - The device number specified by (s), (d1), or (d2) is not a multiple of 16.
  - The device specified by (s) is outside the range of actual inputs.
  - The device specified by (d1) is outside the range of actual outputs.
  - In the device specified by (d2) and later, 16×(n) points of data is outside the range of the relevant device.
  - (n) is outside the range from 2 to 8.

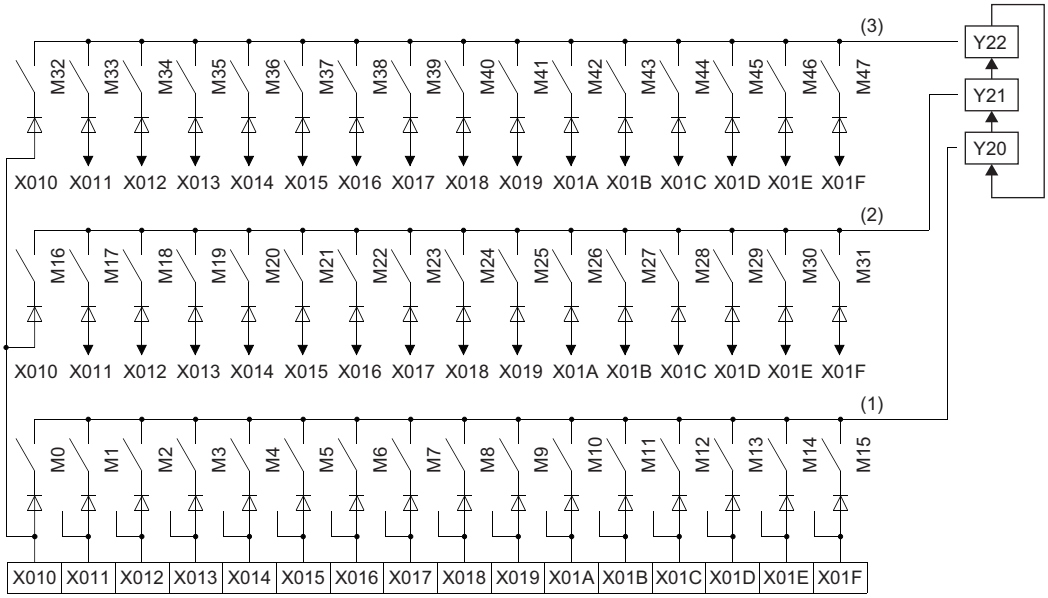
■Program example

A program that reads the matrix data (16 points × three columns) connected to X10 and later, and stores the read data to M0 and later when X0 turns on



[Operation]

- When Y20 turns on, the instruction reads the input signals of the 1st column, and stores the read data to M0 to M15.
- When Y21 turns on, the instruction reads the input signals of the 2nd column, and stores the read data to M16 to M31.
- When Y22 turns on, the instruction reads the input signals of the 3rd column, and stores the read data to M32 to M47.



(1) 1st column

(2) 2nd column

(3) 3rd column

Precautions

- Note that the MTR instruction directly operates the actual input/output. Even when the command of the MTR instruction turns off, the output that has been turned on by the MTR instruction is not turned off. Turn off the output specified by (d1) in the program.
- The MTR instruction execution interval should be longer than the total response time of the input and output modules. If the MTR instruction execution interval is shorter than the above time, inputs cannot be read normally. If the scan time in the program is short, select the constant scan and set longer scan time than the total of the response time.

Operation error

Error code (SD0)	Description
2820H	A device other than the input (X) is specified by (s).
	A device other than the output (Y) is specified by (d1).

18 CPU MODULE DATABASE ACCESS FUNCTION

18.1 Database Access Instructions

The database access instructions add, update, obtain, or delete data with respect to the tabular data such as product and production information managed as databases in the programmable controller.
These instructions construct a database from the Unicode text file that defines information such as a table configuration, and operates the database thus constructed. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

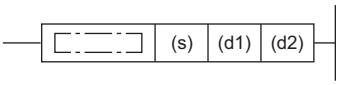
Importing data to the data base

DBIMPORT(P)

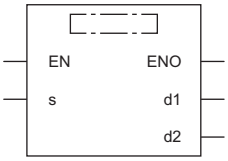
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- These instructions cannot be used for the R00CPU, R01CPU, and R02CPU.

These instructions import the data stored in the Unicode text file at the path specified by (s) and construct a database.

Ladder	ST
	ENO:=DBIMPORT(EN,s,d1,d2); ENO:=DBIMPORTP(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBIMPORT	
DBIMPORTP	

Setting data

■Descriptions, ranges, and data types

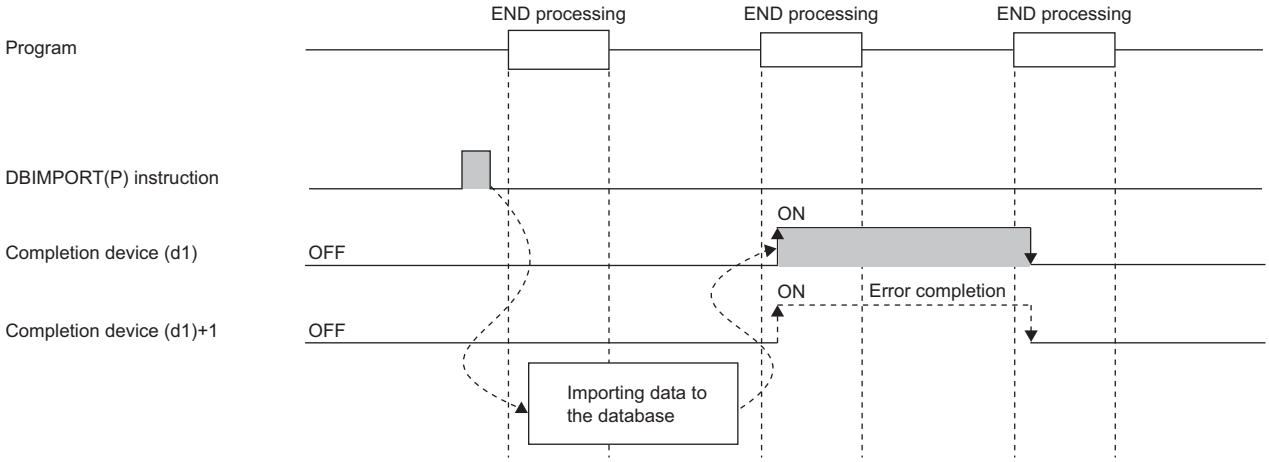
Operand	Description	Range	Data type	Data type (label)
(s)	Start device for storing the Unicode text file name Example: "2:\database1\recipe\recipe_db.txt" Within 255 characters	—	Unicode string	ANYSTRING_DOUBLE
(d1)	Completion device (start device that turns on for one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Normal completion • Other than 0000: Error completion (error code)	—	Word	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	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions import the data stored in the Unicode text file specified by (s) and construct a database. Information such as a table configuration needs to be defined in advance in the Unicode text file used by the DBIMPORT(P) instruction. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- When the database that is already open exists and the DBIMPORT(P) instruction is executed, it is completed with an error.
- Upon successful completion, the completion signal in the completion device (d1)+0 is turned on and 0 is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBIMPORT(P) instruction.



- SM753 (File being accessed) turns on while the DBIMPORT(P) instruction is being executed.*1 While SM753 is on, the DBIMPORT(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBIMPORT(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.*1 If executed, no processing is performed.
- When DBIMPORT(P) instruction is executed, if a database with the name specified in the Unicode text file specified by (s) exists on the same path as the Unicode text file, the database with the same name on the SD memory card (in the database folder) is deleted and a new database is created with the name specified in the Unicode text file.

*1 For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBIMPORT(P) instruction is executed during execution of the database access instruction.
- The Unicode text file specified by (s) does not exist.
- The number of fields specified in the field name row of the Unicode text file does not match the number of fields in the record row.
- A table definition start tag or end tag is missing in the Unicode text file.
- An out-of-range value is set for the key constraint in the Unicode text file.
- The database name, table name, or field name in the Unicode text file exceeds 32 characters.
- The number of tables or fields in the Unicode text file exceeds the maximum number.
- An out-of-spec data type is specified in the Unicode text file.
- The number of records in the Unicode text file exceeds the maximum number (for a programmable controller CPU with firmware version earlier than "28").
- An access to the database has failed.
- The database name contains an invalid character.
- The total number of characters used in the database name specified in the Unicode text file and those used in the folder path (including the drive path character) specified by (s) exceeds 128.
- The database that is already open exists and the DBIMPORT(P) instruction is executed.
- The number of characters on the comment line exceeds the maximum number.
- A character other than those that can be represented as ASCII codes (0020H to 007EH) is used in the folder name of the Unicode text file specified by (s).
- The database name in the Unicode text file exists on the SD memory card and the database is used by another function when the DBIMPORT(P) instruction is executed.

If an error is detected because of the Unicode text format, the DBIMPORT(P) instruction turns on the error termination signal in (d1)+1 and stores the Unicode text line where an error was detected in SD760 and SD761.

Operation error

Error code (SD0)	Description
2840H	A numerical value other than 2 is specified for the drive number in (s).
3405H	The character string (path) in the device specified by (s) exceeds 255 characters.

For the error code stored in the completion status of the operand, refer to the following.

☞ Page 1086 Error codes related to database access instructions

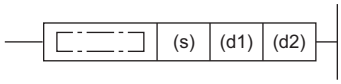
Exporting data from the data base

DBEXPORT(P)

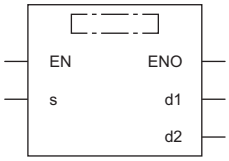
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- These instructions cannot be used for the R00CPU, R01CPU, and R02CPU.

These instructions export the data stored in the specified database to the Unicode text file.

Ladder	ST
	ENO:=DBEXPORT(EN,s,d1,d2); ENO:=DBEXPORTP(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBEXPORT	
DBEXPORTP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s)	Start device for storing the database folder path Example: "2:\database1\recipe" Within 128 characters	—	Unicode string	ANYSTRING_DOUBLE
(d1)	Completion device (start device that turns on for one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Normal completion • Other than 0000: Error completion (error code)	—	Word	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	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

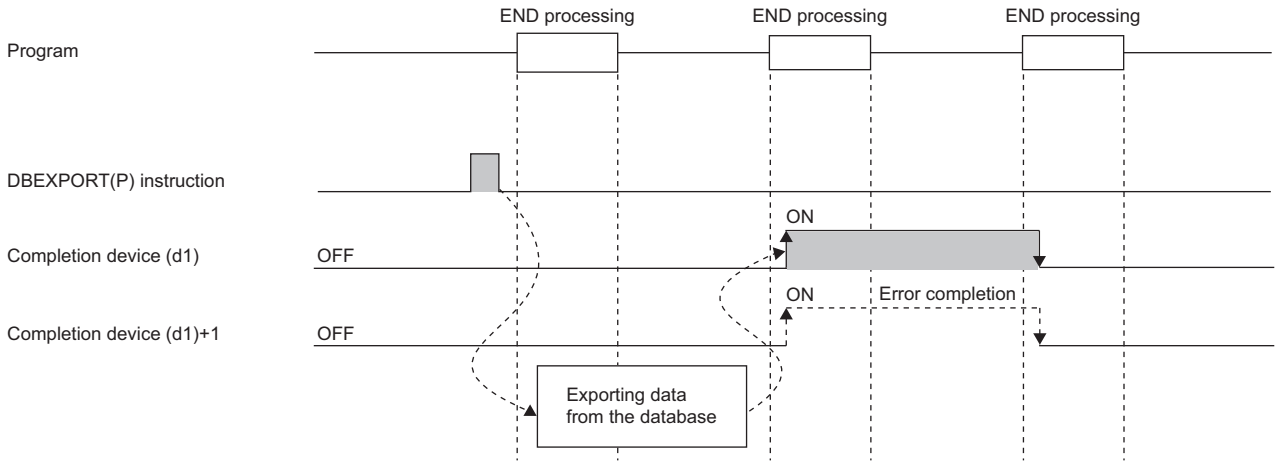
Processing details

- These instructions export the data in the database stored in the database folder at the path specified by (s) to the Unicode text file.
- The Unicode text file is created in the folder where the database folder is stored. The file name is "database_name.txt". If the same Unicode text file already exists, the file is overwritten with the exported data.

Ex.

When the path of the database folder is "2:\database\recipe1", executing the instruction creates Unicode text file "2:\database\recipe1.txt" and exports data to the file.

- Upon successful completion, the completion signal in the completion device (d1)+0 is turned on and 0 is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBEXPORT(P) instruction.



- The internal configuration of the Unicode text file to which data is exported is the same as the file to which data is imported by the DBIMPORT(P) instruction. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- SM753 (File being accessed) turns on while the DBEXPORT(P) instruction is being executed.*1 While SM753 is on, the DBEXPORT(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBEXPORT(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.*1 If executed, no processing is performed.
- During transaction, the DBEXPORT(P) instruction cannot be executed.

*1 For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBEXPORT(P) instruction is executed during execution of the database access instruction.
- The path specified by (s) is not a database.
- Writing data to the Unicode text failed due to the failure to access the database.
- The number of characters of the path (including the drive path character) specified by (s) exceeds 128.
- A database built or operated with a programmable controller CPU with firmware version "28" or later is specified in (s) for a programmable controller CPU with firmware version earlier than "28".
- The DBEXPORT(P) instruction is executed during transaction.

Operation error

Error code (SD0)	Description
2840H	A numerical value other than 2 is specified for the drive number in (s).

For the error code stored in the completion status of the operand, refer to the following.

☞ Page 1086 Error codes related to database access instructions

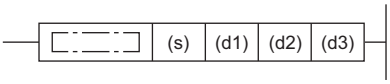
Opening the data base

DBOPEN(P)

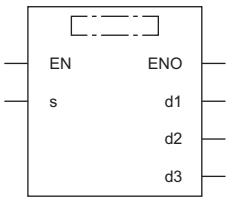
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- These instructions cannot be used for the R00CPU, R01CPU, and R02CPU.

These instructions connect to the database specified by (s) and make it available.

Ladder	ST
	ENO:=DBOPEN(EN,s,d1,d2,d3); ENO:=DBOPENP(EN,s,d1,d2,d3);

FBD/LD



■Execution condition

Instruction	Execution condition
DBOPEN	
DBOPENP	

Setting data

■Descriptions, ranges, and data types

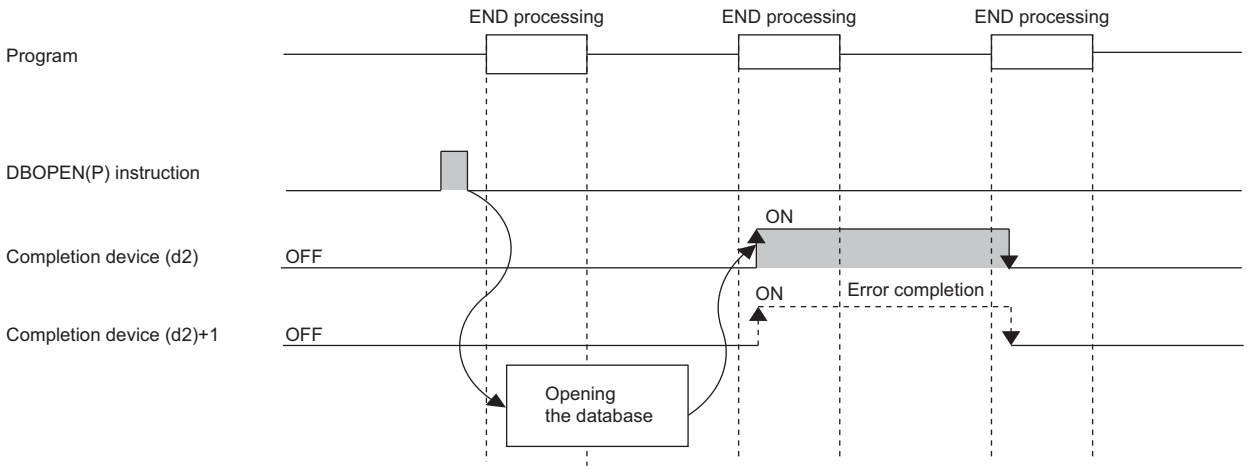
Operand	Description	Range	Data type	Data type (label)
(s)	Start device for storing the folder path of the database to be opened. Example: "2:\database1\recipe" Within 128 characters	—	Unicode string	ANYSTRING_DOUBLE
(d1)	Database identification number	1 to 4	16-bit signed binary	ANY16
(d2)	Completion device (start device that turns on for one scan upon completion of instruction) • (d2)+0: Completion signal • (d2)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d3)	Completion status • 0000: Normal completion • Other than 0000: Error completion (error code)	—	Word	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	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—
(d3)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions connect to the database stored in the folder path specified by (s) and makes it available.
- Specify "(drive number):(database folder path)" for the storage location. The drive number is fixed to 2 (SD memory card).
- Upon normal completion, the instruction stores the identification number of the connected database in the device (d1). The completion signal in the completion device (d2)+0 is turned on and 0 is stored as the completion status in the device (d3).
- The following figure shows the operation of the completion device at completion of the DBOPEN(P) instruction.



- The DBOPEN(P) instruction enables connections to a maximum of four different databases at the same time.
- During transaction, the database cannot be newly opened.
- SM753 (File being accessed) turns on while the DBOPEN(P) instruction is executed.*1 While SM753 is on, the DBOPEN(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBOPEN(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.*1 If executed, no processing is performed.

*1 For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error termination signal in (d2)+1 is turned on and an error code is stored in (d3).

- The DBOPEN(P) instruction is executed during execution of the database access instruction.
- The storage location specified by (s) does not exist.
- An attempt is made to connect to the database that has already been connected.
- The DBOPEN(P) instruction is executed for a database exceeding the maximum number of databases that can be connected concurrently.
- The number of characters of the path (including the drive path character) specified by (s) exceeds 128.
- The database is opened during transaction.

Operation error

Error code (SD0)	Description
2820H	The area specified by (d2) exceeds the applicable range of the device/label used.
2840H	A numerical value other than 2 is specified for the drive number in (s).

For the error code stored in the completion status of the operand, refer to the following.

☞ Page 1086 Error codes related to database access instructions

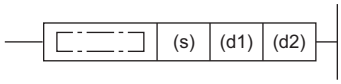
Closing the data base

DBCLOSE(P)

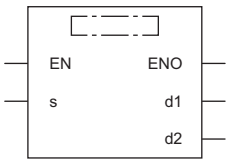
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- These instructions cannot be used for the R00CPU, R01CPU, and R02CPU.

These instructions clear the connection from the specified database.

Ladder	ST
	ENO:=DBCLOSE(EN,s,d1,d2); ENO:=DBCLOSEP(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBCLOSE	
DBCLOSEP	

Setting data

■Description, range, data type

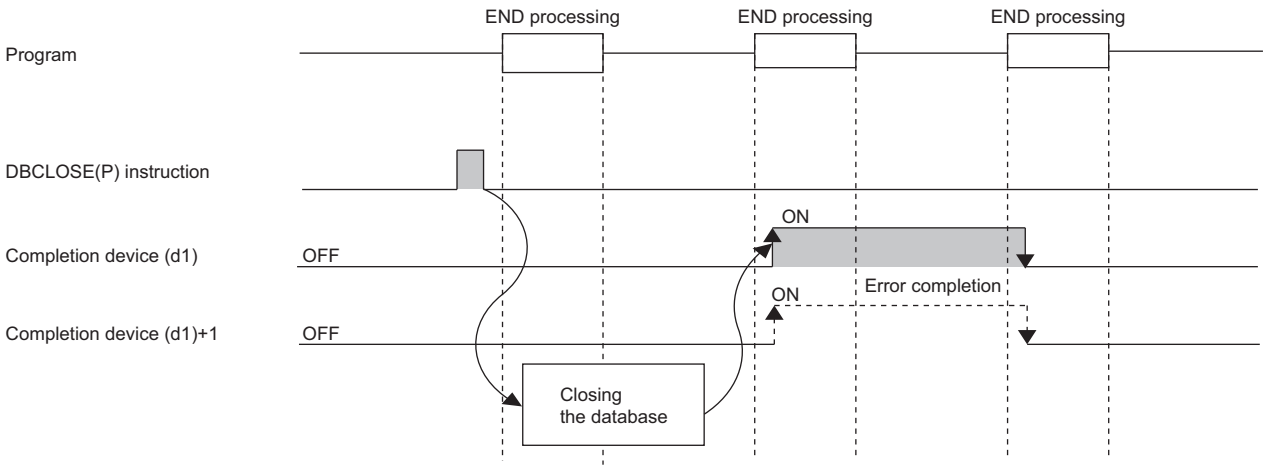
Operand	Description	Range	Data type	Data type (label)
(s)	Database identification number	1 to 4	16-bit signed binary	ANY16
(d1)	Completion device (start device that turns on for one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Normal completion • Other than 0000: Error completion (error code)	—	Word	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	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions release the connection between the database identification number specified by (s) and the corresponding database.
- Upon successful completion, the completion signal in the completion device (d1)+0 is turned on and 0 is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBCLOSE(P) instruction.



- If the DBCLOSE(P) instruction is executed before DBCOMMIT(P) or DBROLBAK(P) while the transaction is run by the DBTRANS(P) instruction, the transaction is determined in the status at the execution of the DBCLOSE(P) instruction.
- SM753 (File being accessed) turns on while the DBCLOSE(P) instruction is executed.*1 While SM753 is on, the DBCLOSE(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBCLOSE(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.*1 If executed, no processing is performed.

*1 For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBCLOSE(P) instruction is executed during execution of the database access instruction.
- The identification number specified by (s) is an already disconnected database.
- An identification number outside the setting range is specified by (s).

Operation error

Error code (SD0)	Description
2820H	The area specified by (d1) exceeds the applicable range of the device/label used.

For the error code stored in the completion status of the operand, refer to the following.

☞ Page 1086 Error codes related to database access instructions

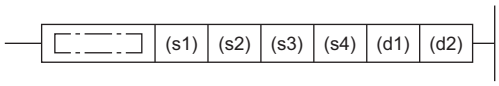
Adding a record to the data base

DBINSERT(P)

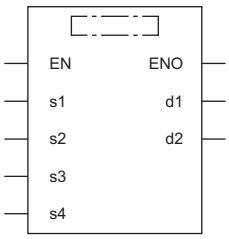
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- These instructions cannot be used for the R00CPU, R01CPU, and R02CPU.

These instructions add a record to the table of the database corresponding to the specified identification number.

Ladder	ST
	ENO:=DBINSERT(EN,s1,s2,s3,s4,d1,d2); ENO:=DBINSERTP(EN,s1,s2,s3,s4,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBINSERT	
DBINSERTP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Database identification number	1 to 4	16-bit signed binary	ANY16
(s2)	Start device for storing the database table names.*1	—	Unicode string	ANYSTRING_DOUBLE
(s3)	Start device for storing the database field names.*1	—	Word	ANY16*2
(s4)	Start device for storing insertion data	—	Word	ANY16*2
(d1)	Completion device (start device that turns on for one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Completed successfully • Other than 0000: Completed with an error (error code)	—	Word	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 The table and field names are case-sensitive.

*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

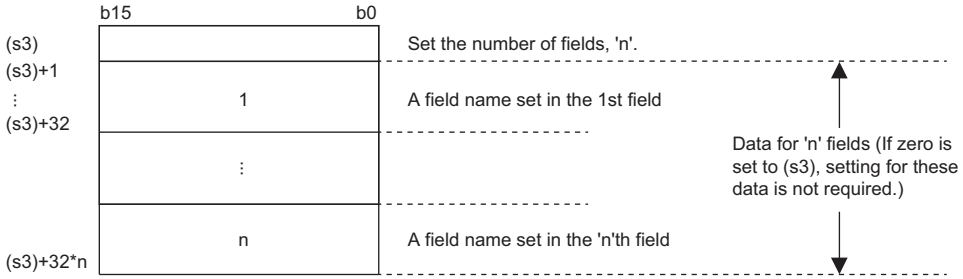
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(s4)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Database field name

Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Number of fields	Specify the number of fields to which a value is to be added. Specify a value equal to or less than the number of fields of the table specified in (s2). If 0 is specified, all fields of the table are subject to output.*1	0 to 128*2	User
+1 to +□	Field name	Specify the name of each field. Specify field names, each fixed to 32 characters, by the number of fields with Unicode character strings. For the name less than 32 characters, the character string should be right-justified and filled with 0000H to become a 32-character string. The last address □ in (s3) varies according to the number of fields. □=32×n (n: Number of fields) This setting of this item is not necessary when "0" is specified in (s3)+0. *1(Ignored if specified.)	—	User

*1 Programmable controller CPU with firmware version "28" or later supports this processing.
*2 "1" to "16" for a programmable controller CPU with firmware version earlier than "28"

The following figure shows the format of (s3).

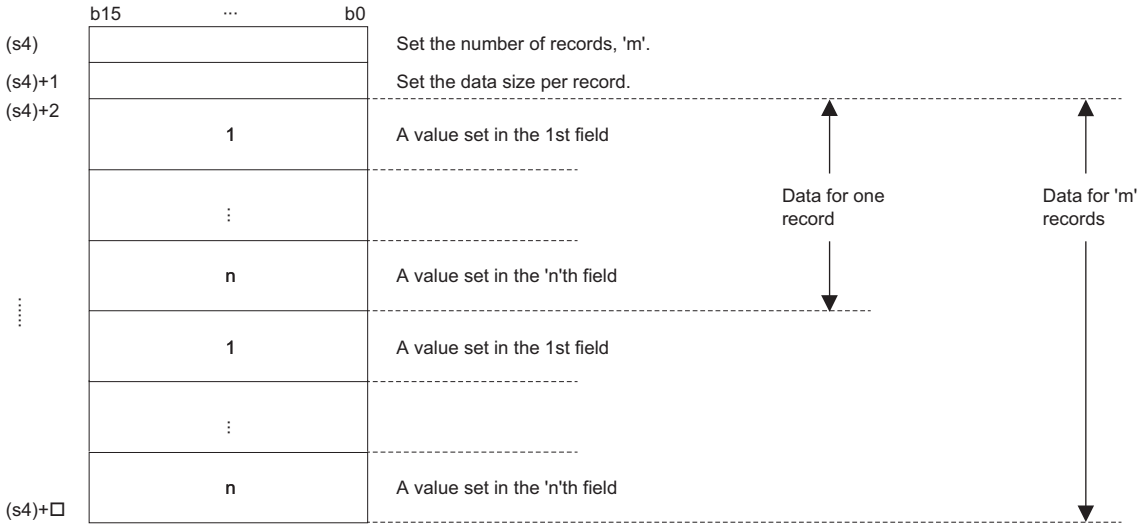


■Insertion data

Operand: (s4)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Number of records	Specify the number of records to be added. The number of records to be added is a value which satisfies the following formula:*1 Number of records (m) ≤ (32768/Data size per record)	1 to m*1	User
+1	Size	Specify the size of one record.	Depends on the data type	User
+2 to +□	Value to be added	Specify the data for the number of fields specified by (s3) for the number of records (m) specified by (s4)+0.	Depends on the data type	User

*1 "1" to "16" for a programmable controller CPU with firmware version earlier than "28"

The following figure shows the format of (s4). Set the value corresponding to each field set in (s3).



- The data size of each value follows the size of the data type of each field.

Data type of field	Data size (unit: word)
BOOL	1
WORD	1
DWORD	2
INT	1
DINT	2
REAL	2
LREAL	4
STRING	• Even number of characters: (Number of characters÷2)+1 • Odd number of characters: Rounding up the number of characters÷2 [Example] STRING: 32: (32÷2)+1=17 STRING: 15: (15÷2)=7.5 rounding up→8
WSTRING	Number of characters+1 [Example] WSTRING: 32: (32)+1=33

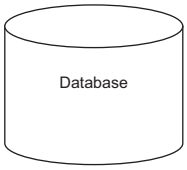
Ex.

When registering a record with ID = 0003H to the product information table (proInfo) in the database using the DBINSERT(P) instruction

[Product information table (proInfo)]

ID	Product	Size(x)	Size(y)	Size(z)
0001H	"AAA1"	80	100	60
0002H	"BBB2"	40	90	40
0003H	"CCC3"	40	80	40
⋮				

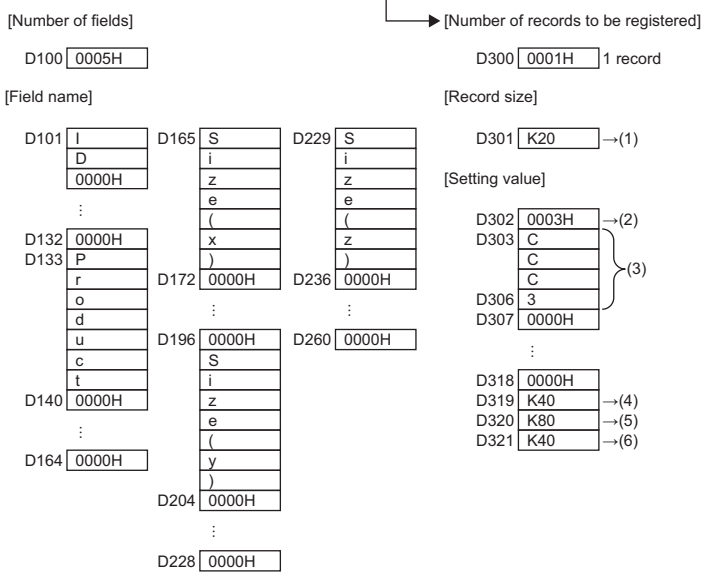
ID	WORD type
Product	WSTRING type (15 characters)
Size(x)	WORD type
Size(y)	WORD type
Size(z)	WORD type



[Program]

Table name

The program registers the information of ID 0003H by using the DBINSERT instruction.



(1) WSTRING type (16 words) + WORD type (1 word) × 4 = 20 words
(2) Setting value in field "ID"
(3) Setting value in field "Product"
(4) Setting value in field "Size(x)"
(5) Setting value in field "Size(y)"
(6) Setting value in field "Size(z)"

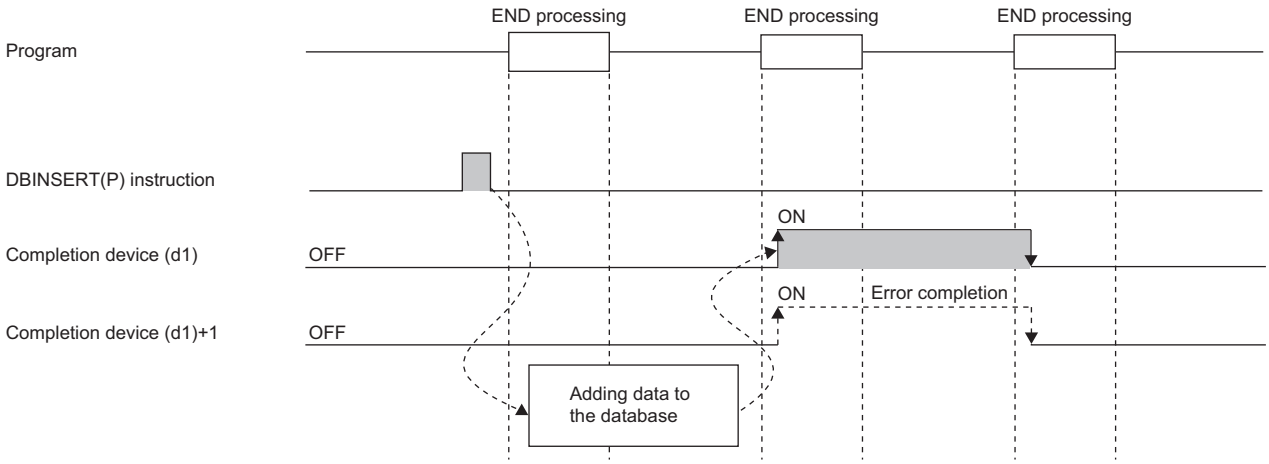
Processing details

- These instructions add a record to the table specified by (s2) in the database corresponding to the identification number specified by (s1).
- Specify the number of fields of the record to be added and field names in (s3). For the field names to be added, not all fields making up the table need to be specified. NULL is stored in a field which is not specified.
- Specify the number of records to be added and the size and value per record in (s4).
- The following figure shows an example when a record is added to the table recipeA by using the DBINSERT(P) instruction.

(1)	(2)	(3)
1	ProductA	100
2	ProductB	200

(1) Field 1: WORD
(2) Field 2: WSTRING (16 characters maximum)
(3) Field 3: INT

- Upon successful completion, the completion signal in the completion device (d1)+0 is turned on and 0 is stored as the completion status in the device (d2).
- Upon completion with an error, the error completion signal in the completion device (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBINSERT(P) instruction.



- SM753 (File being accessed) turns on while the DBINSERT(P) instruction is executed.*2 While SM753 is on, the DBINSERT(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBINSERT(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.*2 If executed, no processing is performed.

*2 For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBINSERT(P) instruction is executed during execution of the database access instruction.
- An identification number outside the range is specified by (s1).
- An identification number of a database which is not open is specified by (s1).
- A database built or operated with a programmable controller CPU with firmware version "28" or later is specified by (s1) for a programmable controller CPU with firmware version earlier than "28".
- The table name specified by (s2) does not exist.
- The number of characters of the table name specified by (s2) exceeds 32.
- An out-of-range value is specified in (s3) for the number of fields to be added.
- An out-of-range field name is set in the field name specified by (s3)+1 to (s3)+□.
- An out-of-range value is specified in (s4) for the number of records to be added.
- Database insertion processing failed.
- The range of the data for one record set in (s4)+2 does not match the size specified by (s4)+1.

Operation error

Error code (SD0)	Description
2820H	The area specified by (s2), (s3), (s4), or (d1) exceeds the applicable range of the device/label used.

For the error code stored in the completion status of the operand, refer to the following.

☞ Page 1086 Error codes related to database access instructions

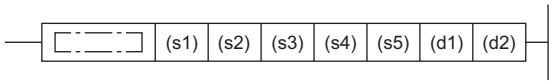
Updating the record in the data base

DBUPDATE(P)

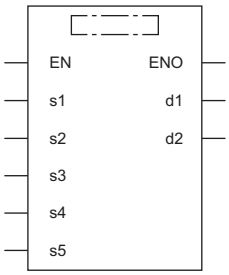
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- These instructions cannot be used for the R00CPU, R01CPU, and R02CPU.

These instructions update all records that meet the specified condition in the specified table corresponding to the specified identification number.

Ladder	ST
	ENO:=DBUPDATE(EN,s1,s2,s3,s4,s5,d1,d2); ENO:=DBUPDATEP(EN,s1,s2,s3,s4,s5,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBUPDATE	
DBUPDATEP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Database identification number	1 to 4	16-bit signed binary	ANY16
(s2)	Start device for storing the table name of the database table to be updated.*1	—	Unicode string	ANYSTRING_DOUBLE
(s3)	Start device for storing the field name of the database to be updated.*1	—	Word	ANY16*2
(s4)	Start device for storing the updated data	—	Word	ANY16*2
(s5)	Start device for storing the update conditions (a maximum of two conditions).	—	Word	ANY16*2
(d1)	Completion device (start device that turns on for one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Completed successfully • Other than 0000: Completed with an error (error code)	—	Word	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 The table and field names are case-sensitive.
*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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(s4)	—	—	○	—	—	—	—	○	—	—	—	—
(s5)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Update target field name

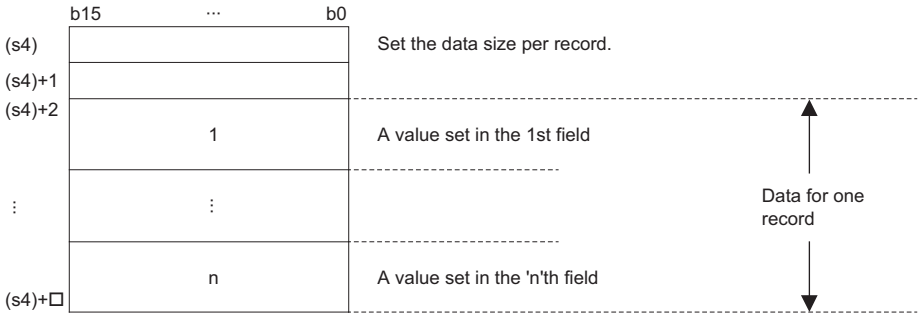
Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Number of fields	Specify the number of fields to be updated.	1 to 128*1	User
+1 to +□	Field name	Specify the name of each field. Specify field names, each fixed to 32 characters, by the number of fields with Unicode character strings. For the name less than 32 characters, the character string should be right-justified and filled with 0000H to become a 32-character string. The last address □ in (s3) varies according to the number of fields. □=32×n (n: number of fields)	—	User

*1 "1" to "16" for a programmable controller CPU with firmware version earlier than "28"

■Updated data

Operand: (s4)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Data size	Set the data size of the field to be updated.	—	User
+1	Not used	—	—	—
+2 to +□	Set value	Set the updated data.	Depends on the data type	User

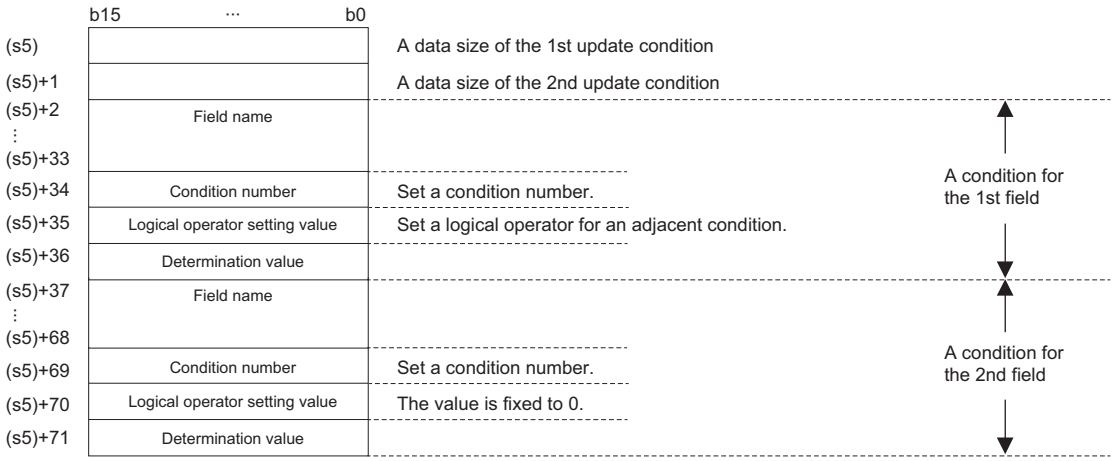
The following figure shows the setting format of (s4). Set the value corresponding to each field in (s3). The data size of each set value follows the size of the data type of each field. (☞ Page 1055 DBINSERT(P))



■Update condition

Operand: (s5)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Data size of the first condition	Set the data size of the first update condition.	1 to 125	User
+1	Data size of the second condition	Set the data size of the second update condition. • 0: No condition • Other than 0: Data size	0 to 125	User
+2 to +□	Update condition	Set the update conditions. The last address in (s5) varies depending on the data type of the determination value. (s5)+2 ⋮ (s5)+33: (1) (s5)+34: (2) (s5)+35: (3) (s5)+36 ⋮ (s5)+□: (4) ■Field name (1) Specify the field name with a Unicode character string in 32 characters (fixed). For the name less than 32 characters, the character string should be right-justified and filled with 0000H to become a 32-character string. ■Condition number (2) Set the number indicating an update condition. • 1: = (Equal to the determination value) • 2: != (Other than the determination value) • 3: < (Smaller than the determination value) • 4: > (Greater than the determination value) • 5: =< (Equal to or less than the determination value) • 6: >= (Equal to or greater than the determination value) • 7: is NULL (NULL (no value is set)) • 8: is not NULL (Not NULL (a value is set)) ■Logical operator setting value (3) Set the operator with an adjacent condition. • 0: No operator • 1: AND • 2: OR When one update condition is specified, set the logical operator setting value of the first update condition to 0. In this case, the second update condition is ignored if specified. Be sure to set the logical operator setting value of the second update condition to 0. ■Determination value (4) Set the value used to determine the update condition. Also when the condition number is 7 or 8, prepare a determination value area for the data size.	—	User

The following figure shows the setting format of (s5). When the data type is WORD, set as many field names, condition numbers, logical operator setting value, and determination values as there are update conditions. The data size of the determination value follows the size of the data type of each field. (☞ Page 1055 DBINSERT(P))



Ex.

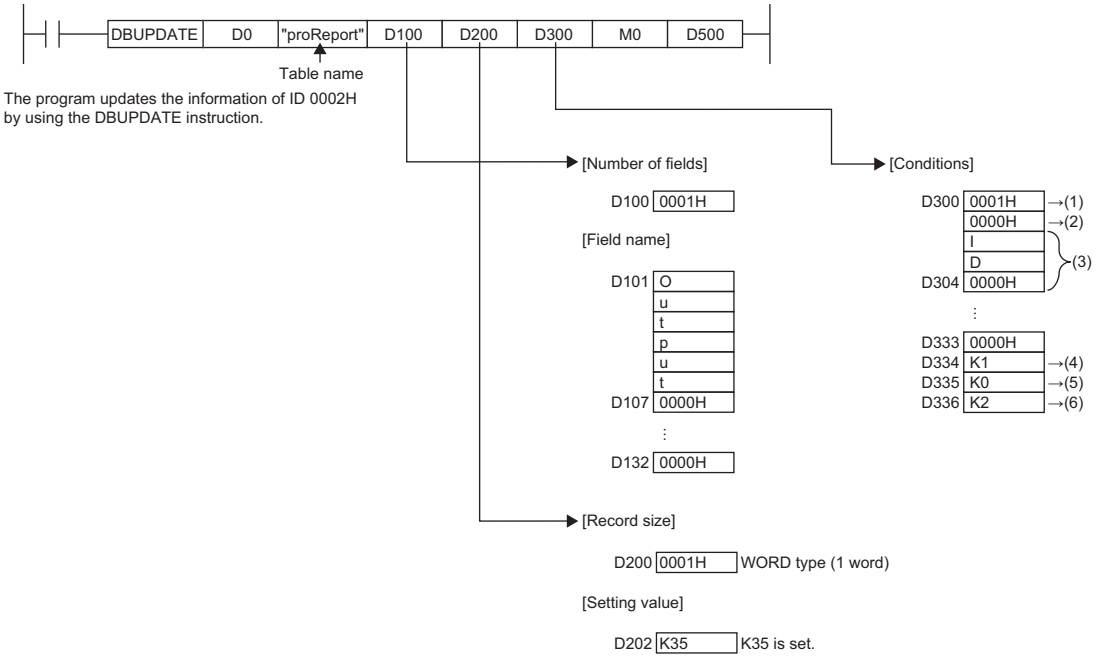
When updating Output of a record with ID = 0002H to 35 using the DBUPDATE(P) instruction

[Production result table (proReport)]

ID	Product	Output
0001H	"AAA1"	32
0002H	"BBB2"	35
0003H	"CCC3"	38
⋮		

Database

[Program]



(1) Second condition data size→WORD type (1 word)
(2) 0 for the second condition which is not used
(3) Field name of first update condition
(4) The condition number is "=". Set K1.
(5) No logical operator setting value is used. Set K0.
(6) Set K2 for the determination value.

Processing details

- Updates all record that meets the condition specified by (s5) in the table specified by (s2) in the database specified by the identification number specified by (s1).
- Specify the field name of the record to be updated in (s3). Not all fields in the table need to be specified but at least one field needs to be specified.
- Specify the value of the record to be updated in (s4). The set value in any field not specified in (s3) is not updated.
- Specify the condition to be updated in (s5). At least one condition needs to be specified and a maximum of two conditions can be specified.
- When the DBUPDATE(P) instruction updates the table recipeA record that matches "field1="2" of the update condition in which the update target field name is 2 and the updated data is New-Product1, the following occurs.

Table recipeA (before update)

Field 1 WORD	Field 2 WSTRING (16 characters maximum)	Field 3 INT
1	ProductA	100
2	ProductB	200
3	ProductC	300

Table recipeA (after update)

Field 1 WORD	Field 2 WSTRING (16 characters maximum)	Field 3 INT
1	ProductA	100
2	New-Product1	200
3	ProductC	300

- When the DBUPDATE(P) instruction updates the table recipeA record that matches "field1>="2" and field1<=3 of the update condition in which the update target field name is 2 and the updated data is New-Product1, the following occurs.

Table recipeA (before update)

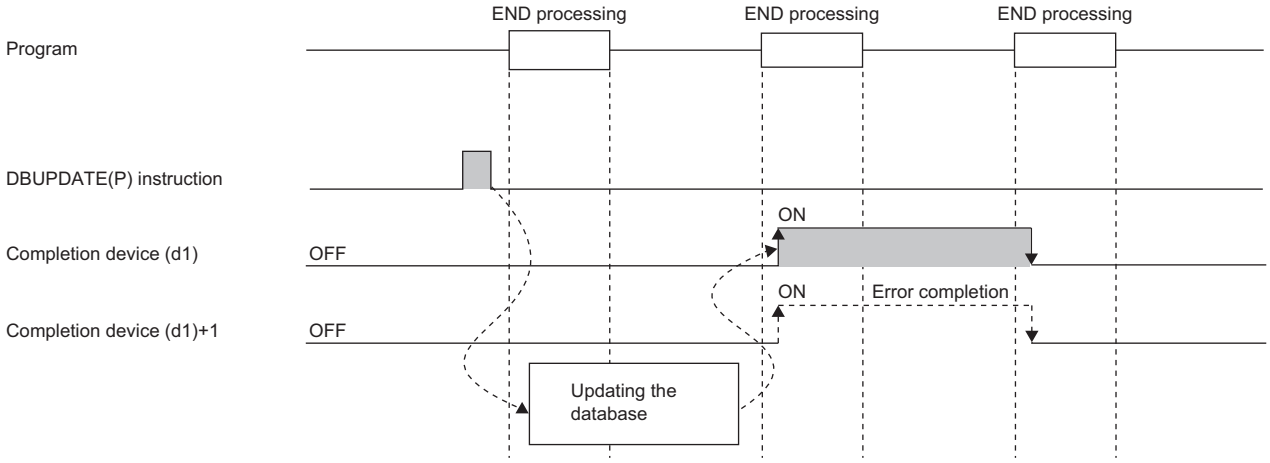
Field 1 WORD	Field 2 WSTRING (16 characters maximum)	Field 3 INT
1	ProductA	100
2	ProductB	200
3	ProductC	300
4	ProductD	200

Table recipeA (after update)

Field 1 WORD	Field 2 WSTRING (16 characters maximum)	Field 3 INT
1	ProductA	100
2	New-Product1	200
3	New-Product1	300
4	ProductD	400

- Upon successful completion, the completion signal in the completion device (d1)+0 is turned on and 0 is stored as the completion status in the device (d2).

- The following figure shows the operation of the completion device at completion of the DBUPDATE(P) instruction.



- SM753 (File being accessed) turns on while the DBUPDATE(P) instruction is executed.*1 While SM753 is on, the DBUPDATE(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBUPDATE(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.*1 If executed, no processing is performed.

*1 For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBUPDATE(P) instruction is executed during execution of the database access instruction.
- An identification number outside the range is specified by (s1).
- An identification number of a database which is not open is specified by (s1).
- A database built or operated with a programmable controller CPU with firmware version "28" or later is specified by (s1) for a programmable controller CPU with firmware version earlier than "28".
- The table name specified by (s2) does not exist.
- The number of characters of the table name specified by (s2) exceeds 32.
- An out-of-range value is specified in (s3) for the number of fields to be updated.
- An out-of-range field name is set in the field name specified by (s3)+1 to (s3)+□.
- An out-of-range value is specified in (s5) for the condition size.
- The size of the first condition in (s5) is set to 0.
- An out-of-range value is specified in (s5) for the condition symbol.
- An out-of-range value is specified in (s5) for the logical operator setting value.
- Database update processing failed.
- The logical operator setting value specified by (s5)+35 is 1 or 2, and that in (s5)+1 is set to 0.
- The range of the data for one record set in (s4)+2 does not match the size specified by (s4).
- The field name to be set in (s5)+2 is left unset.

Operation error

Error code (SD0)	Description
2820H	The area specified by (s2), (s3), (s4), (s5), or (d1) exceeds the applicable range of the device/label used.

For the error code stored in the completion status of the operand, refer to the following.

☞ Page 1086 Error codes related to database access instructions

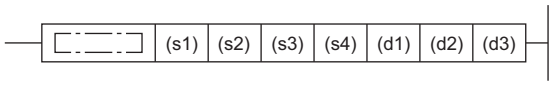
Searching the record in the data base

DBSELECT(P)

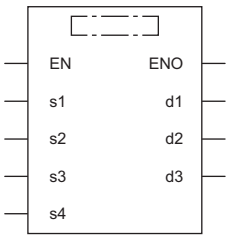
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- These instructions cannot be used for the R00CPU, R01CPU, and R02CPU.

These instructions search the records in the table in the database corresponding to the specified identification number.

Ladder	ST
	ENO:=DBSELECT(EN,s1,s2,s3,s4,d1,d2,d3); ENO:=DBSELECTP(EN,s1,s2,s3,s4,d1,d2,d3);

FBD/LD



■Execution condition

Instruction	Execution condition
DBSELECT	
DBSELECTP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Database identification number	1 to 4	16-bit signed binary	ANY16
(s2)	Start device for storing the table name of the database to be searched.*1	—	Unicode string	ANYSTRING_DOUBLE
(s3)	Start device for storing the field name of the database to be searched.*1	—	Word	ANY16*2
(s4)	Start device for storing the search conditions (a maximum of two conditions).	—	Word	ANY16*2
(d1)	Search result	—	Word	ANY16*2
(d2)	Completion device (start device that turns on for one scan upon completion of instruction) • (d2)+0: Completion signal • (d2)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d3)	Completion status • 0000: Completed successfully • Other than 0000: Completed with an error (error code)	—	Word	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 The table and field names are case-sensitive.

*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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(s4)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—
(d3)	—	—	○	—	—	—	—	○	—	—	—	—

■Search target table name

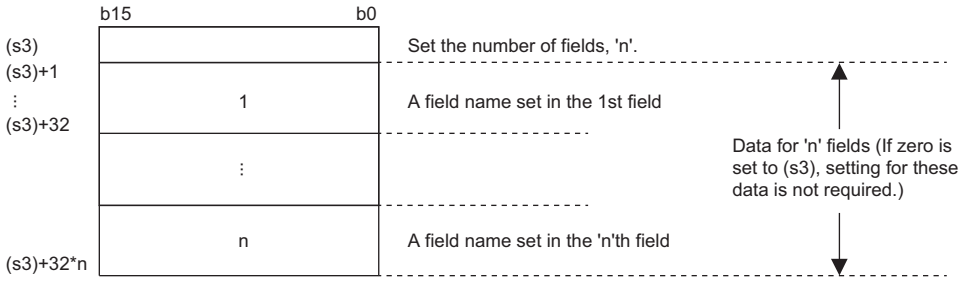
Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0 to +□	Table name	Specify the search target table name with up to 32 characters.	—	User

■Search target field name

Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Number of fields	Specify the number of fields to be searched. If 0 is specified, all fields of the table are subject to output.	0 to 128*1	User
+1 to +□	Field name	Specify each search target field name with 32 characters. The last address in (s3) varies according to the number of fields. □=33×n Specify field names, each fixed to 32 characters, by the number of fields with Unicode character strings. For the name less than 32 characters, the character string should be right-justified and filled with 0000H to become a 32-character string. This setting of this item is not necessary when "0" is specified in (s3)+0. (Ignored if specified.)	—	User

*1 "0" to "16" for a programmable controller CPU with firmware version earlier than "28"

The following figure shows the format of (s3).

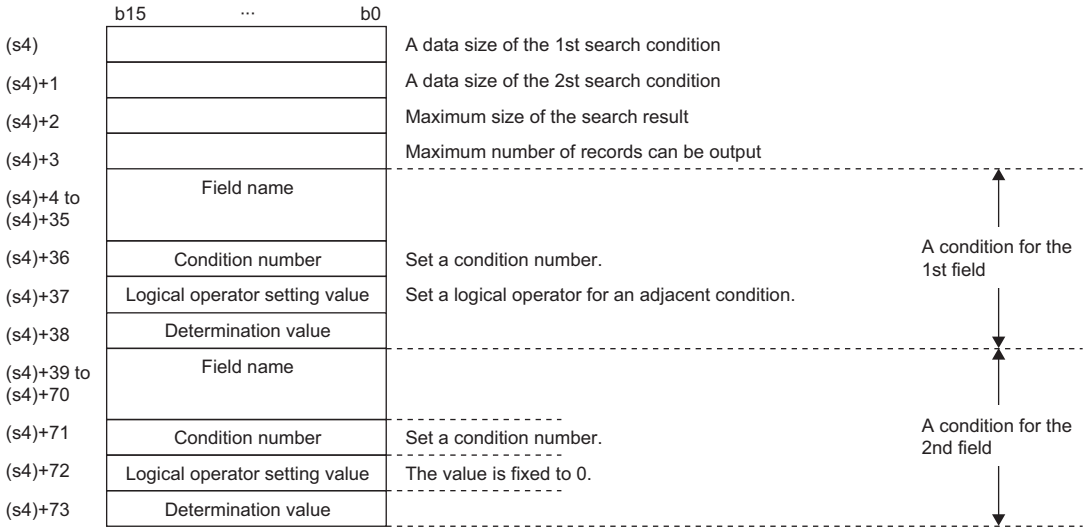


■Search condition

Operand: (s4)					
Device	**Item**	**Description**	**Setting range**	**Set by**	**Data type**
+0	Data size of the first condition	Set the data size of the first search condition in units of words in (s4)+4. If 0 is set in (s4), all records in the table are to be output. • 0: No condition • Other than 0: Data size of field (When (s4) is set to 0, set (s4)+1 also to 0.)	0 to 125	User	WORD
+1	Data size of the second condition	Set the data size of the second search condition in (s4)+4. • 0: No condition • Other than 0: Data size of field	0 to 125	User	
+2	Maximum output size	Set the maximum output size in the search result (d1). • 0: Default value (1024 words)	0 to 32768*1	User	
+3	Maximum number of outputs	Set the maximum number of outputs in the search result (d1). • 0: Default number of outputs (16 outputs)	0 to 64	User	
+4 to +□	Search condition	Set the search conditions. The last address in (s4) varies depending on the data type of the determination value. (s4)+4 ⋮ (s4)+35: (1) (s4)+36: (2) (s4)+37: (3) (s4)+38 ⋮ (s4)+□: (4) ■Field name (1) Specify field names, each fixed to 32 characters, by the number of fields with Unicode character strings. For the name less than 32 characters, the character string should be right-justified and filled with 0000H to become a 32-character string. ■Condition number (2) Set the number indicating a search condition. • 1: = (Equal to the determination value) • 2: != (Other than the determination value) • 3: < (Smaller than the determination value) • 4: > (Greater than the determination value) • 5: =< (Equal to or less than the determination value) • 6: >= (Equal to or greater than the determination value) • 7: is NULL (NULL (no value is set)) • 8: is not NULL (Not NULL (a value is set)) ■Logical operator setting value (3) Set the operator with an adjacent condition. • 0: No operator • 1: AND • 2: OR When one search condition is specified, set the logical operator setting value of the first search condition to 0. In this case, the second search condition is ignored if specified. Be sure to set the logical operator setting value of the second update condition to 0. ■Determination value (4) Set the value used to determine the update condition. Also when the condition number is 7 or 8, prepare a determination value area for the data size.	—	User	

*1 "0" to "3072" for a programmable controller CPU with firmware version earlier than "28"

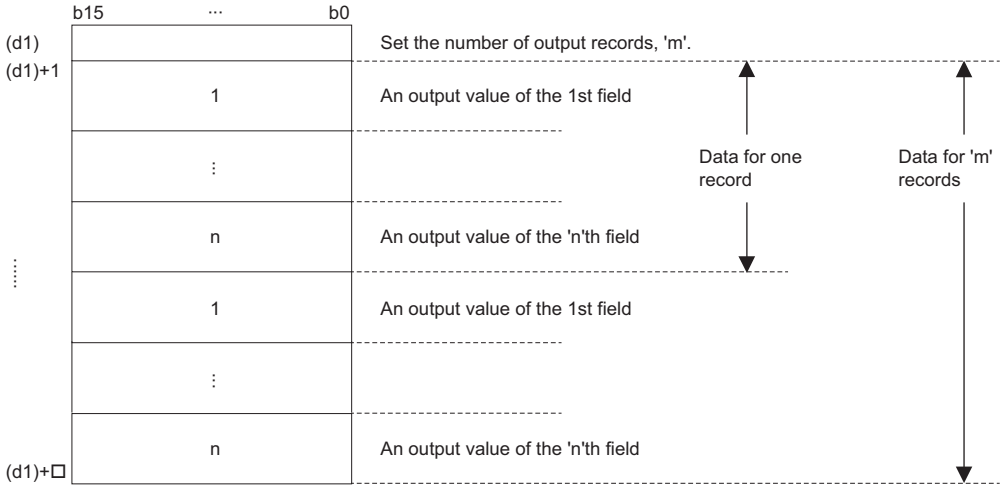
The following figure shows the setting format of (s4). (In the case of data type WORD)
Set as many field names, condition numbers, logical operator setting values, and determination values as there are update conditions. The data size of the determination value follows the size of the data type of each field. (☞ Page 1055 DBINSERT(P))



■Search result

Operand: (d1)					
Device	**Item**	**Description**	**Setting range**	**Set by**	**Data type**
+0	Number of outputs	The number of records that meet the condition set in (s4) is output. The number of outputs should be within the maximum number of outputs set in (s4)+3, and any record exceeding the maximum number of outputs is not output.	0 to 64	System	WORD
+1 to +□	Output value	The value of the records that meet the condition set in (s4) is output. Records are output right-justified in the range from (d1) to (d1)+(maximum output size), and any record that does not fit in the range is not output.	Depends on the data type	System	

The following figure shows the setting format of (d1). Data is output according to the data type corresponding to the field specified by (s3).



Ex.

When retrieving "Size(z)" with ID = 0001H using the DBSELECT(P) instruction

[Product information table (proInfo)]

ID	Product	Size(x)	Size(y)	Size(z)
0001H	"AAA1"	80	100	60
0002H	"BBB2"	40	90	40
⋮				

Database

[Program]

DBSELECT | D0 | D10 | D100 | D200 | D300 | M0 | D500

The program searches and outputs the Size(z) value of ID 0001H by using the DBSELECT instruction.

[Output result]

D300 0001H 1 record

[Searched value]

D301 K60 →(9)

[Table name]

D10 p
r
o
l
n
f
o
D17 0000H
⋮
D42 0000H

[Number of fields]

D100 0001H

[Field name]

D101 S
i
z
e
(
z
)
D108 0000H
⋮
D132 0000H

[Conditions] Set conditions to search the value of ID 0001H.

D200 0001H →(1)
D201 0000H →(2)
D202 K20 →(3)
D203 K0 →(4)
D204 I
D
D206 0000H
(5)
⋮
D235 0000H
D236 K1 →(6)
D237 K0 →(7)
D238 K1 →(8)

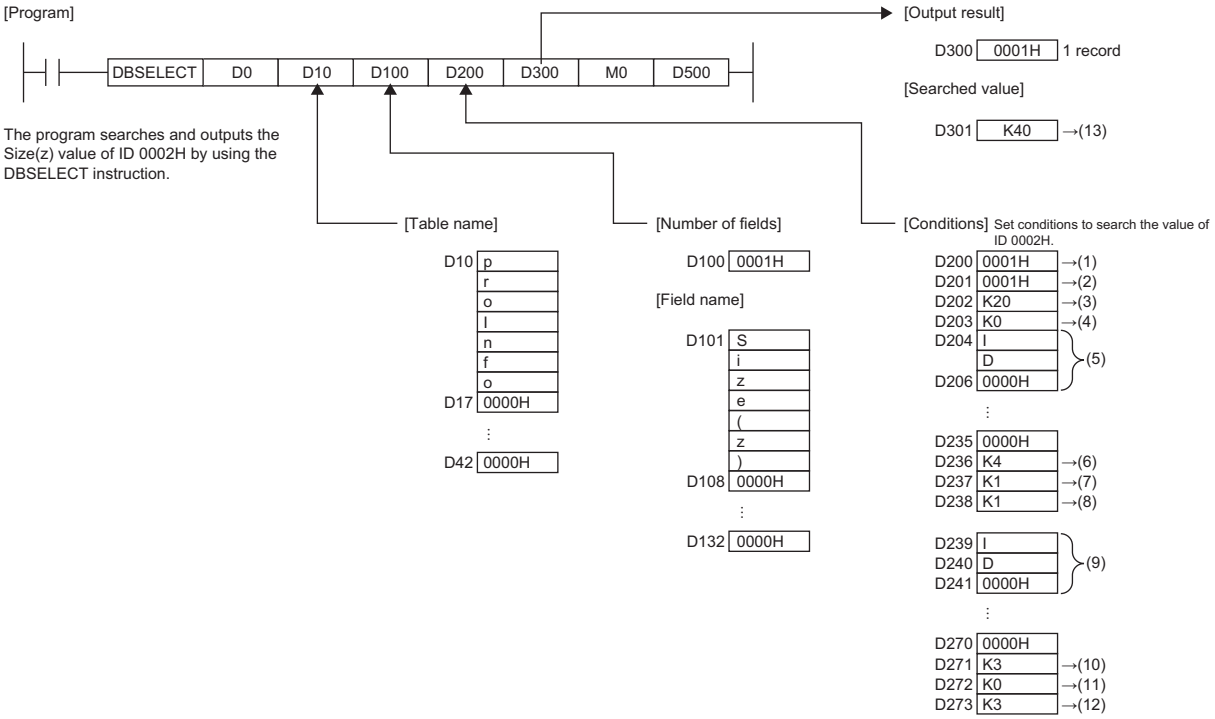
(1) Second condition data size→WORD type (1 word)
(2) 0 for the second condition which is not used
(3) Maximum output size → 20 words
(4) Maximum number of outputs → Default (0)
(5) Field name of first search condition
(6) The condition number is "=". Set K1.
(7) No logical operator setting value is used. Set K0.
(8) Set K1 for the determination value.
(9) K60 is output in WORD type (1 word).

Ex.

When retrieving the "Size(z)" value greater than ID = 0001H and less than ID = 00003H using the DBSELECT(P) instruction

[Product information table (proInfo)]

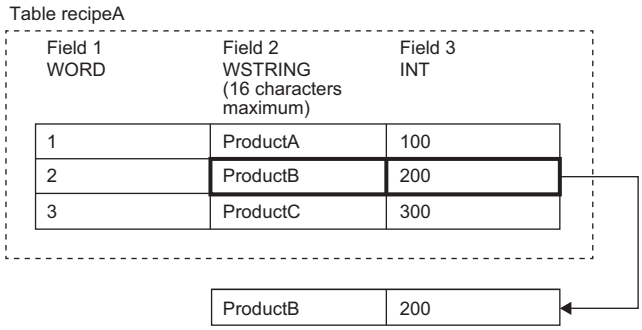
ID	Product	Size(x)	Size(y)	Size(z)
0001H	"AAA1"	80	100	60
0002H	"BBB2"	40	90	40
0003H	"CCC3"	60	110	50
⋮				



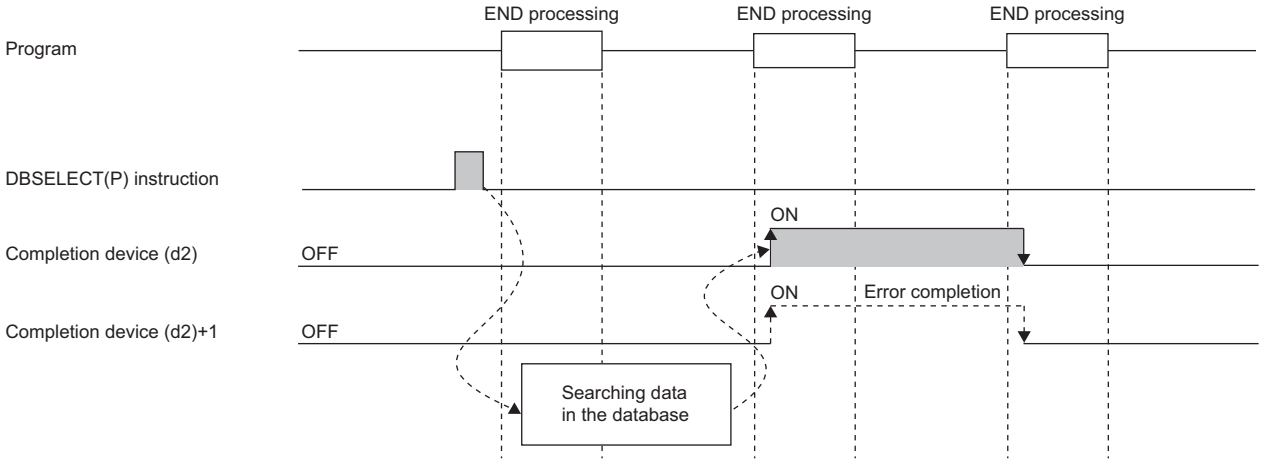
(1) Second condition data size→WORD type (1 word)
(2) Second condition data size→WORD type (2 word)
(3) Maximum output size → 20 words
(4) Maximum number of outputs → Default (0)
(5) Field name of first search condition
(6) The condition number is ">". Set K4.
(7) The logical operator setting value is "AND". Set K1.
(8) Set K1 for the determination value.
(9) Field name of second search condition
(10)The condition number is "<". Set K3.
(11)No logical operator setting value is used. Set K0.
(12)Set K3 for the determination value.
(13)K40 is output in WORD type (1 word).

Processing details

- These instructions search the records in the table specified by (s2) in the database corresponding to the identification number specified by (s1). The maximum number of outputs is 64.
- Specify the field name of the record to be searched for in (s3).
- Specify the search conditions in (s4). A maximum of two conditions can be specified.
- The search result is stored in (d1).
- Even when executed from the interrupt program, this instruction performs a record search.
- When the DBSELECT(P) instruction retrieves and outputs a table recipeA record which matches the conditions in which the output fields are 2 and 3, and the condition is field 1=2, the following occurs.



- Upon normal completion, the completion signal in the completion device (d2) is turned on and 0 is stored as the completion status in the device (d3).
- The following figure shows the operation of the completion device at completion of the DBSELECT(P) instruction.



- SM753 (File being accessed) turns on while the DBSELECT(P) instruction is executed.*1 While SM753 is on, the DBSELECT(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBSELECT(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.*1 If executed, no processing is performed.

*1 For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d2)+1 is turned on and an error code is stored as the completion status in the device (d3).

- The DBSELECT(P) instruction is executed during execution of the database access instruction.
- An invalid identification number is specified by (s1).
- An identification number of a database which is not open is specified by (s1).
- A database built or operated with a programmable controller CPU with firmware version "28" or later is specified by (s1) for a programmable controller CPU with firmware version earlier than "28".
- The table name specified by (s2) does not exist.
- The number of fields to be searched according to (s3) exceeds the maximum value.
- An out-of-range field name is set in the field name specified by (s3)+1 to (s3)+□.
- The number of records to be searched according to (s4) exceeds the maximum value.
- The size of the field to be searched according to (s4) is outside the range.
- Database selection processing failed.
- The number of records output to (d1) exceeds the number specified by (s4).
- The size of the records output to (d1) exceeds the size specified by (s4).
- The logical operator setting value specified by (s4) is out of the range.
- The logical operator setting value specified by (s4)+37 is 1 or 2, and that in (s4)+1 is set to 0.
- 0 is set in (s4) and a value other than 0 is set in (s4)+1.

Operation error

Error code (SD0)	Description
2820H	The area specified by (s2), (s3), (s4), (d1), or (d2) exceeds the applicable range of the device/label used.

For the error code stored in the completion status of the operand, refer to the following.

☞ Page 1086 Error codes related to database access instructions

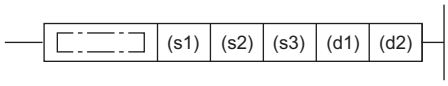
Deleting the record in the data base

DBDELETE(P)

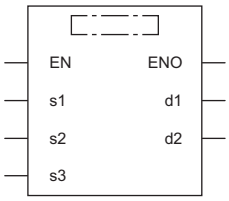
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- These instructions cannot be used for the R00CPU, R01CPU, and R02CPU.

These instructions delete the record that meets the specified condition in the specified table in the database corresponding to the specified identification number.

Ladder	ST
	ENO:=DBDELETE(EN,s1,s2,s3,d1,d2); ENO:=DBDELETEP(EN,s1,s2,s3,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBDELETE	
DBDELETEP	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(s1)	Database identification number	1 to 4	16-bit signed binary	ANY16
(s2)	Start device for storing the table name of the database to be deleted.*1	—	Unicode string	ANYSTRING_DOUBLE
(s3)	Start device for storing the deletion conditions (a maximum of two conditions).	—	Word	ANY16*2
(d1)	Completion device (start device that turns on for one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Completed successfully • Other than 0000: Completed with an error (error code)	—	Word	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

*1 The table names are case-sensitive.
*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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

■Deletion condition

Operand: (s3)					
Device	**Item**	**Description**	**Setting range**	**Set by**	**Data type**
+0	Data size of the first condition	Set the data size of the first deletion condition.	1 to 125	User	WORD
+1	Data size of the second condition	Set the data size of the second deletion condition. • 0: No condition • Other than 0: Data size	0 to 125	User	
+2 to +□	Deletion condition*1	Set the deletion conditions. Set as many deletion conditions as there are records to be deleted. The last address in (s3) varies depending on the data type of the determination value. (s3)+2 ⋮ (s3)+33: (1) (s3)+34: (2) (s3)+35: (3) (s3)+36 ⋮ (s3)+□: (4) ■Field name (1) Specify the field name with a Unicode character string in 32 characters (fixed). For the name less than 32 characters, the character string should be right-justified and filled with 0000H to become a 32-character string. ■Condition number (2) Set the number indicating a deletion condition. • 1: = (Equal to the determination value) • 2: != (Other than the determination value) • 3: < (Smaller than the determination value) • 4: > (Greater than the determination value) • 5: =< (Equal to or less than the determination value) • 6: >= (Equal to or greater than the determination value) • 7: is NULL (NULL (no value is set)) • 8: is not NULL (Not NULL (a value is set)) ■Logical operator setting value (3) Set the operator with an adjacent condition. • 0: No operator • 1: AND • 2: OR When one deletion condition is specified, set the logical operator setting value of the first deletion condition to 0. In this case, the second deletion condition is ignored if specified. Be sure to set the logical operator setting value of the second update condition to 0. ■Determination value (4) Set the value used to determine the update condition. The data size of the determination value follows the size of the data type of each field. (☞ Page 1055 DBINSERT(P)) Also when the condition number is 7 or 8, prepare a determination value area for the data size.	—	User	

*1 Set as many field names, condition numbers, logical operator setting values, and determination values as there are update conditions. (☞ Page 1062 DBUPDATE(P))

Processing details

- These instructions delete the record that meets the condition specified by (s3) in the table specified by (s2) in the database corresponding to the identification number specified by (s1).
- Specify the deletion conditions in (s3). A maximum of two deletion conditions can be specified.
- If this instruction is executed by an interrupt program during execution of another instruction, no processing is performed.
- When the DBDELETE(P) instruction deletes a table recipeA record which matches the conditions in which the condition is field 1=2, the following occurs.

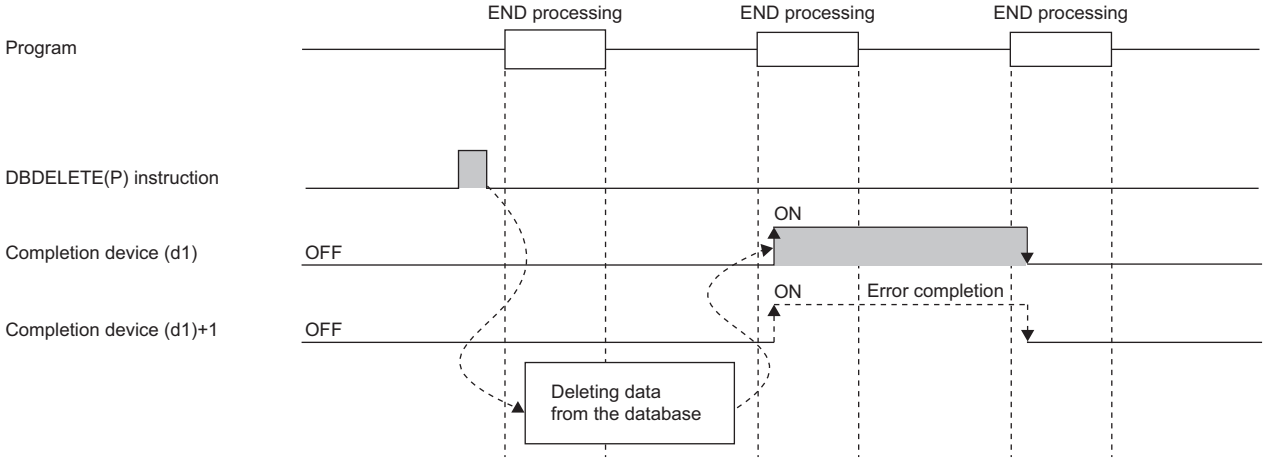
Table recipeA (before deletion)

Field 1 WORD	Field 2 WSTRING (16 characters maximum)	Field 3 INT
1	ProductA	100
2	ProductB	200
3	ProductC	300

Table recipeA (after deletion)

Field 1 WORD	Field 2 WSTRING (16 characters maximum)	Field 3 INT
1	ProductA	100
3	ProductC	300

- Upon successful completion, the completion signal in the completion device (d1)+0 is turned on and 0 is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBDELETE(P) instruction.



- SM753 (File being accessed) turns on while the DBDELETE(P) instruction is executed.*2 While SM753 is on, the DBDELETE(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBDELETE(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.*2 If executed, no processing is performed.

*2 For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- This instruction is executed during execution of the database access instruction.
- An identification number outside the range is specified by (s1).
- An identification number of a database which is not open is specified by (s1).
- A database built or operated with a programmable controller CPU with firmware version "28" or later is specified by (s1) for a programmable controller CPU with firmware version earlier than "28".
- The table name specified by (s2) does not exist.
- The number of characters of the table name specified by (s2) exceeds 32.
- An out-of-range value is specified in (s3) for the deletion condition.
- Database deletion processing failed.
- An out-of-range value is specified in (s3) for the logical operator setting value.
- The logical operator setting value specified by (s3)+35 is 1 or 2, and that in (s3)+1 is set to 0.
- The field name to be set in (s3)+2 is left unset.
- An out-of-range field name is set in the field name specified by (s3)+2 to (s3)+□.

Operation error

Error code (SD0)	Description
2820H	The area specified by (s2), (s3), or (d1) exceeds the applicable range of the device/label used.

For the error code stored in the completion status of the operand, refer to the following.

☞ Page 1086 Error codes related to database access instructions

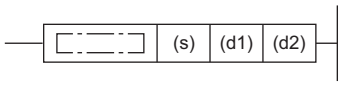
Starting a transaction

DBTRANS(P)

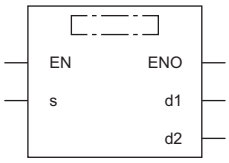
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- These instructions cannot be used for the R00CPU, R01CPU, and R02CPU.

These instructions declare the start of a transaction in relation to the database corresponding to the specified identification number.

Ladder	ST
	ENO:=DBTRANS(EN,s,d1,d2); ENO:=DBTRANSP(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBTRANS	
DBTRANSP	

Setting data

■Description, range, data type

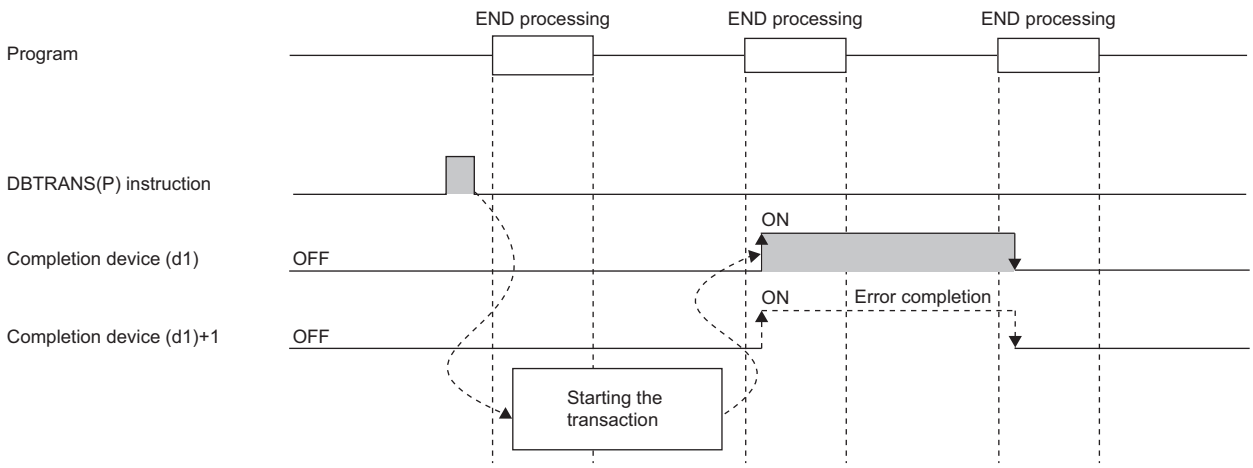
Operand	Description	Range	Data type	Data type (label)
(s)	Database identification number	1 to 4	16-bit signed binary	ANY16
(d1)	Completion device (start device that turns on for one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Normal completion • Other than 0000: Error completion (error code)	—	Word	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	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions declare the start of a transaction in relation to the database corresponding to the identification number specified by (s). However, if a data base other than the one specified by (s) is open, the DBTRANS instruction cannot start a transaction and is completed with an error.
- Upon normal completion, the completion signal in the completion device (d1) is turned on and 0 is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBTRANS(P) instruction.



- After a transaction is started by the DBTRANS(P) instruction, the DBCOMMIT(P) instruction needs to be executed to determine the transaction or the DBROLBAK(P) instruction needs to be executed to restore the state before the start of the transaction. (If the DBCLOSE(P) instruction is executed before DBCOMMIT(P) or DBROLBAK(P), the transaction is determined in the status at the execution of the DBCLOSE(P) instruction.)
- SM753 (File being accessed) turns on while the DBTRANS(P) instruction is executed.*1 While SM753 is on, the DBTRANS(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBTRANS(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.*1 If executed, no processing is performed.

*1 For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBTRANS(P) instruction is executed during execution of the database access instruction.
- An identification number outside the range is specified.
- The DBTRANS(P) instruction is executed while the transaction has already been started.
- A data base other than the one specified by (s) is open.
- The identification number of a database which is not open is specified by (s).

Operation error

Error code (SD0)	Description
2820H	The area specified by (d1) exceeds the applicable range of the device/label used.

For the error code stored in the completion status of the operand, refer to the following.

☞ Page 1086 Error codes related to database access instructions

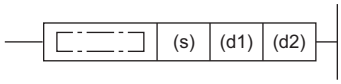
Committing a transaction

DBCOMMIT(P)

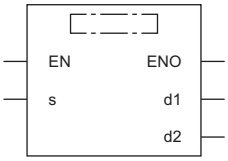
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- These instructions cannot be used for the R00CPU, R01CPU, and R02CPU.

These instructions commit the transaction in relation to the database corresponding to the specified identification number.

Ladder	ST
	ENO:=DBCOMMIT(EN,s,d1,d2); ENO:=DBCOMMITP(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBCOMMIT	
DBCOMMITP	

Setting data

■Description, range, data type

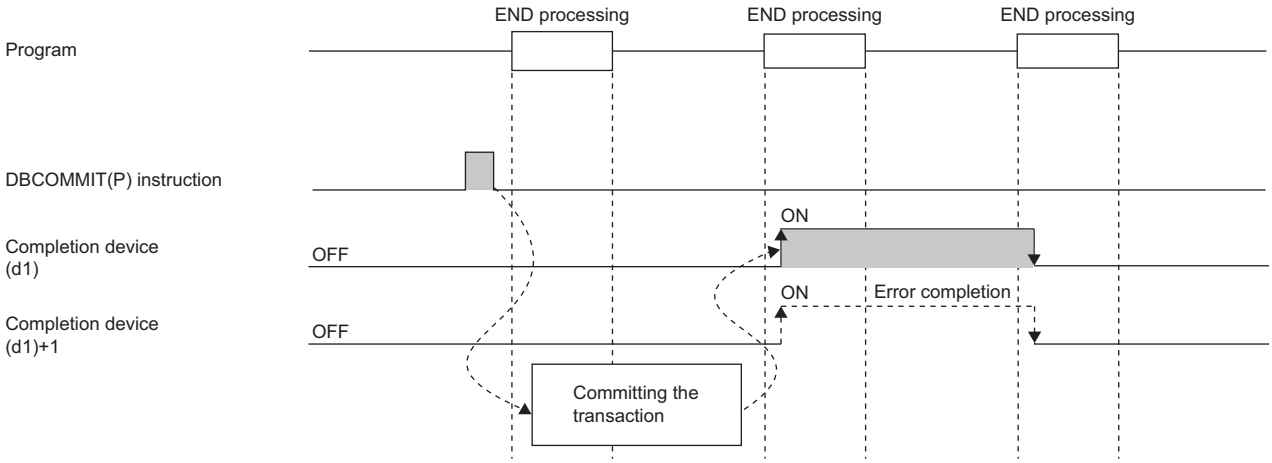
Operand	Description	Range	Data type	Data type (label)
(s)	Database identification number	1 to 4	16-bit signed binary	ANY16
(d1)	Completion device (start device that turns on for one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Normal completion • Other than 0000: Error completion (error code)	—	Word	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	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions commit the transaction in relation to the database corresponding to the identification number specified by (s).
- Upon normal completion, the completion signal in the completion device (d1) is turned on and 0 is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBCOMMIT(P) instruction.



- SM753 (File being accessed) turns on while the DBCOMMIT(P) instruction is being executed.*1 While SM753 is on, the DBCOMMIT(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBCOMMIT(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.*1 If executed, no processing is performed.

*1 For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBCOMMIT(P) instruction is executed during execution of the database access instruction.
- An identification number outside the range is specified.
- The DBCOMMIT(P) instruction is executed while no transaction is going on.
- The identification number of a database which is not open is specified by (s).

Operation error

Error code (SD0)	Description
2820H	The area specified by (d1) exceeds the applicable range of the device/label used.*2

*2 For details, refer to the following.
☞ Page 53 Checking the ranges of instruction runtime devices and labels

For the error code stored in the completion status of the operand, refer to the following.
☞ Page 1086 Error codes related to database access instructions

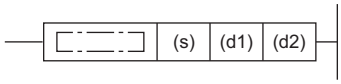
Performing a database rollback

DBROLBAK(P)

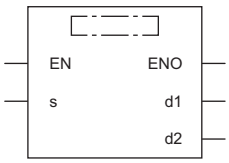
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- These instructions cannot be used for the R00CPU, R01CPU, and R02CPU.

These instructions execute the rollback of the database corresponding to the specified identification number.

Ladder	ST
	ENO:=DBROLBAK(EN,s,d1,d2); ENO:=DBROLBAKP(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DBROLBAK	
DBROLBAKP	

Setting data

■Description, range, data type

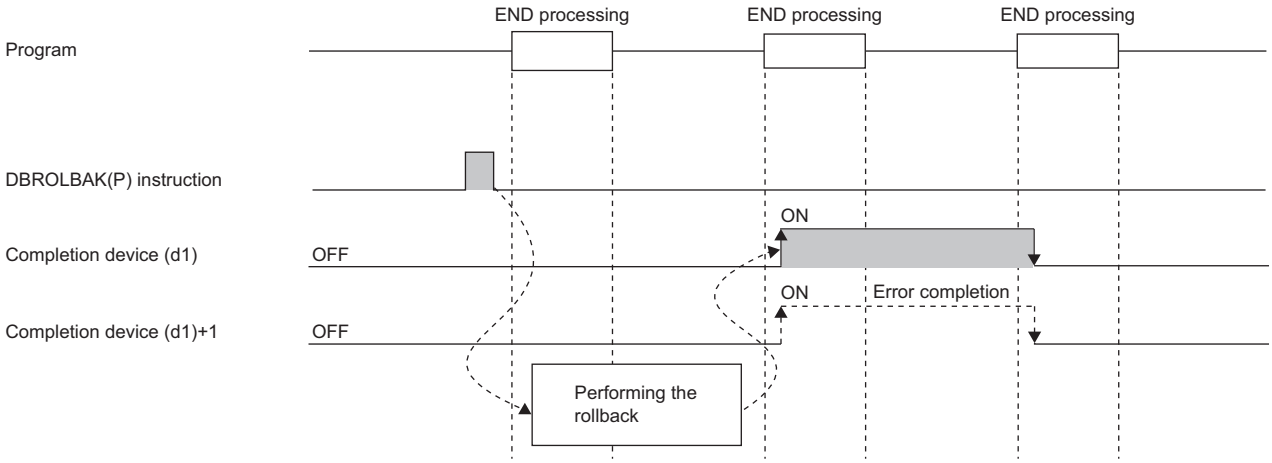
Operand	Description	Range	Data type	Data type (label)
(s)	Database identification number	1 to 4	16-bit signed binary	ANY16
(d1)	Completion device (start device that turns on for one scan upon completion of instruction) • (d1)+0: Completion signal • (d1)+1: Error completion signal	—	Bit	ANYBIT_ARRAY (Number of elements: 2)
(d2)	Completion status • 0000: Normal completion • Other than 0000: Error completion (error code)	—	Word	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	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions execute the rollback in relation to the database corresponding to the identification number specified by (s).
- Upon normal completion, the completion signal in the completion device (d1) is turned on and 0 is stored as the completion status in the device (d2).
- The following figure shows the operation of the completion device at completion of the DBROLBAK(P) instruction.



- SM753 (File being accessed) turns on while the DBROLBAK(P) instruction is executed.*1 While SM753 is on, the DBROLBAK(P) instruction cannot be executed. If executed, no processing is performed. However, the instruction has been completed with an error while the database access instruction is being executed.
- When "To Use or Not to Use the Built-in Database Access" is set to "Use" in the module parameter of the CPU module, the DBROLBAK(P) instruction cannot be executed while SM1498 (CPU module database start-up flag) is off.*1 If executed, no processing is performed.

*1 For the firmware versions supporting SM753 and SM1498, refer to the list of special relay areas. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

Precautions

In the following cases, the error completion signal in (d1)+1 is turned on and an error code is stored as the completion status in the device (d2).

- The DBROLBAK(P) instruction is executed during execution of the database access instruction.
- An identification number outside the specified range is specified.
- The DBROLBAK(P) instruction is executed while no transaction is going on.
- The identification number of a database which is not open is specified by (s).

Operation error

Error code (SD0)	Description
2820H	The area specified by (d1) exceeds the applicable range of the device/label used.

For the error code stored in the completion status of the operand, refer to the following.

☞ Page 1086 Error codes related to database access instructions

Error codes related to database access instructions

The following table lists the error codes that could be stored in the completion status of database access instructions.

Error code	Error content	Action
0103H	CPU internal error	Turn power off and turn it on again or reset the system and check whether the instruction can be executed.
0104H	Another database access instruction is being executed.	Check that another database access instruction is not executed before one database access instruction is completed.
0105H	• The specified path is not a database folder. • The database is broken.	Check whether the folder specified by the path contains a database file. If not, correct the path properly.
0106H	• A file in the database cannot be opened. • A database built with a different firmware version is specified.	Create a new database using the DBIMPORT(P) instruction.
0107H	An invalid drive number is specified by the database access instruction.	Check whether the specified drive number is 2. If not, specify drive number 2 correctly.
0108H	Deletion of the database failed.	Check that another function is not accessing the database and create a new database using the DBIMPORT(P) instruction.
07D1H	The specified table is not found in the database.	Check that the database contains the specified table.
0940H	An invalid character is included in the table name.	Begin the table name with a non-numeric character and do not include an invalid character.
0941H	The table name is duplicated.	Database table names are not case-sensitive. Check whether same characters are used to set Unicode text file table names by differentiating them with uppercase and lowercase characters.
0942H	The specified table is not found in the database.	Check that the database contains the specified table.
0945H	The specified database is in use by another function.	After the database operation by another function has been completed, execute the instruction again.
0946H	The specified table is in use by another function.	After the table operation by another function has been completed, execute the instruction again.
094AH	The database name is not specified.	Specify a correct database name.
094BH	No database is stored in the specified folder.	Check that the folder name is correct and that the folder contains a database-related file.
094CH	The field name contains an invalid character.	Do not include any invalid character in the field name.
094DH	The specified field name is duplicated.	Check for duplicated field names in Unicode text files or instruction arguments. If any, correct it to prevent duplication.
094EH	A non-existent field is specified.	Check that the specified field exists in the table.
0950H	An invalid character is included in the index.	Do not include any invalid character in the index.
0951H	An attempt was made to set empty data (NULL value) in the main key or in the field that has a NOT NULL constraint.	Confirm the table structure. Check whether the field permits NULL. If not, set significant data.
0952H	An attempt was made to add the same value to the field that has a primary key constraint.	Check that the setting value of the record attempted to add to the field that has a primary key constraint is not the same as the record already registered.
0953H	If overlapped data are set to the field where an index is set repeatedly, the data for the index become large and the memory of the CPU module cannot secure enough free space.	Change the table definition where the error is detected and re-build the database.
0954H	The setting value cannot be automatically converted to the format appropriate for the data type.	Check that the format of the setting value meets the specifications.
0959H	The data type that is checked during importing differs from the data type defined in the table.	Check the Unicode text to verify that the setting value to be imported is appropriate for the data type of the field.
095CH	The data types of compared values are incorrect such as comparing the sizes of character strings.	Correct the values to be compared.
095DH	Converting the data type of the setting value specified in an argument of the database access instruction failed and therefore the setting value cannot be stored in the field.	Check that the setting value meets the data type specifications.
095EH	An attempt was made to assign more than one index to one field.	Check that an attempt has not been made to add an index to the field where the primary key or external key is set. Check also that an attempt has not been made to assign more than one index to one field.
095FH	The index name is duplicated.	Check that each index name is not duplicated.
096CH	The setting value to be added to the field where the external key is specified as a key constraint is not included in the setting values of the reference.	Check the setting values of the reference.
096EH	The record cannot be deleted because it is referenced by another table.	Check whether the setting values of the record to be deleted include the value of a table referenced by an external key.

Error code	Error content	Action
0970H	An attempt was made to add or delete a record but failed because no table was available for referencing a setting value included in the record.	Check that a referenced table is available.
0971H	An attempt was made to add a setting value not registered in the referenced table.	Check the setting values in the referenced table.
0972H	The data type in the field referenced by an external key is inconsistent.	Check that the data types in both fields match.
0974H	There is no table to be referenced by an external key.	Specify a correct table name.
0976H	There is no field of the table to be referenced by an external key.	Specify a correct field name.
0984H	A transaction is already running.	Check that multiple transactions are not executed concurrently.
09C2H	An operation is attempted for a database not supported by the CPU module used.	■For a programmable controller CPU with firmware version earlier than "28" After executing the DBEXPORT(P) instruction with a programmable controller CPU with firmware version "28" or later, use the acquired Unicode text file to execute the DBIMPORT(P) instruction with a programmable controller CPU with firmware version earlier than "28". ■For a programmable controller CPU with firmware version "28" or later After executing the DBEXPORT(P) instruction with a programmable controller CPU with firmware version earlier than "28", use the acquired Unicode text file to execute the DBIMPORT(P) instruction with a programmable controller CPU with firmware version "28" or later.
09C3H	A database created by executing the DBIMPORT(P) instruction on a programmable controller CPU with firmware version earlier than "28" was processed with a programmable controller CPU with firmware version "28" or later, and then processed on a programmable controller CPU with firmware version earlier than "28".	• Process the database where an error occurs with a programmable controller CPU with firmware version "28" or later. • To process the database on a programmable controller CPU with firmware version earlier than "28", execute the DBEXPORT(P) instruction on a programmable controller CPU with firmware version "28" or later, and then execute the DBIMPORT(P) instruction on a programmable controller CPU with firmware version earlier than "28" using the output Unicode text file.
09D7H	BOOL data is not specified.	Specify BOOL data.
09F9H	A non-existent field is specified to add a record.	Check the table structure.
0A0AH	A prohibited field name is specified.	Specify another field name.
0A16H	An attempt was made to add a record with a null value set in the field where the primary key was set.	Check the Unicode text to verify that a null value is not set in the field where the primary key is set.
0A17H	An attempt was made to add a record with the same value included in the field where the primary key was set.	Check the Unicode text to verify that the same value is not set in the field where the primary key is set.
0A2BH	An attempt was made to set multiple primary keys.	Check the key setting line in the Unicode text to verify that multiple primary keys are not set.
0A2FH	The primary key does not exist in the referenced table.	Check the Unicode text to verify that the setting value to be imported is appropriate for the data type of the field.
0A30H	The setting values in the field where an external key is set do not include a setting value in the field where the primary key is set.	Check that the setting value in the field where an external key is to be added is included in the setting values in the field where the primary key is set.
0A31H	An attempt was made to change or delete a setting value in the field referenced by an external key.	Check the reference relationships among the tables.
0A50H	An attempt was made to assign more than one index name to the same table.	Check the content of the Unicode text to avoid duplication of index names.
0A5DH	The data types of the fields to be connected do not match.	Check that the data types of the fields to be connected match.
0A5FH	A field name is duplicated in the same table.	Check whether a field name is specified more than once in the same table. If so, correct either field name.
0AAAH	A negative value is specified in the unsigned integral data type field.	Check that the field data type accepts negative values.
0AB1H	An attempt was made to specify a character string for the integral data type.	Check that no character string is specified for an integral data type field.
0AB9H	An attempt was made to assign an index to a field where no value has been set.	Check that values have been set in all field records where an index is to be specified.
0AECH	Table opening failed.	Turn power off and turn it on again or reset the system and check whether the table can be opened.
1000H	The Unicode text file specified for importing does not exist.	Check whether the file indicated by the folder path is a Unicode text file. If not, specify a correct folder path.
1002H	There is no identification number that can be allocated.	A maximum of four databases can be opened. Adjust the number of databases that will be opened to 4.
1003H	The identification number of a database which is not open is specified.	Use the DBOPEN(P) instruction to open the database in advance and obtain an identification number.

Error code	Error content	Action
1004H	The identification number specified by the instruction is out of range.	Specify the identification number obtained by the DBOPEN(P) instruction.
1007H	The number of fields to be specified by the instruction is not specified.	Set 1 or greater for the number of fields in the argument.
1008H	• The number of fields specified by the instruction is out of range. • The field name specified by the instruction is out of range.	• Set 128 or less for the number of fields in the argument.[*1] • Specify a field name within the available range (up to 128th field) in the argument.
1009H	The number of records to be specified by the instruction is not specified.	Set 1 or greater for the number of records in the argument.
100AH	The number of records specified by the instruction is out of range.	Set the number of records within the range in the argument.
100CH	The Unicode text table or set-value delimitation does not follow the format.	Check that the delimiters in the table or set-value delimitations follow the format.
100DH	An error occurred in the database.	Turn power off and turn it on again or reset the system and check whether the same symptom recurs.
100EH	The condition number specified by the instruction is out of range.	Check that the condition number range is correct.
100FH	When two conditions are specified by the instruction, the logical operator setting value is 0.	Check that the logical operator setting value in the argument is 1 or 2.
1010H	The maximum number of databases that can be transacted is exceeded.	Check that a transaction is not executed for two or more databases or that another transaction is not executed while one transaction is already running.
1011H	The commit or rollback instruction is executed without starting a transaction.	Check that the commit or rollback instruction is not executed while no transaction is running.
1012H	The value of the size per record used for adding or update is not appropriate.	Check that the data size of each field matches the size of the data to be added or updated.
1013H	A data type which is not supported is specified.	Check the data type of the Unicode text.
1014H	• Another database is opened during execution of the transaction. • The DBEXPORT(P) instruction is executed during transaction.	• Do not open a database newly during execution of the transaction. • Execute the DBEXPORT(P) instruction after the transaction is completed.
1015H	A transaction is executed for the multiple databases that were already open.	Close the databases that are not targeted for a transaction.
101BH	The first condition is not set in the relevant argument of each instruction.	Check that the field size set as the first condition in the argument is other than 0 or whether the field name is null.
101CH	The data size of a field is not included in multiple condition settings.	Check that the data size of the field specified by the condition is 0.
101DH	A field name is not included in multiple condition settings.	Check that the field name specified by the condition is not null.
101EH	The value of the inter-condition operator is out of range.	Check that the value of the inter-condition operator is correct.
101FH	The value of the inter-condition operator in the second condition setting is out of range.	Check that a value other than 0 is not set in the logical operator setting value of the second condition.
1020H	The setting value is not within the range specified for the data type.	Check that the setting value is within the range.
1021H	An attempt was made to open a database which was already open.	Check that the path corresponding to (s) of the DBOPEN(P) instruction is not specified more than once.
1022H	The DBIMPORT(P) instruction was executed while the database was already open.	Close the database that is already open, and execute the DBIMPORT(P) instruction.
1023H	A table number outside the range is specified.	Check that the table number is within the range.
102EH	The field size specified in the conditions is out of range.	Check that the field size in the conditions is within the range.
1030H	The output size exceeds the setting value.	Adjust the output size.
1031H	The number of outputs exceeds the setting value.	Adjust the number of outputs.
1032H	Since the SD memory card was removed while opening the database with the DBOPEN(P) instruction or executing a database access instruction, the database access instruction cannot be executed or the CPU module database access (from external device) function cannot be executed from an external device.	Power off and on the CPU module or reset it.
1040H	The total size of records to be added exceeds the limit.	Adjust the total size of records to be added.
2000H	The format of the Unicode text file is incorrect.	• Check whether the format of the Unicode text file is correct. If not, correct it. • Check whether the number of characters in the comment line is within the range.
2001H	The key setting of the Unicode text file is incorrect.	Check that an out-of-range value is not set with regard to the key constraint of the Unicode text file.

Error code	Error content	Action
2002H	The number of characters making up a database name, table name, or field name exceeds the limit.	• Check that the database name, table name, or field name in the Unicode text file does not exceed 32 characters. • Check whether 128 or more tabs are used in a single line.
2003H	The number of tables exceeds the limit.	Check that the number of tables in the Unicode text file does not exceed 32.*1
2004H	The format of the setting value of the record in the Unicode text file is incorrect.	Check that the format of the setting value in the Unicode text file is correct.
2005H	The maximum number of records that can be imported is exceeded.	Check that the number of records in the text file does not exceed the maximum number (100,000).
2006H	An invalid character is included in the database name.	Check that an invalid character is not included in the database name in the text file.
2007H	The format of the row index is incorrect.	Check that the format of the row index is correct.
2009H	Failed to read data from the Unicode text file.	• Check the status of the SD memory card. • Check the contents of the Unicode text file.
200AH	Failed to write data to the Unicode text file.	Check the status of the SD memory card.
200BH	The number of fields in the Unicode text file exceeds the maximum number.	Check that the number of fields in the Unicode text file does not exceed 128.*1
200CH	In the Unicode text file, the number of columns set in the field row does not match that set in another row.	Check the Unicode text file to verify that the number of columns set in the field row (number of fields) matches that set in another row.
200DH	The number of indexes in the Unicode text file exceeds the maximum number.	Check that the number of indexes in the Unicode text file does not exceed the number of table fields.
200EH	The number of characters of the main key name in the Unicode text file, external key name, or index name exceeds the maximum number.	Check whether the numbers of key name and index name characters are each 16 or less.
200FH	The database access instruction cannot be executed because the SD memory card is write-protected. Or, startup of the CPU module database access (from external device) function from an external device failed.	Clear the write protection of the SD memory card.
2010H	The database access instruction cannot be executed or the CPU module database access function cannot be executed from an external device because the SD memory card does not have enough free space (10M bytes or more).	Secure enough free space of the SD memory card.
2011H	The number of characters making up the path of the database exceeds 128.	Reduce the number of characters making up the path to 128 or less.
2020H	The CPUDB folder configuration is incorrect.	• Decrease the number of databases in the CPUDB folder to 32 or less. • Delete folders which are not databases in the CPUDB folder. • Refer to the error database check file (2:\CPUDB\ErrorDB.txt) and delete a folder which caused the error.
2021H	The database is broken.	• Create a new database. • Refer to the error database check file (2:\CPUDB\ErrorDB.txt) and delete a database which caused the error.
2023H	Startup of the database failed.	• Power off and on the CPU module or reset it. • Check the data memory capacity and secure a free space of 1K byte or more.
2024H	Suspension of the database failed. Or, the database was suspended because the SD memory card was removed.	• Power off and on the CPU module or reset it. • Insert the SD memory card again, and power off and on the CPU module or reset it.
2025H	The database cannot be recognized. Or recovery of the database failed.	• Check the status of the SD memory card. • If the error database check file (2:\CPUDB\ErrorDB.txt) exists and it lists a database which resides under the CPUDB folder or its subfolder, delete the database with its folder and then create a new database. • If the problem persists after powering off and on the CPU module or reset it, or the error database check file (2:\CPUDB\ErrorDB.txt) exists and it lists a database which resides under other than the CPUDB folder or its subfolder, delete the dbmaintainpath.txt file on the $MELPRJ$\$DBASYS$ folder in the SD memory card, delete all database folders (excluding the CPUDB folder), and then create a new database using the DBIMPORT(P) instruction.
Others	• The character code of the Unicode text file is incorrect. • The database to be accessed is in an invalid status.	• Check the character code of the Unicode text file. • If the same error occurs even after powering off and on or reset the CPU module, delete the access-target database folder, and create a new database by using the DBIMPORT(P) instruction.

*1 Use 16 or less fields/tables for a programmable controller CPU with firmware version earlier than "28".

19 CLOCK

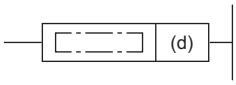
19.1 Clock Instructions

Reading clock data

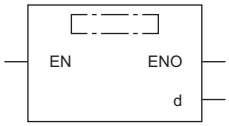
DATERD(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions read "year, month, day, hour, minute, second, and day of week" from the clock element of the CPU module.

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□\□, U3E□\(H)G□	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 CPU module, 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

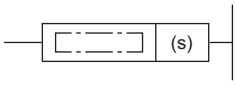
There is no operation error.

Writing clock data

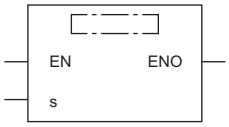
DATEWR(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions write the clock data stored in the specified device and later to the clock element of the CPU module.

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions write the clock data stored in the device specified by (s) and later to the clock element of the CPU module.

(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
3405H	The data in the device specified by (s) to (s)+6 are out of range.
3425H	A time value less than an hour from the start time of the daylight-saving time is set in (s) to (s)+6.

Point

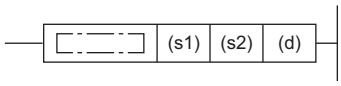
When clock data is changed, "clock setting" (event code: 24000) is saved to the event history. That is, "clock setting" is saved to the event history when this instruction is executed.

Adding clock data

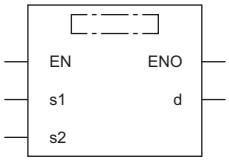
DATE+(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions add time data.

Ladder	ST*1
	ENO:=DATEPLUS(EN,s1,s2,d); ENO:=DATEPLUSP(EN,s1,s2,d);

FBD/LD



(☐ is replaced by either of the following: DATEPLUS, DATEPLUSP.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■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☐\☐, U3E☐\(H)G☐	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

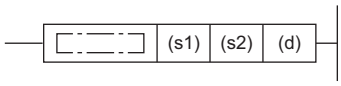
Error code (SD0)	Description
3405H	The data in the device specified by (s1) or (s2) is out of range.

Subtracting clock data

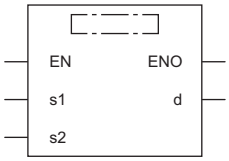
DATE-(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions subtract time data.

Ladder	ST*1
	ENO:=DATEMINUS(EN,s1,s2,d); ENO:=DATEMINUSP(EN,s1,s2,d);

FBD/LD



(☐ is replaced by either of the following: DATEMINUS, DATEMINUSP.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■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☐\☐, U3E☐\(H)G☐	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

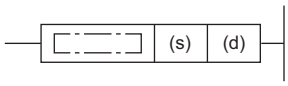
Error code (SD0)	Description
3405H	The data in the device specified by (s1) or (s2) is out of range.

Converting time data from hour/minute/second to second

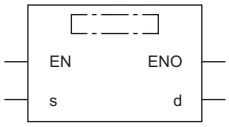
TIME2SEC(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

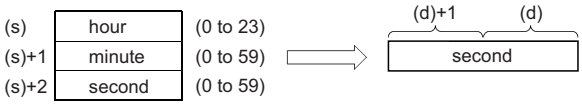
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○*1	○	○	○	○	○	○	○	—	—	—	—

*1 FX and FY cannot be used.

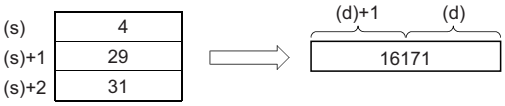
Processing details

- 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

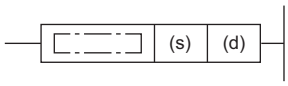
Error code (SD0)	Description
3405H	The data in the device specified (s) is out of range.

Converting time data from second to hour/minute/second

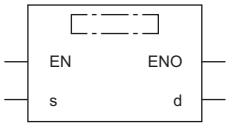
SEC2TIME(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

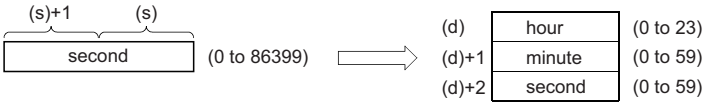
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

*1 FX and FY cannot be used.

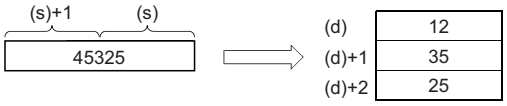
Processing details

- 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
3405H	The data in the device specified (s) is out of range.

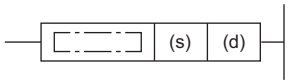
Converting date and time data from date and time to second

DATE2SEC(P)(_U)

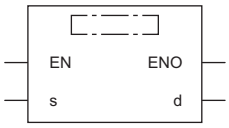
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.
- The RnPCPU with firmware version "21" or later supports these instructions. Use an engineering tool with version "1.040S" or later.

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

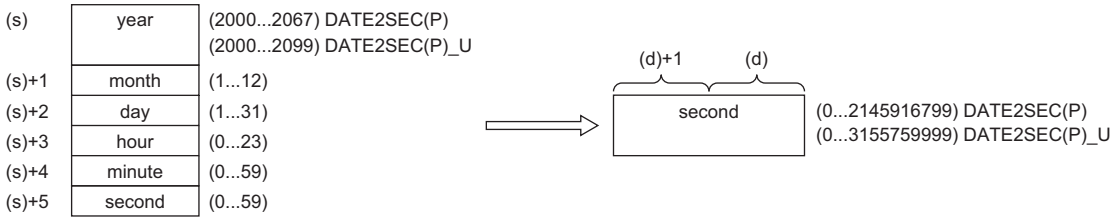
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○*1	○	○	○	○	○	○	○	—	—	—	—

*1 FX and FY cannot be used.

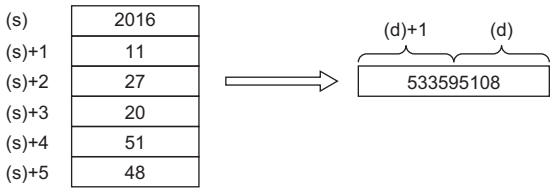
Processing details

- 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.



- 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.



- Changing clock data (including time adjustment by the time zone and 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
3405H	The data in the device specified (s) is out of range.
	A non-existent date and time is specified. (Example: February 30, 2016)

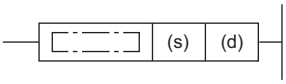
Converting date and time data from second to date and time

SEC2DATE(P)(_U)

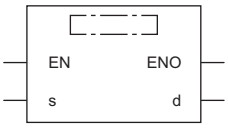
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU with firmware version "06" or later support these instructions. Use an engineering tool with version "1.047Z" or later.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "31" or later support these instructions. Use an engineering tool with version "1.040S" or later.
- The RnPCPU with firmware version "21" or later supports these instructions. Use an engineering tool with version "1.040S" or later.

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

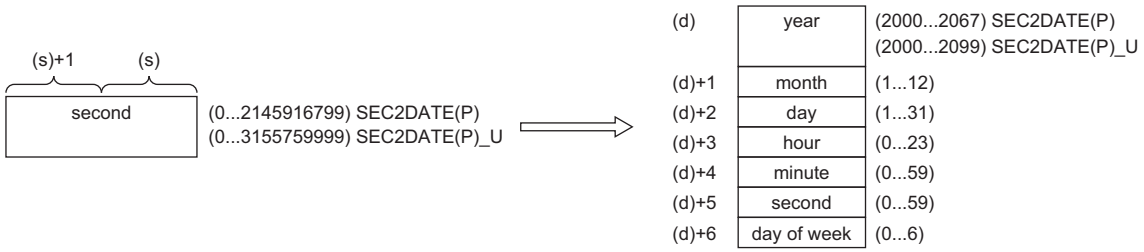
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	○	○	○	○	○	○	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

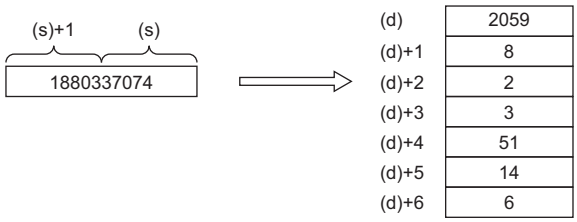
*1 FX and FY cannot be used.

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

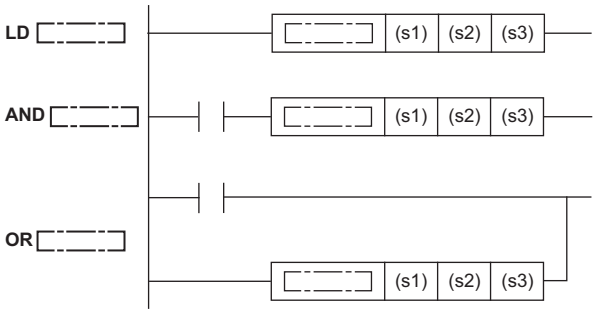
Error code (SD0)	Description
3405H	The data in the device specified (s) is out of range.

Comparing date data

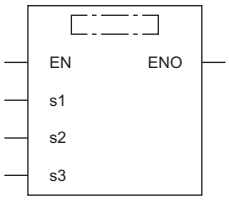
LDDT□, ANDDT□, ORDT□

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions compare the specified date data, or compare the date data with the current date.

Ladder	ST*1
 (□ 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.)*2

FBD/LD



(□ is replaced by a combination of LDDT_, ANDDT_, or ORDT_ and EQ, NE, GT, LE, LT, or GE.)*2

*1 The engineering tool with version "1.035M" or later supports the ST.
*2 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

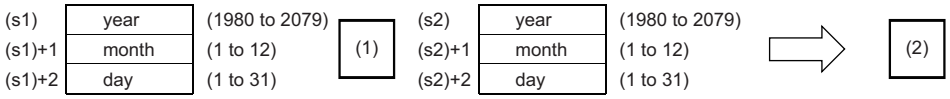
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	○	○	○	○	○	—	—	○	○	—	—	—

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

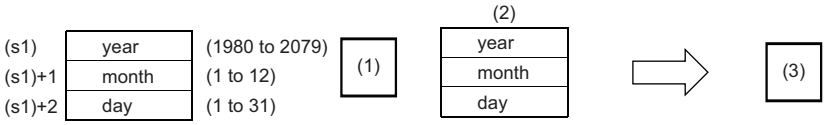
These instructions compare the date data in the device specified by (s1) with the date data in the device specified by (s2) in accordance with the conditions set by (s3). (Devices are used as a normally open contact.)



(1)Relational operator
(2)Comparison result

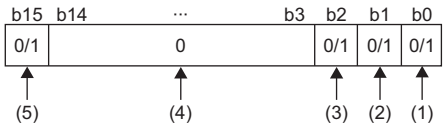
• Comparing the specified date data with the current date

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 1 is set to the 15 bit, the data in the device specified by (s1) is compared with the current date in accordance with the conditions set in the 0 to 2 bits.

- When 0 is set to the 0 to 2 bits, 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 the 15 bit, 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.

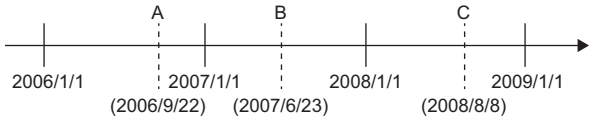
(s3) value when comparing two specified date data	(s3) value when comparing the specified date data with the current date	Comparison target	Description
0001H	8001H	Day	Only data in the device specified by (s1)+2 is compared.
0002H	8002H	Month	Only data in the device specified by (s1)+1 is compared.
0003H	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 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 setting area in the device/label memory, 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 CPU module 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	Non-continuity	Continuity
Month	Non-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.

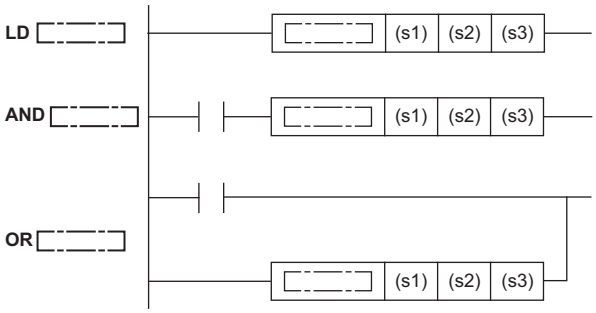
Comparing time data

LDTM□, ANDTM□, ORTM□

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions compare the specified time data, or compare the specified time data with the current time.

Ladder



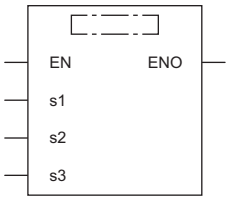
(□ is replaced by TM=, TM<>, TM>, TM<=, TM<, or TM>=.)

ST*1

```
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.)*2

FBD/LD



(□ is replaced by a combination of LDTM_, ANDTM_, or ORTM_ and EQ, NE, GT, LE, LT, or GE.)*2

*1 The engineering tool with version "1.035M" or later supports the ST.
*2 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

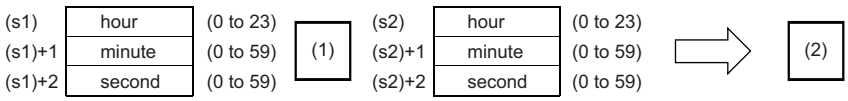
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	○	○	○	○	○	—	—	○	○	—	—	—

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

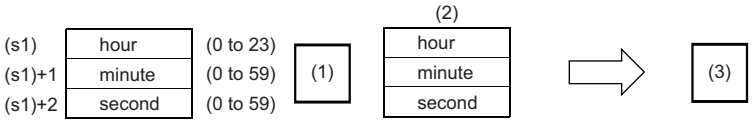
These instructions compare the time data in the device specified by (s1) with the time data (hour, minute, second) in the device specified by (s2) in accordance with the conditions set by (s3). (Devices are used as a normally open contact.)



(1)Relational operator
(2)Comparison result

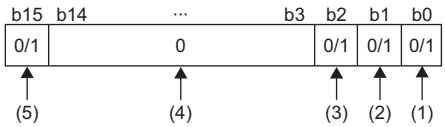
• Comparing the specified time data with the current time data

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 1 is set to the 15 bit, the data in the device specified by (s1) is compared with the current time in accordance with the conditions set in the 0 to 2 bits.

- 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.

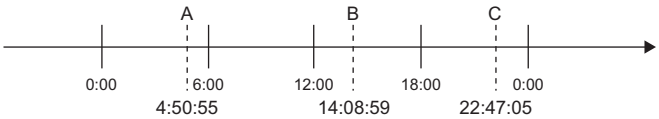
(s3) value when comparing two specified time data	(s3) value when comparing with current time data	Comparison target time	Description
0001H	8001H	Second	Only data in the device specified by (s1)+2 is compared.
0002H	8002H	Minute	Only data in the device specified by (s1)+1 is compared.
0003H	8003H	Minute, 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 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 setting area in the device/label memory, 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 CPU module 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.

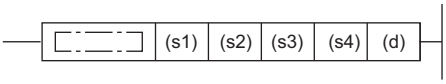
Outputting a comparison result of time data

TCMP(P)

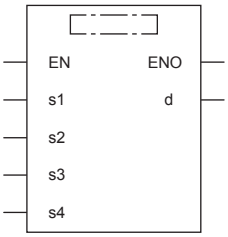
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

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

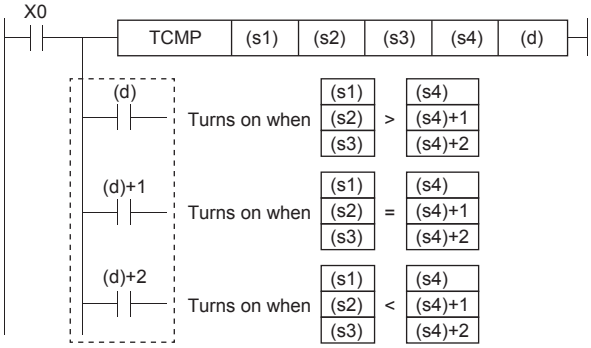
■Applicable devices

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□\□, U3E□\(H)G□	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
3405H	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

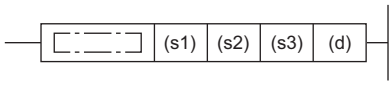
Outputting a band comparison result of time data

TZCP(P)

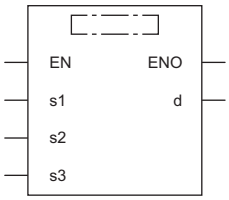
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support these instructions. Use an engineering tool with version "1.020W" or later.

These instructions compare the band between the time data of lower limit value (s1) and the time data of upper limit value (s2) with the time data (s3) to be compared, and according to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.

Ladder	ST
	ENO:=TZCP(EN,s1,s2,s3,d); ENO:=TZCPP(EN,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
TZCP	
TZCPP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device for storing the lower limit value of the time data (hour, minute, second) to be compared	Refer to the function details.	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(s2)	Start device for storing the upper limit value of the time data (hour, minute, second) to be compared	Refer to the function details.	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(s3)	Start device for storing the time data (hour, minute, second) to be compared	Refer to the function details.	16-bit signed binary	ANY16_ARRAY (Number of elements: 3)
(d)	Start device for storing the comparison result	—	Bit	ANYBIT_ARRAY (Number of elements: 3)
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

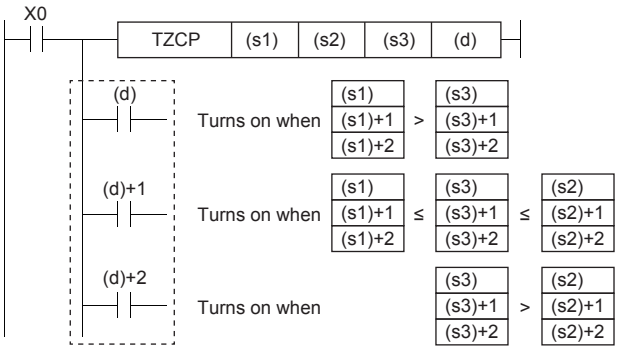
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(s3)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○*1	—	—	—	—	○	—	—	—	—

*1 T, ST, and C cannot be used.

Processing details

- These instructions compare the band between the time data of lower limit value (s1) and the time data of upper limit value (s2) with the time data (s3) to be compared, and according to the comparison result (below, within zone, or above), (d), (d)+1, or (d)+2 is turned on.



Device	Clock data	Data range
(s1), (s2), (s3)	Hour	0 to 23
(s1)+1, (s2)+1, (s3)+1	Minute	0 to 59
(s1)+2, (s2)+2, (s3)+2	Second	0 to 59

Precautions

- Set (s1) to a value less than (s2). If (s1) is set to a value greater than (s2), (s2) is treated as the same value as (s1).

Operation error

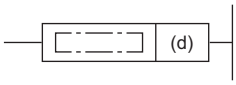
Error code (SD0)	Description
3405H	The value specified by (s1), (s2), or (s3) is outside the following range. 0 to 23
	The value specified by (s1)+1, (s2)+1, (s3)+1, (s1)+2, (s2)+2, or (s3)+2 is outside the following range. 0 to 59

Reading expansion clock data

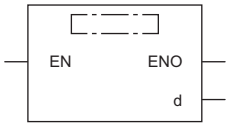
S(P).DATERD

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions read clock data including millisecond from the clock elements in the CPU module.

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions read "year, month, day, hour, minute, second, day of week, and millisecond" from the clock element of the CPU module, 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

- These instructions read clock data and store it in the device even when incorrect click data is set in the CPU module. (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 CPU module and the data read by the S(P).DATERD instruction.)

Operation error

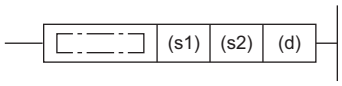
There is no operation error.

Adding expansion clock data

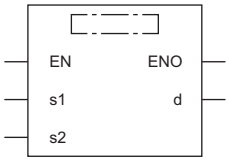
S(P).DATE+

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions add time data.

Ladder	ST*1
	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.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■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□\□, U3E□\(H)G□	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

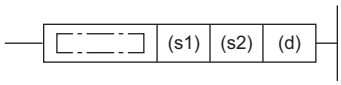
Error code (SD0)	Description
3405H	The data in the device specified by (s1) or (s2) is out of range.

Subtracting expansion clock data

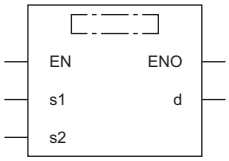
S(P).DATE-

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions subtract time data.

Ladder	ST*1
	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.)

*1 The engineering tool with version "1.035M" or later supports the ST.

■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☐\☐, U3E☐\(H)G☐	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)

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
3405H	The data in the device specified by (s1) or (s2) is out of range.

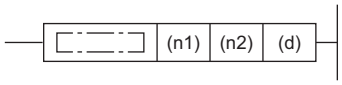
19.2 Timing Check Instructions

Generating timing pulses

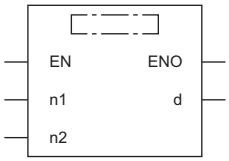
DUTY

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

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.

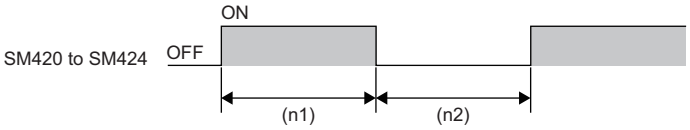
■Applicable devices

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□\□, U3E□\(H)G□	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

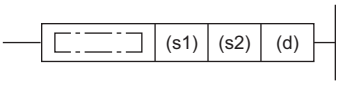
Error code (SD0)	Description
2820H	The device other than SM420 to SM424 is specified by (d).

Measuring time of the specified data

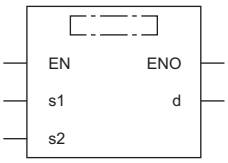
TIMCHK

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	○	○	○	○	—	—	○	○	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—

Processing details

- This instruction measures the on time of the device specified by (s1) 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.

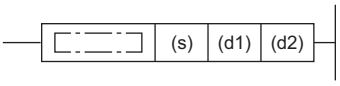
Hour meter

HOURM

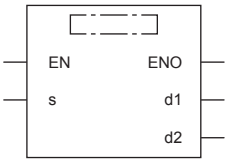
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support this instruction. Use an engineering tool with version "1.020W" or later.

This instruction measures the period of time for which the start contact is ON in units of hour, and turns on the device specified by (d2) when the accumulated ON time reaches the time (16-bit binary data) specified in (s).

Ladder	ST
	ENO:=HOURM(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
HOURM	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Time after which the alarm (d2) is set to on (unit: hour)	0 to 32767	16-bit signed binary	ANY16
(d1)	Device where the measured current value is stored	—	16-bit signed binary	ANY16_ARRAY (Number of elements: 2)
(d2)	Device to be turned on when a timeout occurs (alarm output)	—	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	○	○	○	○	—	—	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○*2	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.
*2 T, ST, C, and FD cannot be used.

Processing details

- This instruction measures the period of time for which the start contact is ON in units of hour, and turns on the device specified by (d2) when the accumulated ON time reaches the time (16-bit binary data) specified in (s).
- In (s), specify the period of time until the device specified by (d2) is turned on in units of hour.
- The measured current value in units of hour is stored in (d1).
- The measured current value of less than one hour (in units of second) is stored in (d1)+1.
- Even after the alarm output specified by (d2) turns ON, the measurement is continued.
- When (d1) reaches the maximum value (32767) and also (d1)+1 reached reaches the maximum value (3599), the measurement is stopped.
- This instruction operates even if a negative value is set in the device specified by (d1).
- Changing clock data (including time adjustment by the daylight-saving time function) does not affect the operation of the HOURM instruction.

Precautions

- In cases such as measuring the ON time from initial value or continuing to the measurement even after the current value reaches the maximum value of 16 bits, clear (d1) to (d1)+1 if (d1) is specified with the device, or clear two elements if (d1) is specified with the label.
- To avoid that timer measurement does not work normally, do not use this instruction in the initial execution type program, interrupt program, fixed scan execution type program, and event execution type program.

Operation error

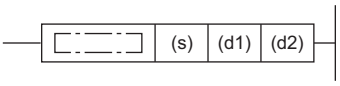
Error code (SD0)	Description
3405H	The range specified by (s) is outside the following range. 0 to 32767

DHOURM

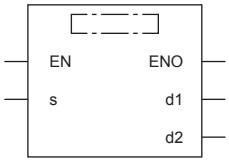
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support this instruction. Use an engineering tool with version "1.020W" or later.

This instruction measures the period of time for which the start contact is ON in units of hour, and turns on the device specified by (d2) when the accumulated ON time reaches the time (32-bit binary data) specified in (s).

Ladder	ST
	ENO:=DHOURM(EN,s,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
DHOURM	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Time after which the alarm (d2) is set to on (unit: hour)	0 to 2147483647	32-bit signed binary	ANY32
(d1)	Device where the measured current value is stored	—	32-bit signed binary	ANY32_ARRAY (Number of elements: 2)
(d2)	Device to be turned on when a timeout occurs (alarm output)	—	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	○	○	○	○	○	○	○	○	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○*2	—	—	—	—	—	—	—	—	—

*1 FX and FY cannot be used.
*2 T, ST, C, and FD cannot be used.

Processing details

- This instruction measures the period of time for which the start contact is ON in units of hour, and turns on the device specified by (d2) when the accumulated ON time reaches the time (32-bit binary data) specified in (s).
- In (s)+1 and (s), specify the period of time until the device specified by (d2) is turned on in units of hour.
- The measured current value in units of hour is stored in (d1)+1 and (d1). ((d1)+1: upper value, (d1): lower value)
- The measured current value of less than one hour (in units of second) is stored in (d1)+2.
- No value is stored in (d1)+3.
- Even after the alarm output specified by (d2) turns ON, the measurement is continued.
- When (d1)+1 and (d1) reaches the maximum value (2147483647) and also (d1)+2 reached reaches the maximum value (3599), the measurement is stopped.
- This instruction operates even if a negative value is set in the device specified by (d1).
- Changing clock data (including time adjustment by the daylight-saving time function) does not affect the operation of the DHOURM instruction.

Precautions

- In cases such as measuring the ON time from initial value or continuing to the measurement even after the current value reaches the maximum value of 32 bits, clear (d1) to (d1)+2 if (d1) is specified with the device, or clear two elements if (d1) is specified with the label.
- To avoid that timer measurement does not work normally, do not use this instruction in the initial execution type program, interrupt program, fixed scan execution type program, and event execution type program.

Operation error

Error code (SD0)	Description
3405H	The range specified by (s) is outside the following range. 0 to 2147483647

20 MODULE ACCESS

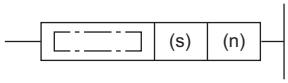
20.1 Module Access Instructions

Performing I/O refresh

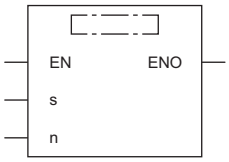
RFS(P)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○*1	—	—	—	—	—	—	—	—	—	—	—
(n)	—	○	○	○	○	—	—	○	○	—	—	—

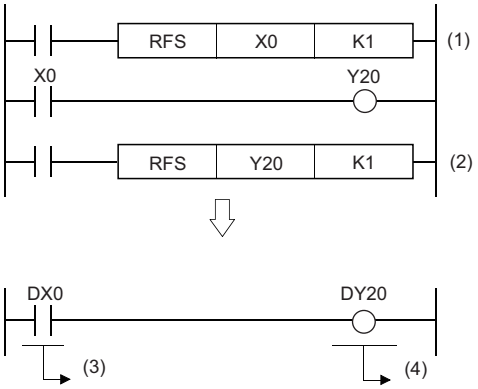
*1 Only X and Y can be used.

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).
- When Process CPUs in redundant mode are used, if inputs (X) or outputs (Y) assigned to a module on an extension base unit are specified from the standby system, no processing is performed.

Ex.

When a program using the RFS instruction is changed to a program using direct access input/output



(1)Refresh X0.
(2)Refresh Y20.
(3)Direct access input
(4)Direct access output

Operation error

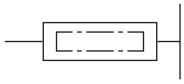
Error code (SD0)	Description
2820H	The range of (n) points from the device specified by (s) exceeds the range of the proximal I/O.

Selecting refresh to be performed

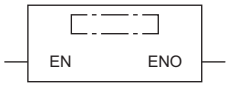
COM(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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*1
  - Link refresh of CC-Link IE Field Network Basic
  - Link refresh of the MELSECNET/H module
  - Intelligent function module refresh*2
  - Refresh of multiple CPU system using the CPU buffer memory (in END processing)
  - Import of input/output outside the group of multiple CPU system
  - 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 and import of input/output outside the group of multiple CPU system	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*1		
Link refresh of CC-Link IE Field Network Basic		
Link refresh of the MELSECNET/H module		
Intelligent function module refresh*2		
Refresh of multiple CPU system using the CPU buffer memory (in END processing)		
Device/label access service processing (communication with the engineering tool, GOT, or other external devices)		

*1 Link refresh of the Simple Motion module (RD77GF) is included.
*2 Buffer memory refresh of the Simple Motion module (RD77GF) and Motion module, and refresh of the RJ71GN11-EIP (EtherNet/IP part) are included.

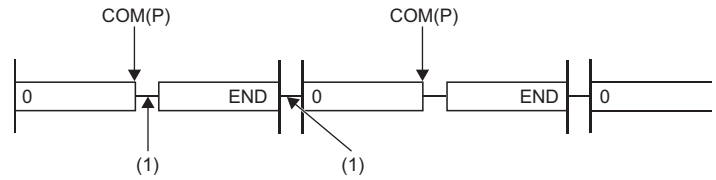
- Select execution or non-execution for b0 to b4, b6, b13, and b15 of SD775 (Selection of refresh processing during the COM instruction execution), and then turn on SM775.

	b15	b14	b13	b12···b7	b6	b5	b4	b3	b2	b1	b0
SD775	1/0	0	1/0	0	1/0	0	1/0	1/0	1/0	1/0	1/0
	(8)		(7)		(6)		(5)	(4)	(3)	(2)	(1)

(1) I/O refresh, import of input/output of non-controlled modules in a multiple CPU system
(2) Link refresh of the CC-Link module
(3) Link refresh of the CC-Link IE Controller Network module and MELSECNET/H module
(4) Intelligent function module refresh*2
(5) Refresh of multiple CPU system using the CPU buffer memory (in END processing)
(6) Link refresh of the CC-Link IE Field Network module*1
(7) Link refresh of CC-Link IE Field Network Basic
(8) Device/label access service processing (communications with the engineering tool, GOT, or other external devices)

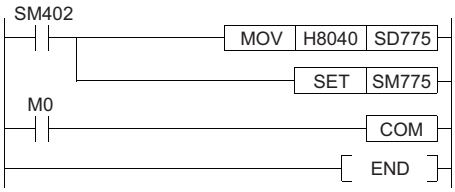
*1 Link refresh of the Simple Motion module (RD77GF) is included.
*2 Buffer memory refresh of the Simple Motion module (RD77GF) and Motion module, and refresh of the RJ71GN11-EIP (EtherNet/IP part) are included.

- 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.



- When Process CPUs in redundant mode are used, if the following processing is executed, no processing is performed.
  - I/O refresh
  - Link refresh of the CC-Link module
  - Intelligent function module refresh

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

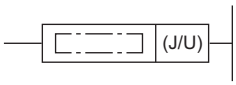
There is no operation error.

Performing module refresh

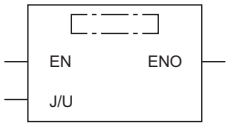
S(P).ZCOM

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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*2	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.)

*2 Buffer memory refresh of the Simple Motion module (RD77GF) and Motion module is performed.

■Applicable devices

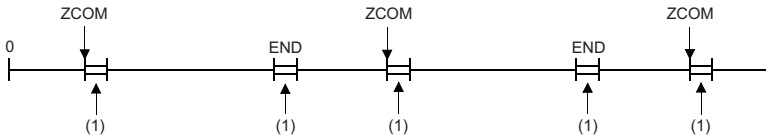
Operand	Bit		Word			Double word		Indirect specification	Constant			Others (J/U)
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(J/U)	—	—	—	—	—	—	—	—	—	—	—	○

- 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 (when refresh settings are configured)*1
  - Link refresh of the MELSECNET/H module (when refresh settings are configured)
  - Link refresh of the CC-Link module (when refresh settings are configured)
  - Intelligent function module refresh (when refresh settings are configured)*2

*1 Link refresh of the Simple Motion module (RD77GF) is included.
*2 Buffer memory refresh of the Simple Motion module (RD77GF) and Motion module, and refresh of the RJ71GN11-EIP (EtherNet/IP part) are included.

Processing details

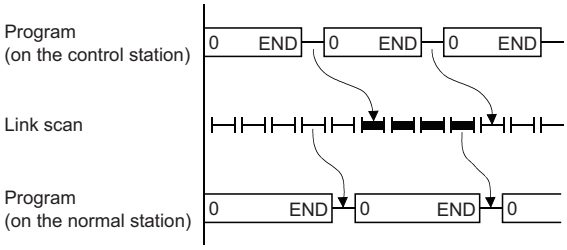
- When executed, the S(P).ZCOM instruction temporarily stops sequence program processing by the CPU module and perform refresh processing for the module specified by (J/U).



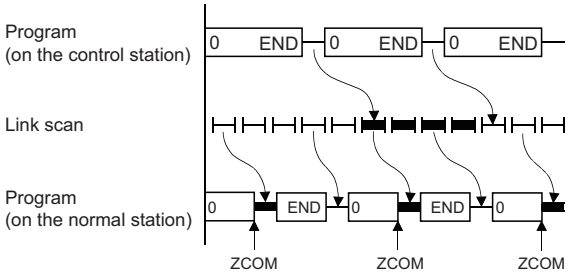
(1) Refresh processing

- The following is applicable when refresh processing of the CC-Link IE Controller Network module or MELSECNET/H module (network between programmable controllers) 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.

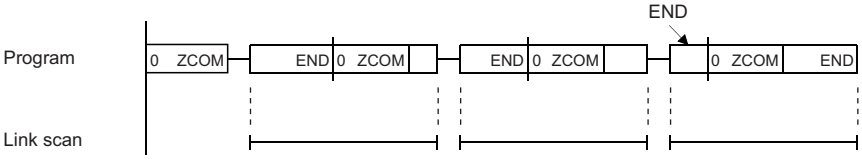
[When the S(P).ZCOM instruction is not used]



[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.



- When the S(P).ZCOM instruction is executed in a redundant system with redundant extension base unit, refer to the following.

☞ Page 1133 COM(P)

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.
2820H	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)

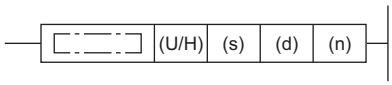
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- FROM(P):

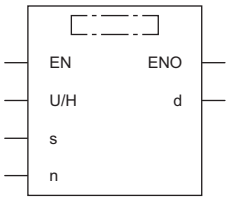
These instructions read n words of data from the buffer memory address in the specified module or another CPU module.

- DFROM(P):

These instructions read n×2 words of data from the buffer memory address in the specified module or another CPU 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 or CPU module	0H to FFH, 3E0H to 3E3H	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.

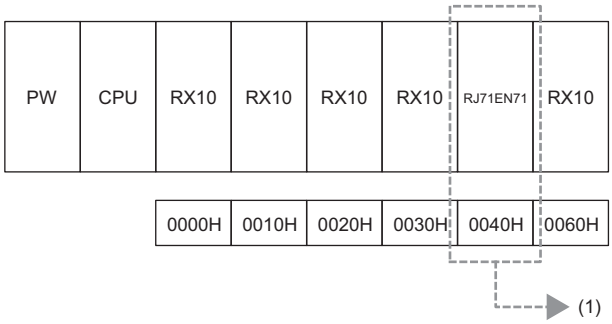
■Applicable devices

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□\□, U3E□\(H)G□	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 CPU module with upper 3 digits when it is represented by 4 hexadecimal digits.



(1) Specify K4 or H4 as the start I/O number of the read-target module.

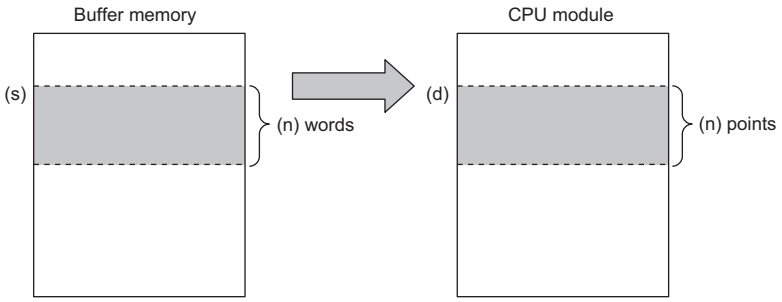
Specify the start I/O numbers of the CPU modules as shown in the table below.

CPU module	Start I/O number
CPU No.1	3E0H
CPU No.2	3E1H
CPU No.3	3E2H
CPU No.4	3E3H

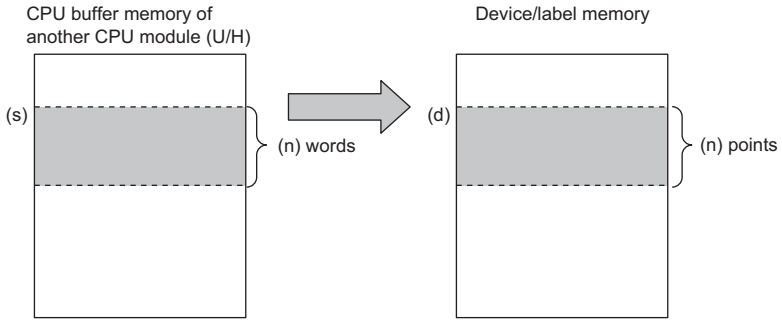
■FROM(P)

- These instructions read (n) words of data from the buffer memory address specified by (s) in the module specified by (U/H) or another CPU module.

• Reading word data from module



• Reading word data from another CPU module

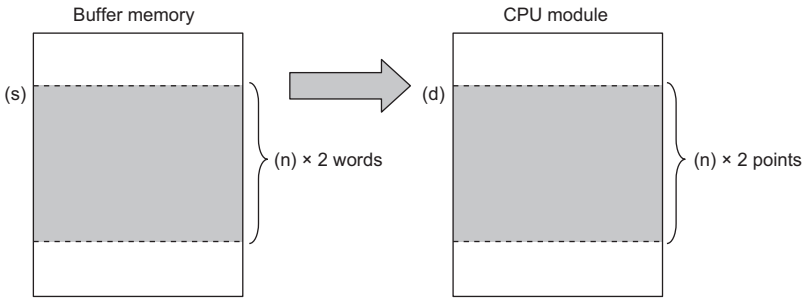


- If the read data (n) is 0, no processing is performed.
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

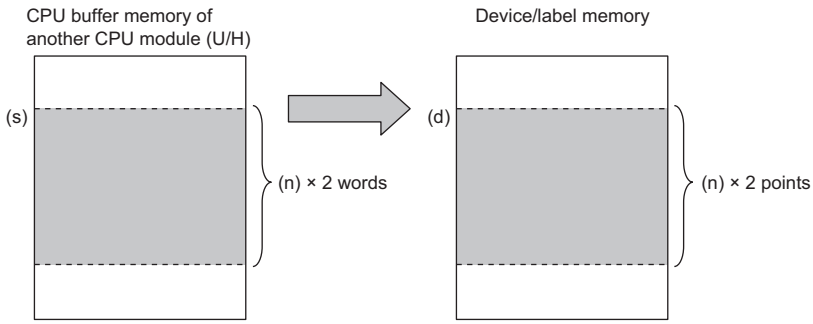
■DFROM(P)

- These instructions read (n)×2 words of data from the buffer memory address specified by (s) in the module specified by (U/H) or another CPU module.

• Reading double word data from module



• Reading double word data from another CPU module



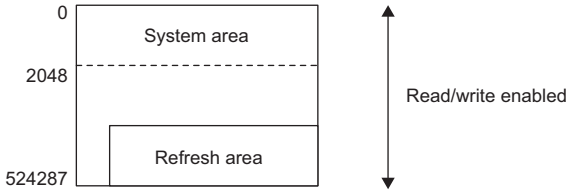
- If the read data (n) is 0, no processing is performed.
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

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 or CPU buffer memory.
	The (n) points of data starting from the address specified by (s) are not within the range of buffer memory or CPU buffer memory. (FROM(P) instruction)
	The (n)×2 points of data starting from the address specified by (s) are not within the range of buffer memory or CPU buffer memory. (DFROM(P) instruction)
3461H	Access to the buffer memory of a module mounted on an extension base unit is attempted from the standby system.

Point

- Module data can also be read using the module access device. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- 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.



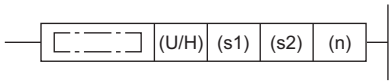
- A CPU buffer memory access device can be used to read data from the CPU buffer memory. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- The FROM(P) and DFROM(P) instructions can read data from the buffer memory address with a capacity of 64K or less. To read data from the buffer memory address with a capacity exceeding 64K, use the FROMD(P) or DFROMD(P) instruction. (☞ Page 1146 FROMD(P), DFROMD(P))

Writing 1-word/2-word data to a module (16-bit specification)

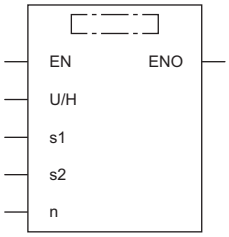
TO(P), DTO(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- TO(P): These instructions write the n points of data from the specified device to the buffer memory in the module/host CPU module.
- DTO(P): These instructions write the n×2 points of data from the specified device to the buffer memory in the module/host CPU module.

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 or CPU module	0H to FFH, 3E0H to 3E3H	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.

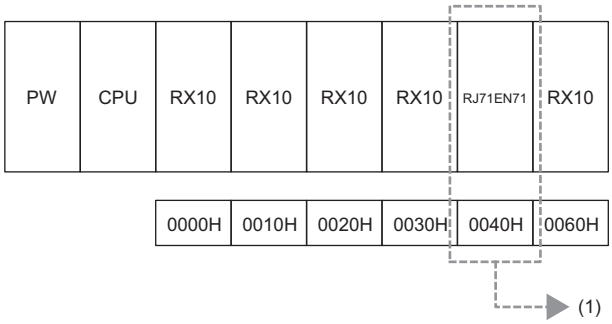
■Applicable devices

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□\□, U3E□\(H)G□	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 CPU module with upper 3 digits when it is represented by 4 hexadecimal digits.



(1) Specify K4 or H4 as the start I/O number of the write-target module.

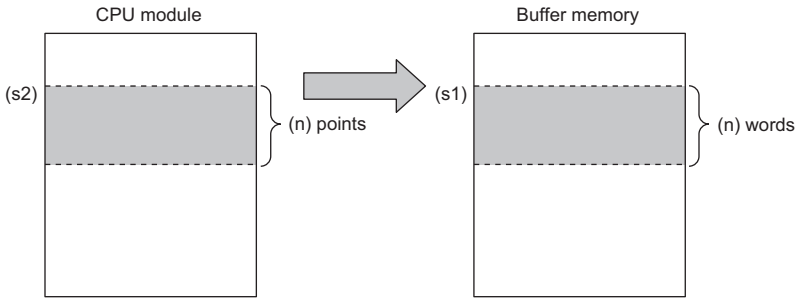
Specify the start I/O numbers of the CPU modules as shown in the table below.

CPU module	Start I/O number
CPU No.1	3E0H
CPU No.2	3E1H
CPU No.3	3E2H
CPU No.4	3E3H

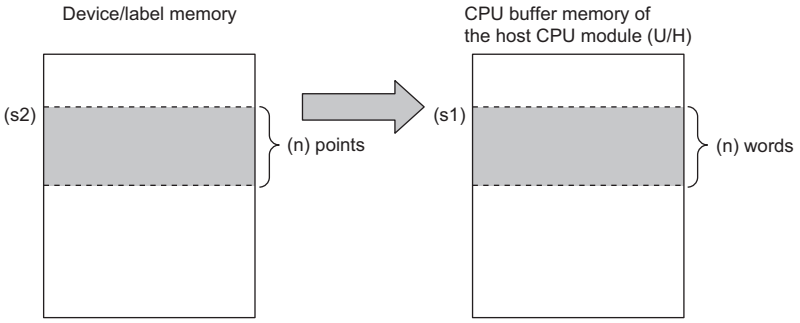
■TO(P)

- These instructions write 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 CPU module specified by (U/H).

• Writing word data to a module

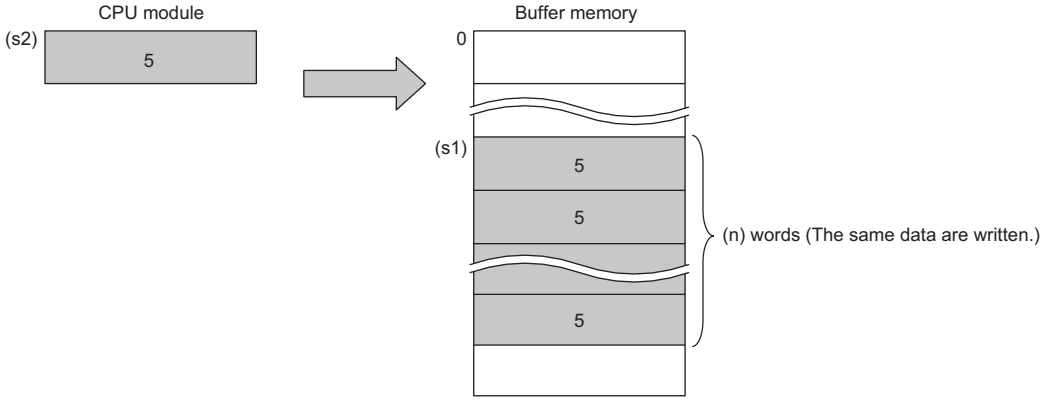


• Writing word data to the host CPU module

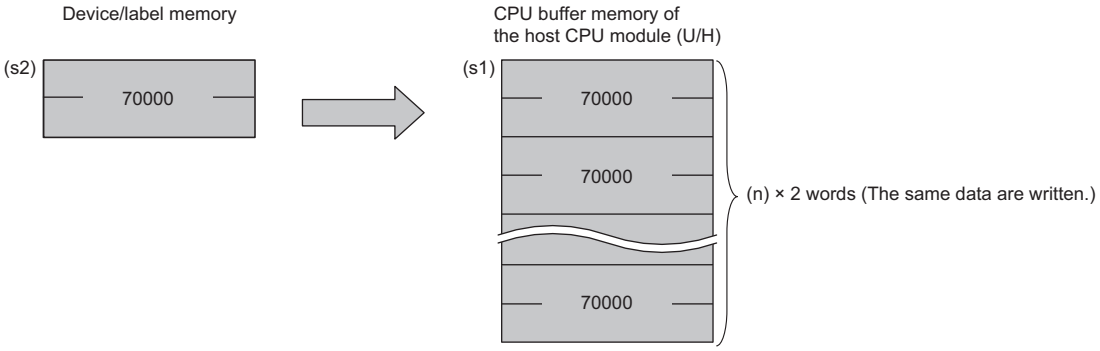


- If a constant is specified in (s2), the instructions write the same data (the value in the device specified by (s2)) to the (n) words from the specified buffer memory address.

• Writing word data to a module



• Writing word data to the host CPU module

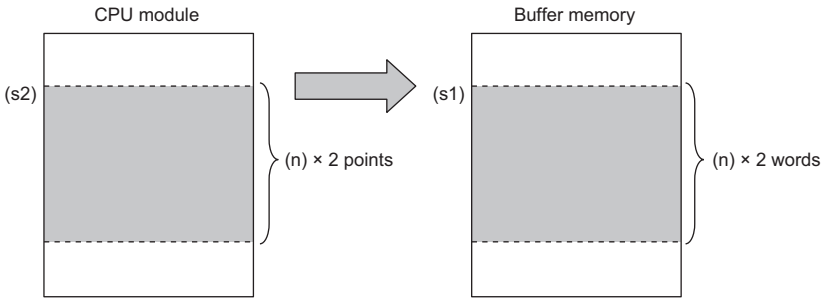


- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

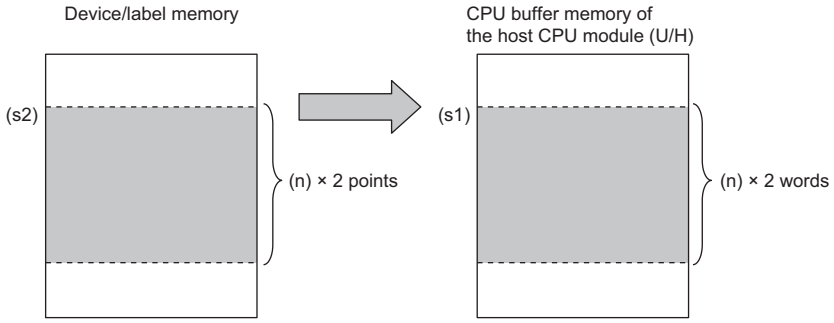
■DTO(P)

- These instructions write 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 CPU module specified by (U/H).

• Writing double word data to a module

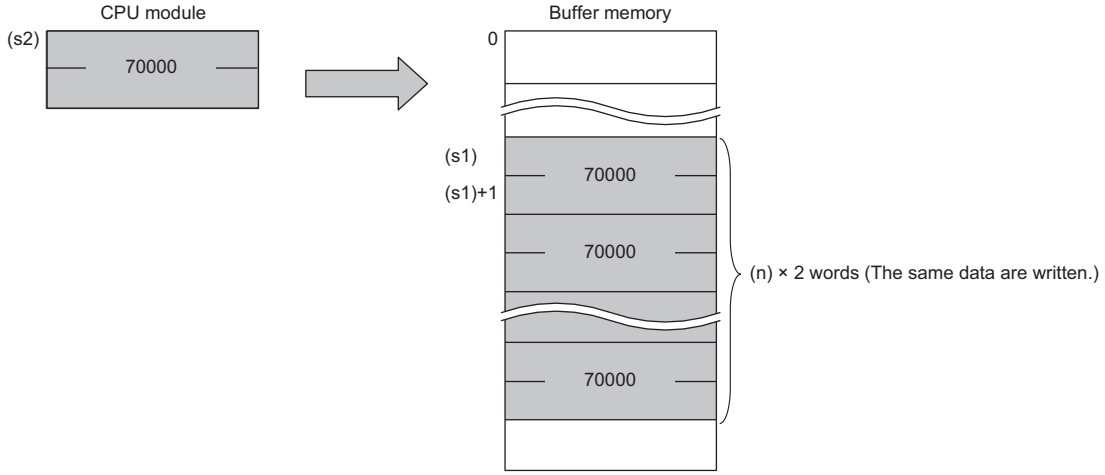


• Writing double word data to the host CPU module

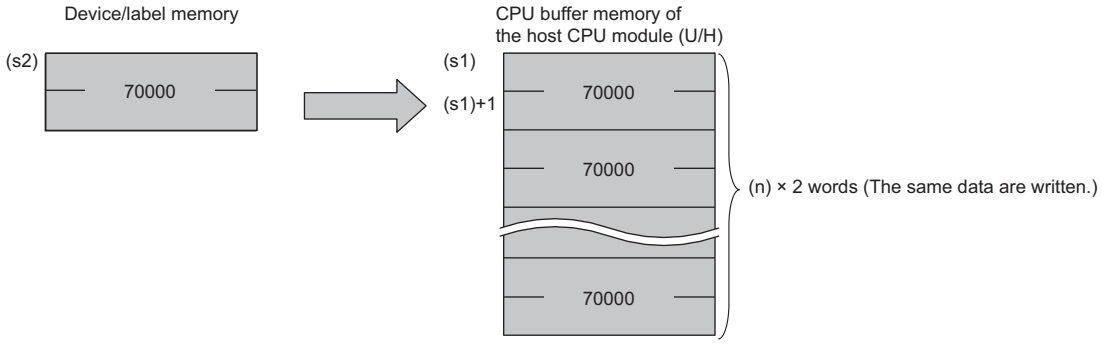


- If a constant is specified in (s2), the instructions write the same data (the value in the device specified by (s2)) to the (n)×2 words from the specified buffer memory address.

• Writing double word data to a module



• Writing double word data to the host CPU module



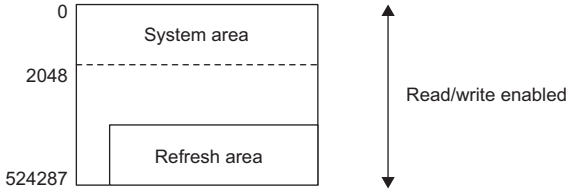
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

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 or CPU buffer memory.
	The (n) points of data starting from the address specified by (s1) are not within the range of buffer memory or CPU buffer memory. (TO(P) instruction)
	The (n)×2 points of data starting from the address specified by (s1) are not within the range of buffer memory or CPU buffer memory. (DTO(P) instruction)
3461H	Access to the buffer memory of a module mounted on an extension base unit is attempted from the standby system.

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.



- A CPU buffer memory access device can be used to write data to the CPU buffer memory. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- The TO(P) and DTO(P) instructions can write data to the buffer memory address with a capacity of 64K or less. To write data to the buffer memory address with a capacity exceeding 64K, use the TOD(P) or DTOD(P) instruction. (☞ Page 1150 TOD(P), DTOD(P))

Reading 1-word/2-word data from another module (32-bit specification)

FROMD(P), DFROMD(P)

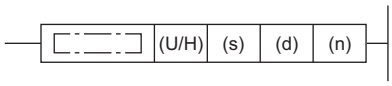
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- FROMD(P):

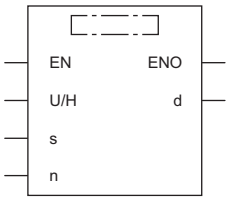
These instructions read n words of data from the buffer memory address in the specified module or another CPU module.

- DFROMD(P):

These instructions read n×2 words of data from the buffer memory address in the specified module or another CPU 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 or CPU module	0H to FFH, 3E0H to 3E3H	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.

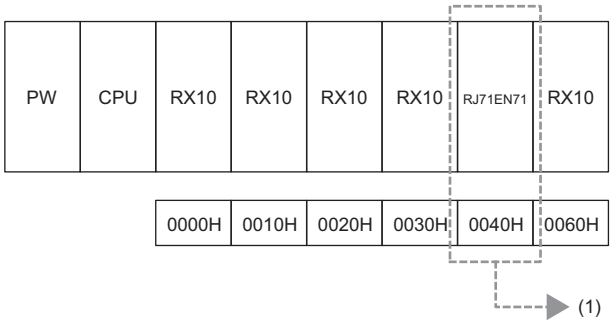
■Applicable devices

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□\□, U3E□\(H)G□	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 CPU module with upper 3 digits when it is represented by 4 hexadecimal digits.



(1) Specify K4 or H4 as the start I/O number of the read-target module.

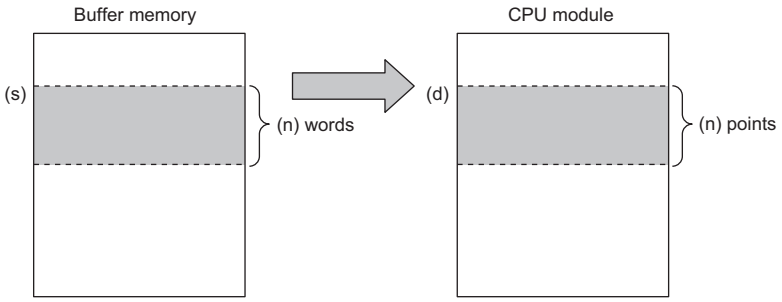
Specify the start I/O numbers of the CPU modules as shown in the table below.

CPU module	Start I/O number
CPU No.1	3E0H
CPU No.2	3E1H
CPU No.3	3E2H
CPU No.4	3E3H

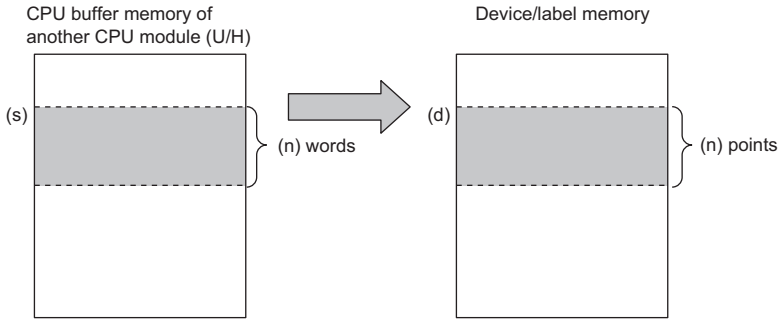
■FROMD(P)

- These instructions read (n) words of data from the buffer memory address specified by (s) in the module specified by (U/H) or another CPU module.

• Reading word data from module



• Reading word data from another CPU module

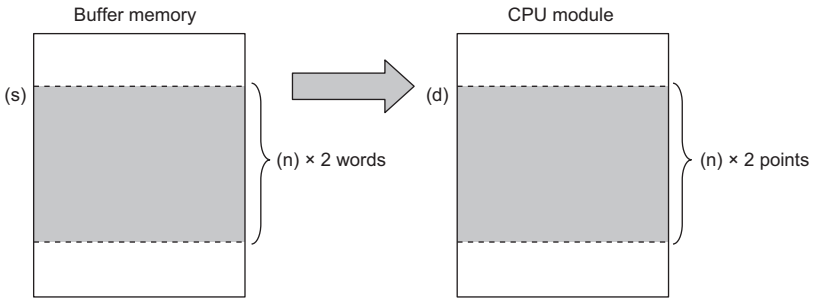


- If the read data (n) is 0, no processing is performed.
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

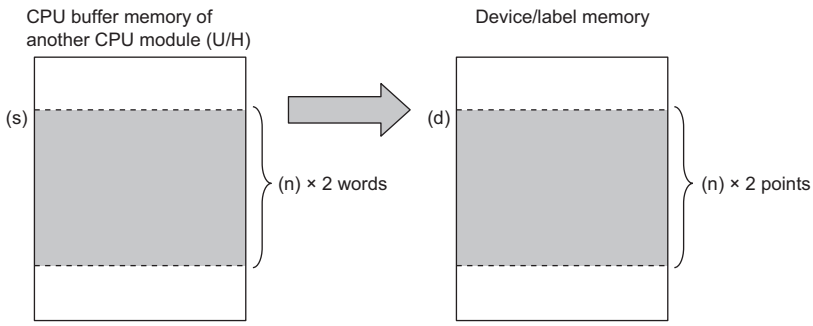
■DFROMD(P)

- These instructions read (n)×2 words of data from the buffer memory address specified by (s) in the module specified by (U/H) or another CPU module.

• Reading double word data from module



• Reading double word data from another CPU module



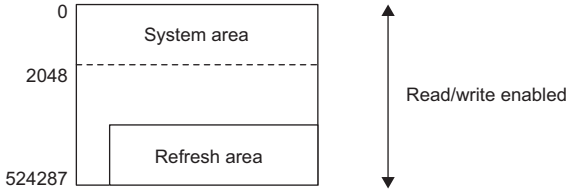
- If the read data (n) is 0, no processing is performed.
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

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 or CPU buffer memory.
	The (n) points of data starting from the address specified by (s) are not within the range of buffer memory or CPU buffer memory. (FROMD(P) instruction)
	The (n)×2 points of data starting from the address specified by (s) are not within the range of buffer memory or CPU buffer memory. (DFROMD(P) instruction)
3461H	Access to the buffer memory of a module mounted on an extension base unit is attempted from the standby system.

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.



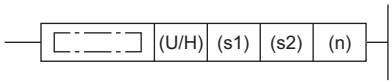
- A CPU buffer memory access device can be used to read data from the CPU buffer memory. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- The FROMD(P) and DFROMD(P) instructions can read data from the buffer memory address with a capacity exceeding 64K.

Writing 1-word/2-word data to a module (32-bit specification)

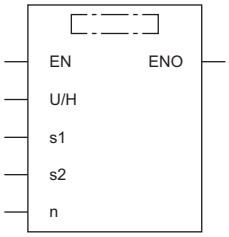
TOD(P), DTOD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- TOD(P): These instructions write the n points of data from the specified device to the buffer memory in the module/host CPU module.
- DTOD(P): These instructions write the n×2 points of data from the specified device to the buffer memory in the module/host CPU module.

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 or CPU module	0H to FFH, 3E0H to 3E3H	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.

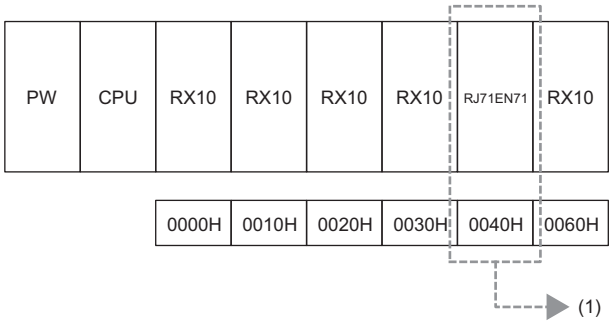
■Applicable devices

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□\□, U3E□\(H)G□	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 CPU module with upper 3 digits when it is represented by 4 hexadecimal digits.



(1) Specify K4 or H4 as the start I/O number of the write-target module.

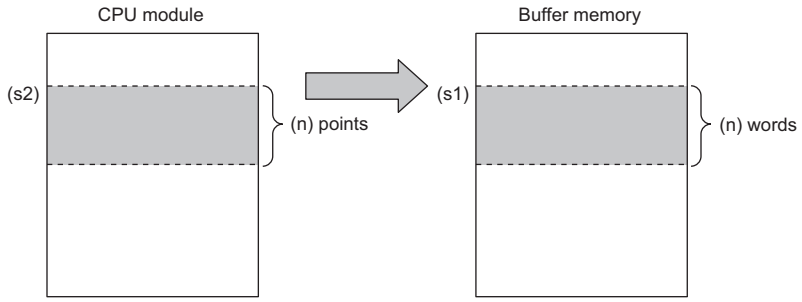
Specify the start I/O numbers of the CPU modules as shown in the table below.

CPU module	Start I/O number
CPU No.1	3E0H
CPU No.2	3E1H
CPU No.3	3E2H
CPU No.4	3E3H

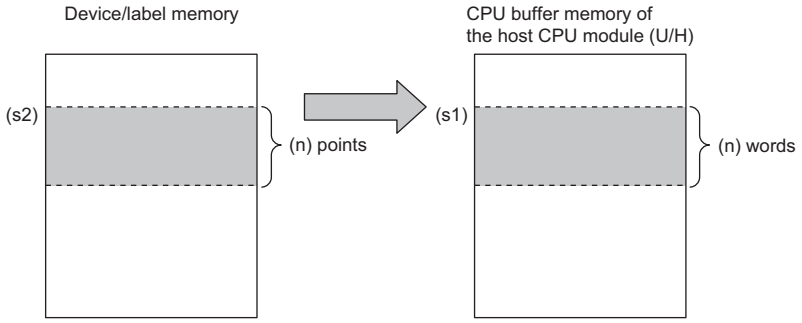
■TOD(P)

- These instructions write 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 CPU module specified by (U/H).

• Writing word data to a module

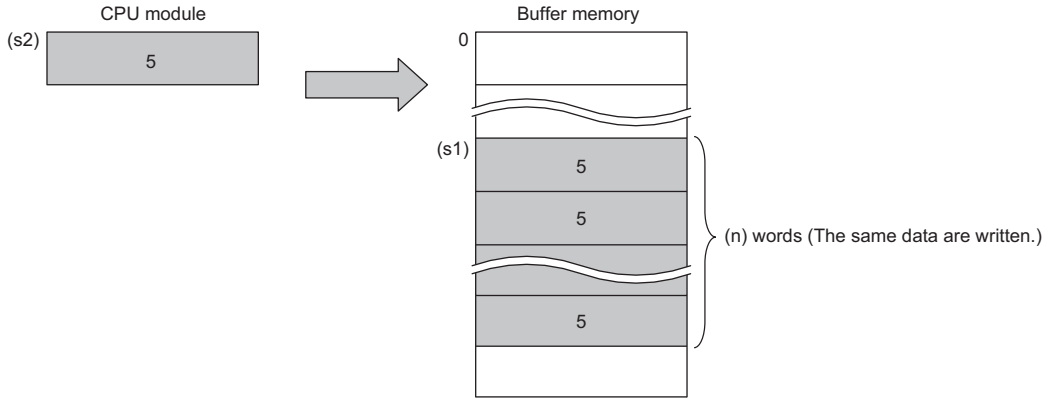


• Writing word data to the host CPU module

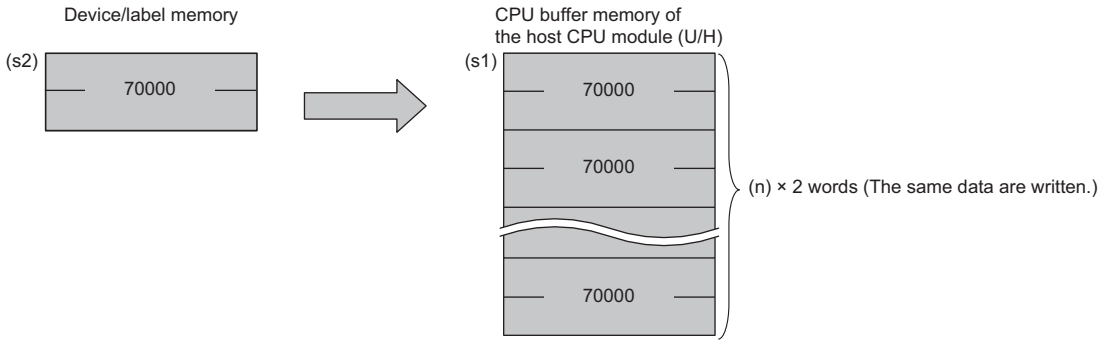


- If a constant is specified in (s2), the instructions write the same data (the value in the device specified by (s2)) to the (n) words from the specified buffer memory address.

• Writing word data to a module



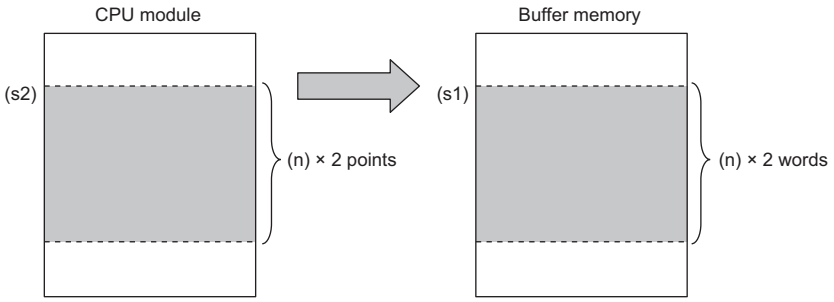
• Writing word data to the host CPU module



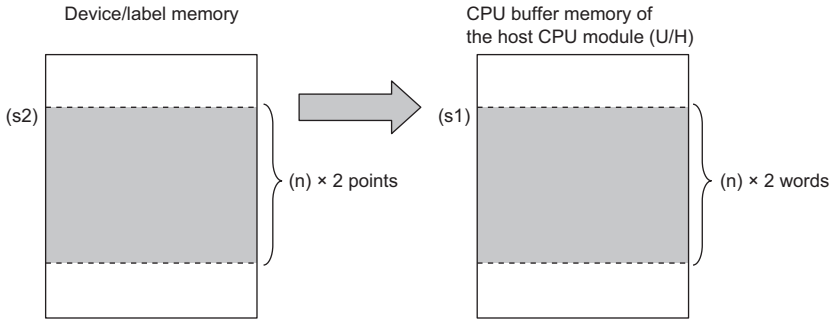
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

■DTOD(P)

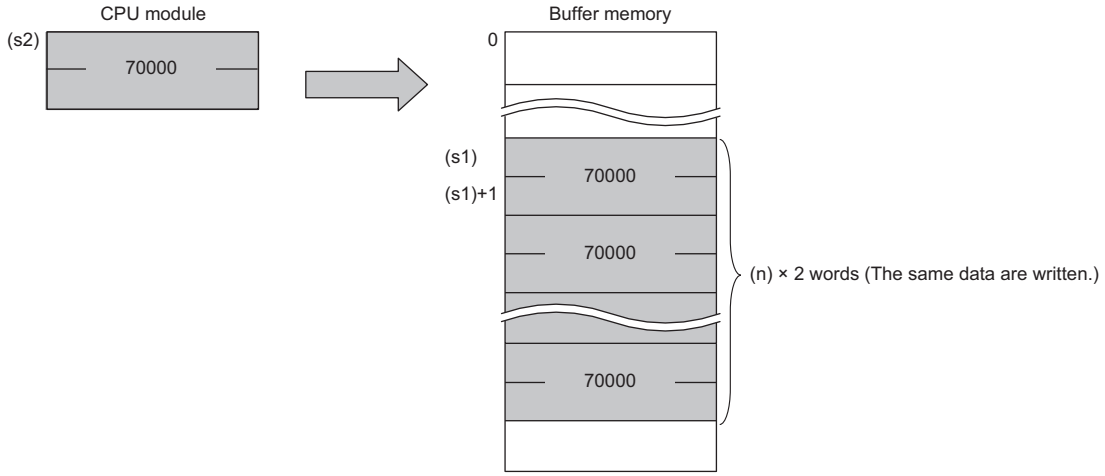
- These instructions write the (n)×2 points of data from the device specified by (s2) to the buffer memory address specified by (s1) and later in the buffer memory in the module or host CPU module specified by (U/H).
  - Writing double word data to a module



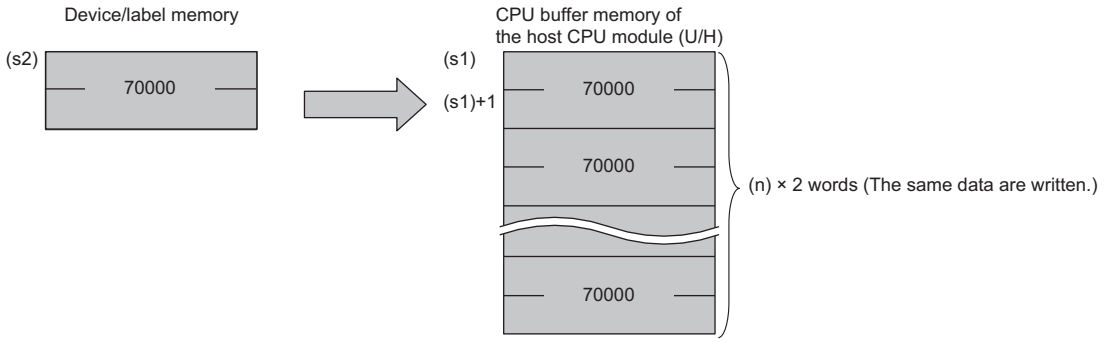
  - Writing double word data to the host CPU module



- 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 the host CPU module



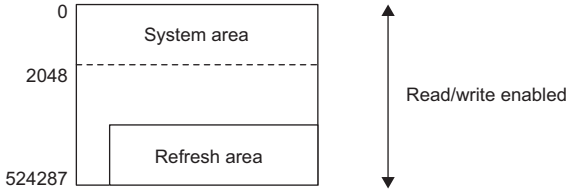
- An instruction which has been executed will result in non-processing if it fails to access the target module because the module is faulty or busy in processing.

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 or CPU buffer memory.
	The (n) points of data starting from the address specified by (s1) are not within the range of buffer memory or CPU buffer memory. (TOD(P) instruction)
	The 2×(n) points of data from the address specified by (s1) is outside the range of buffer memory or CPU buffer memory. (DTOD(P) instruction)
3461H	Access to the buffer memory of a module mounted on an extension base unit is attempted from the standby system.

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.



- A CPU buffer memory access device can be used to write data to the CPU buffer memory. (📖 MELSEC iQ-R CPU Module User's Manual (Application))
- The TOD(P) and DTOD(P) instructions can write data to the buffer memory address with a capacity exceeding 64K.

Reading the module model name

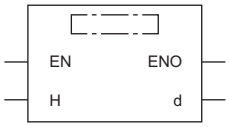
TYPERD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

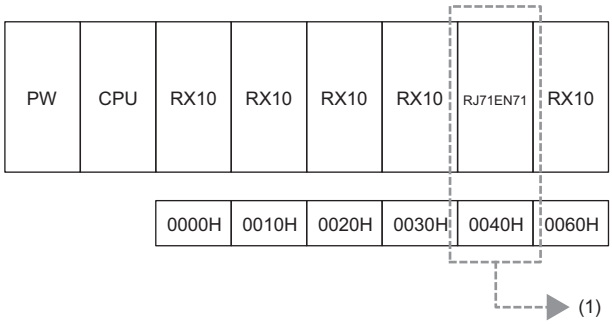
Operand	Description	Range	Data type	Data type (label)
(H)	Start I/O number (first three digits in four-digit hexadecimal representation) of a model read target module	0H to FFH, 3E0H to 3E3H	16-bit unsigned binary	ANY16
(d)	(d)+0: Instruction execution result (d)+1 to (d)+9: Module name	—	Word	ANY16_ARRAY (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□\□, U3E□\(H)G□	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.
  - CPU module
  - 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 K4 or H4 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 CPU modules as shown in the table below.

CPU module	Start I/O number
CPU No.1	3E0H
CPU No.2	3E1H
CPU No.3	3E2H
CPU No.4	3E3H

- 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	Character 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 module on an extension base unit is specified from the standby system in a redundant system with redundant extension base unit.		
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
CPU module	R04CPU
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 character 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 character string consisting of a combination of "character string indicating the module type" and "character string indicating the number of points" is stored.

Ex.

The following table lists character string examples that are stored.

Target module	Character 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 character string examples that indicate the numbers of points.

Number of points	Character 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 reading the module name of the RnENCPU (CPU part) by specifying the I/O number, the following character string is stored.

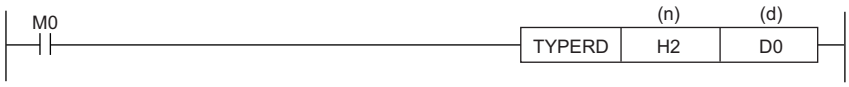
Ex.

The following table lists character string examples that are stored.

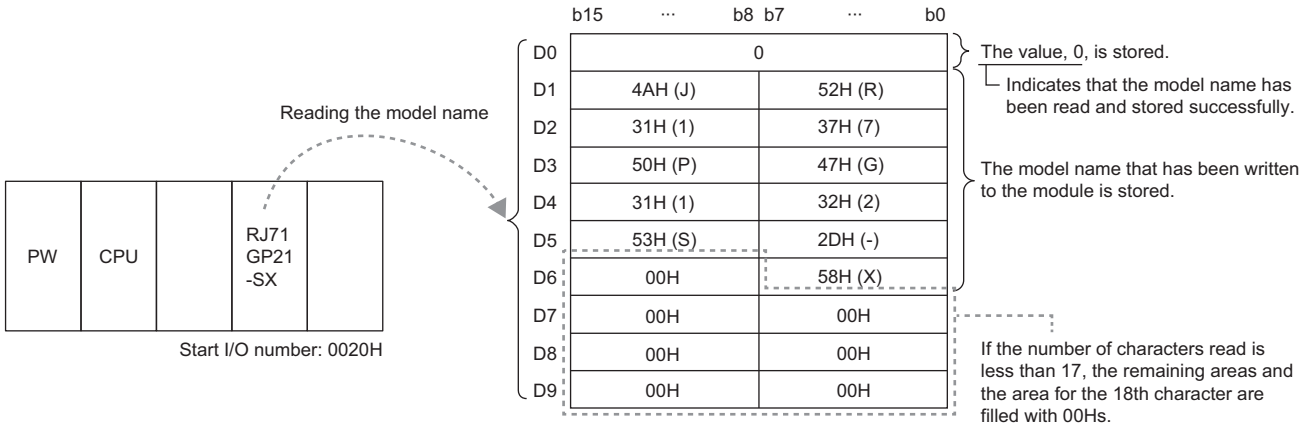
I/O number of the module	Character string example stored
3E00H	R120ENCPU
0000H	_RJ71EN71(E+IEF)*1

*1 For the RnENCPU (network part), the character string of the module selected on the Element Selection window for the Module Configuration of the engineering tool is stored.

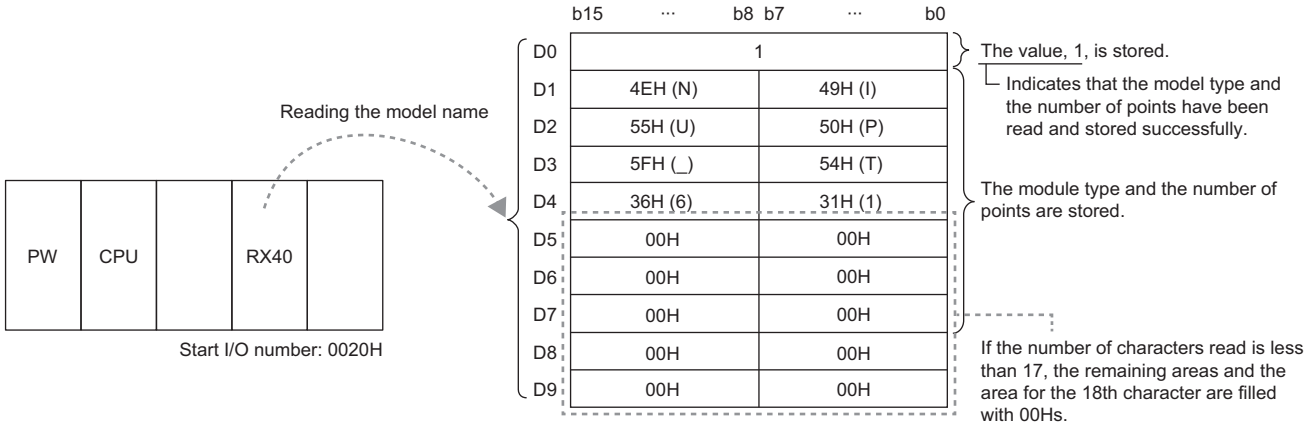
- In a multiple CPU system, the module name can also be read by specifying the module controlled by the CPU module of another CPU.
- In the following program example, when M0 turns on, the module name of the module mounted at I/O number 0020H is stored in D0 and later.



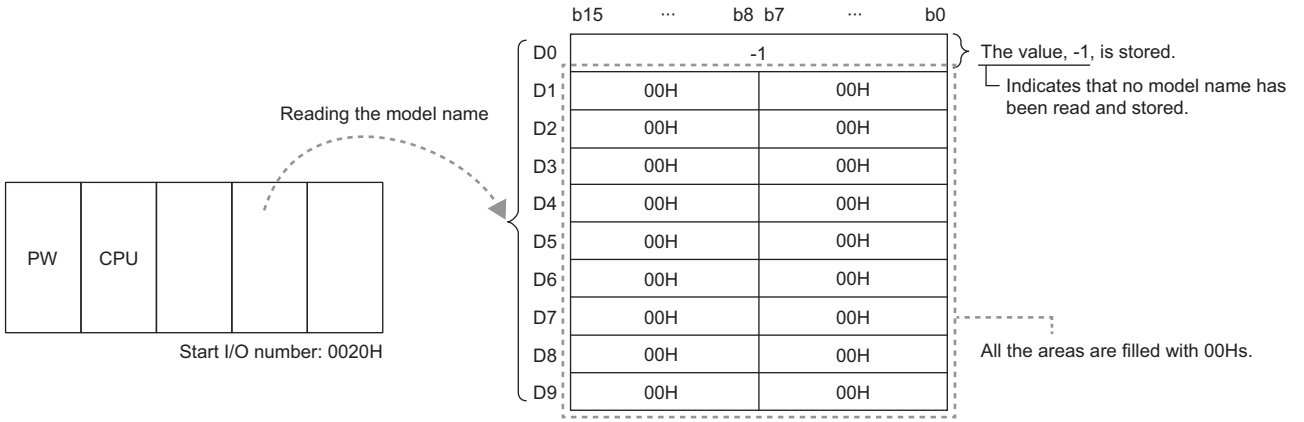
• Module having a module name (example: RJ71GP-SX)



• Module having no module name (example: RX40)



• Empty slot



Operation error

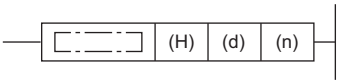
Error code (SD0)	Description
2800H	The value set to (H) is out of the range, 0 to FFH and 3E0 to 3E3H.
2810H	Communications with the read target module is disabled due to a failure of the module.

Reading module specific information

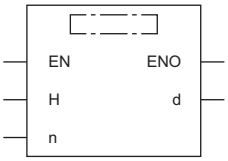
UNIINFRD(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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

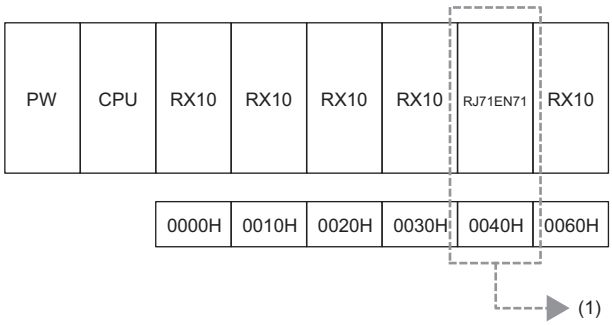
*1 When specifying setting data by using a label, define an array to secure enough operation area and specify an element of the array label.

■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□\□, U3E□\(H)G□	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 K4 or H4 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*3
	b5		
	b6		
	b7		
	b8	Series type	010: MELSEC-Q series module 011: MELSEC iQ-R series module 111: Unknown
	b9		
	b10		
	b11	Group number	000: CPU No.1 control 001: CPU No.2 control 010: CPU No.3 control 011: CPU No.4 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	Inter-module synchronization	00: Not subject to synchronization 01: Preparing for synchronization 10: Synchronized 11: Synchronization error
	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	Online module change	0: Other state (A state other than the states listed below) 1: One of the following states • Online module change being performed (except the state where a module selection is in progress) • A restart of the module in progress • In a redundant system, a state where the information of the module on an extension base unit is attempted to be read from the standby system
	b14	Reserved	0: Fixed
	b15	Module access	0: Access disabled*1 1: Access enabled

*1 When the module is being mounted or removed, or access to a module on an extension base unit is attempted from the standby system in a redundant system with redundant extension base unit

*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.

*3 When the I/O number of the RnENCPU (network part) is specified, this information is read and stored.

- In the following program example, when M0 turns on, the information on the module mounted at I/O number 0040H is stored in D0 and later.

PW	CPU	RX10	RX10	RX10	RX10	RJ71EN71	RX10
		0000H	0010H	0020H	0030H	0040H	0060H

[Result of reading]

- Module information of the RJ71EN71 is read.

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D0	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	1
	(1)	(2)	(3)			(4)			(5)				(6)			

(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

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D1	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
	(1)	(2)	(3)	(4)	(5)	(6)	(7)		(8)		(9)			(10)	(11)	

(1) Module access enabled
(2) Fixed to 0
(3) Online module change not being performed and instruction executed by the control system
(4) Fixed to 0
(5) No fuse blown
(6) Fixed to 0
(7) Fixed to 0
(8) Inter-module synchronized
(9) Fixed to 0
(10)Module ready
(11)Normal operation (no module error)

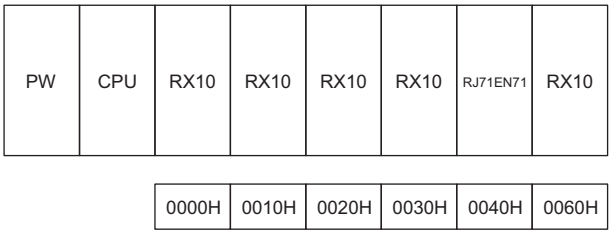
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	(1)	(2)														

(1) A 32-point module is connected in the latter 16 points.
(2) All 0s because information is stored in D0 and D1.

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
D3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	(1)															

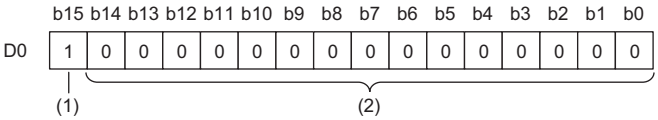
(1) All 0s because information is stored in D0 and D1.

- 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 0050H is stored in D0 and later.

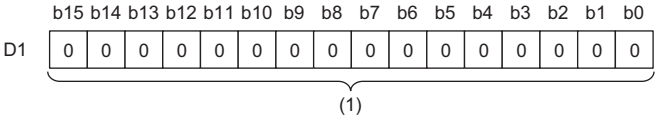


[Result of reading]

- Only the module mount status is on because the I/O number in the latter 16 points of the RJ71EN71 is specified by (H).

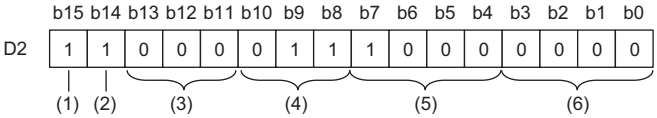


(1) A 32-point module is connected in the latter 16 points.
(2) All 0s because the latter 16 points of a 32-point module is specified in n1.

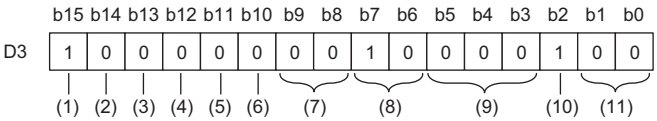


(1) All 0s because the latter 16 points of a 32-point module is specified in n1.

- 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) Online module change not being performed and instruction executed by the control system
(4) Fixed to 0
(5) No fuse blown
(6) Fixed to 0
(7) Fixed to 0
(8) Inter-module synchronized
(9) Fixed to 0
(10)Module ready
(11)Normal operation (no module error)

Operation error

Error code (SD0)	Description
2800H	The value set to (H) is out the range, 0 to FFH.
3405H	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.

21 PARAMETER SETTING OPERATION

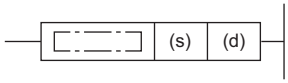
21.1 Routing Information Instructions

Reading routing information

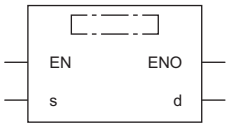
S(P).RTREAD

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These instructions read the data of the specified transfer destination network number from the routing information set in parameter.

Ladder	ST
	ENO:=S_RTREAD(EN,s,d); ENO:=SP_RTREAD(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
S.RTREAD	
SP.RTREAD	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Transfer destination network number	1 to 239	16-bit signed binary	ANY16
(d)	Start device for storing the read data	—	Word	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	—	—	—	○	○	—	—	—
(d)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions read the data of the transfer destination network number specified by (s) from the routing information set in parameter, and store the information in the device specified by (d) and later.
- If the data of the transfer destination network number specified by (s) is not set in parameter, 0 is stored in the device specified by (d) and later.
- The following figure shows the data stored in the device specified by (d) and later.

Start device	Item	Range
(d)	Relay network number	1 to 239
(d)+1	Relay station number	Refer to the following table.
(d)+2	Dummy	—

- The specification ranges of (d)+1 are as follows.

Network type	Specification range
MELSECNET/H	1 to 64
CC-Link IE Controller Network	1 to 120
CC-Link IE Field Network (master station)	Fixed to 125
CC-Link IE Field Network (local station)	1 to 120
CC-Link IE TSN (master station)	Fixed to 125
CC-Link IE TSN (local station)	1 to 120

Operation error

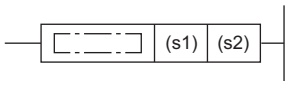
Error code (SD0)	Description
3405H	The value set to (s) is out of the range, 1 to 239.

Registering routing information

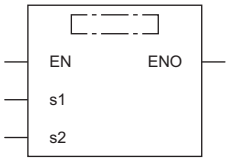
S(P).RTWRITE

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These instructions write routing information in the area with the specified transfer destination network number.

Ladder	ST
	ENO:=S_RTWRITE(EN,s1,s2); ENO:=SP_RTWRITE(EN,s1,s2);

FBD/LD



■Execution condition

Instruction	Execution condition
S.RTWRITE	
SP.RTWRITE	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Transfer destination network number	1 to 239	16-bit signed binary	ANY16
(s2)	Start device where the write data is stored	—	Word	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	○	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—

Processing details

- These instructions write the routing information stored in the device specified by (s2) and later to the area with the transfer destination network number specified by (s1).
- The following figure shows the data stored in the device specified by (s2) and later.

Start device	Item	Range
(s2)	Relay network number	1 to 239
(s2)+1	Relay station number	Refer to the following table.
(s2)+2	Dummy	—

- The specification ranges of (s2)+1 are as follows.

Network type	Specification range
MELSECNET/H	1 to 64
CC-Link IE Controller Network	1 to 120
CC-Link IE Field Network (master station)	Fixed to 125
CC-Link IE Field Network (local station)	1 to 120
CC-Link IE TSN (master station)	Fixed to 125
CC-Link IE TSN (local station)	1 to 120

- If the data of the transfer destination network number specified by (s1) has already been set in parameter, the data is overwritten with the data stored in the device specified by (s2) and later.
- If the data in the device areas specified by (s2) to (s2)+2 are all 0, the data of the transfer destination network number specified by (s1) is deleted from parameter.

Operation error

Error code (SD0)	Description
3405H	The value set to (s1) is out of the range, 1 to 239.
	Any of the data in the device areas specified by (s2) and later exceeds the following range. • (s2): 1 to 239 • (s2)+1: 1 to 120, 125
	The total number of routing information registered in parameter of the network module and registered by using the RTWRITE instruction exceeds 238.
	A transfer destination network number which is not registered in parameter is specified as a deletion target.
	A zero is specified in only either of the device areas specified by (s2) and (s2)+1.

22 CPU MODULE DATA LOGGING FUNCTION

22.1 Logging Instructions

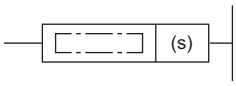
Setting trigger logging

LOGTRG

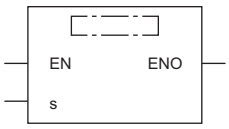
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- The R00CPU does not support this instruction.

This instruction generates a trigger condition for the specified logging setting number in trigger logging.

Ladder	ST
	ENO:=LOGTRG(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
LOGTRG	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Logging setting number	1 to 10	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	$	
(s)	—	—	○	—	○	—	—	○	○	—	—	—

Processing details

- This instruction generates a trigger for trigger logging with the logging setting number specified by (s).
- Specify a value 1 to 10 in (s).
- The LOGTRG instruction turns on the special relay (logging trigger) with the logging setting number in the device specified by (s), executes trigger logging for the specified number of records, latches data, and stops trigger logging.
- The instruction is enabled when "When trigger instruction executed" in the "Trigger condition".
- Even if the LOGTRG instruction is executed, no processing is performed in the following cases.
  - A logging setting number in which an item other than "When trigger instruction executed" is selected in the "Trigger condition" is specified.
  - A logging setting number with no setting is specified.
  - A logging setting number specifying the execution of continuous logging is specified.
  - Another LOGTRG instruction is executed without executing the LOGTRGR instruction after a LOGTRG instruction was executed once.

Precautions

If the LOGTRG instruction is used in an interrupt program, trigger conditions may not occur.

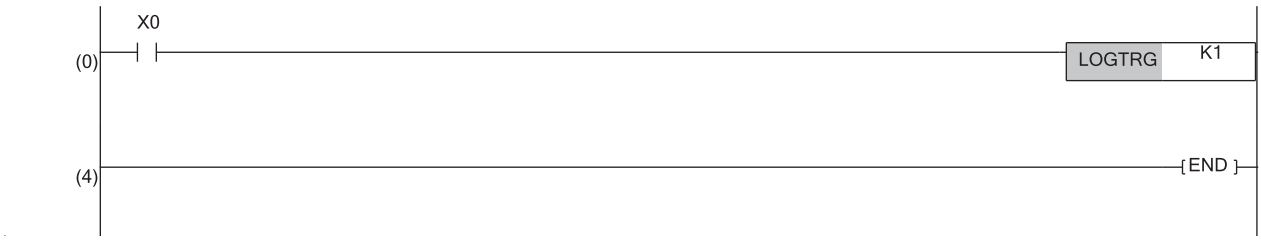
Operation error

Error code (SD0)	Description
3405H	The value set to (s) is out of the range, 1 to 10.

22

■Program example

The program that executes the LOGTRG instruction to the setting No.1 when X0 turns on



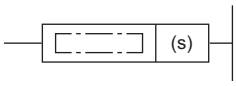
Resetting trigger logging

LOGTRGR

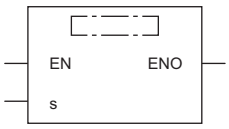
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- The R00CPU does not support this instruction.

This instruction resets the trigger condition of the specified logging setting number.

Ladder	ST
	ENO:=LOGTRGR(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
LOGTRGR	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Logging setting number	1 to 10	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	$	
(s)	—	—	○	—	○	—	—	○	○	—	—	—

Processing details

- This instruction resets the LOGTRG instruction of the logging setting number specified by (s). The instruction disables the LOGTRG instruction of the specified trigger logging setting number.
- The LOGTRGR instruction turns off the special relays (logging completion, logging trigger, and after logging trigger) with the logging setting number in the device specified by (s).
- If the instruction is executed while buffer data is saved to an SD memory card, the execution of the instruction is made to wait until all data is saved completely.

Precautions

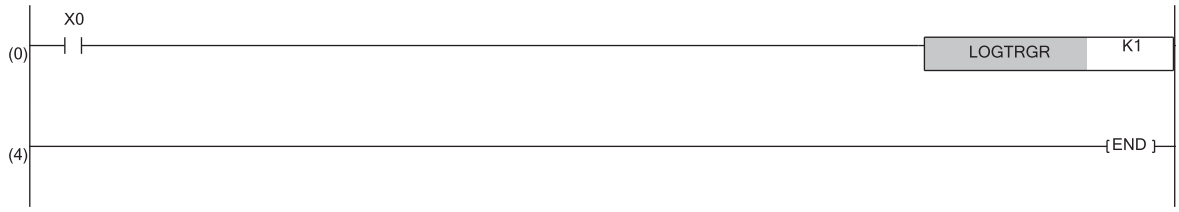
If the LOGTRG instruction is used in an interrupt program, trigger conditions may not occur.

Operation error

Error code (SD0)	Description
3405H	The value set to (s) is out of the range, 1 to 10.

■Program example

The program that resets the LOGTRG instruction for the setting No.1 when X0 turns on



23 RECORDING FUNCTION

23.1 Data Collection Instruction

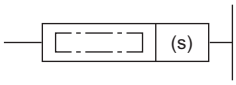
Setting data collection trigger

DATATRG

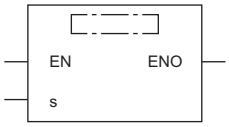
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- The R00CPU, R01CPU, and R02CPU do not support this instruction.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "50" or later support this instruction. Use an engineering tool with version "1.065T" or later. Note that there are restrictions on the production information of the CPU module. For the compatible production information, refer to the MELSEC iQ-R CPU Module User's Manual (Application).
- The RnSFCPU (standard) with firmware version "23" or later supports this instruction. Use an engineering tool with version "1.070Y" or later.

This instruction collects data for the specified setting number of the recording function.

Ladder	ST
	ENO:=DATATRG(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
DATATRG	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Setting number	1 to 4	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	$	
(s)	—	—	○	—	○	—	—	○	○	—	—	—

Processing details

- In the recording function, this instruction collects data for the setting number specified by (s).
- Set a value 1 to 4 in (s).
- This instruction is enabled when "Trigger Instruction" has been set to "Sampling Interval Setting" in "Recording Setting" of the engineering tool.
- No processing is performed when
  - The setting number that is not configured is specified.
  - The sampling method that is set in "Sampling Interval Setting" of "Recording Setting" for the specified setting number is not "Trigger Instruction".
  - The DATATRG is executed again to the setting number to which the DATATRG is already being executed.
  - The DATATRG is executed while the recording startup trigger of the specified setting number is off.
  - The DATATRG is executed while the recording operation of the specified setting number is in the preparation or file-saving stage.

Operation error

Error code (SD0)	Description
3405H	The value set to (s) is out of the range, 1 to 4.

24 BUILT-IN ETHERNET FUNCTION INSTRUCTIONS

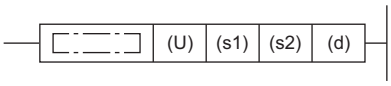
24.1 Open/Close Processing Instructions

Opening a connection

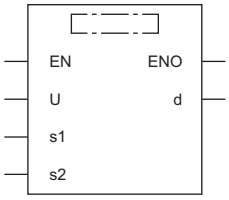
SP.SOCOPEN

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This instruction opens the specified connection.

Ladder	ST
	ENO:=SP_SOCOPEN(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SOCOPEN	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit signed binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(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.	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
+2	Application setting area	Specify the application of a connection. b15 b14 … b10 b9 b8 … b0 (s2)+2 (3) 0 (4) (2) (1) 0 (1) Communication method (protocol) (b8) • 0: TCP/IP • 1: UDP/IP (2) Socket communications function procedure (b9) • 1: No procedure (fixed) (3) Open method (b15, b14) • 00: Active open or UDP/IP • 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.	0001H to 1387H, 1392H to FFFEH (0400H or later recommended)	User
+4 +5	IP address of external device*1	Specify the IP address of an external device.	00000001H to FFFFFFFFH (FFFFFFFFH: simultaneous broadcast)	User
+6	Destination port number	Specify the destination port number.	0001H to FFFFH (FFFFH: simultaneous broadcast)	User
+7 to +9	—	Reserved	—	System

*1 In Unpassive open mode, the IP address of the external device and destination port number are ignored.

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 the SP.SOCOPEN instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

• Completion device (d)

This device turns on during END processing of the scan where the SP.SOCOPEN instruction completes, and turns off during the next END processing.

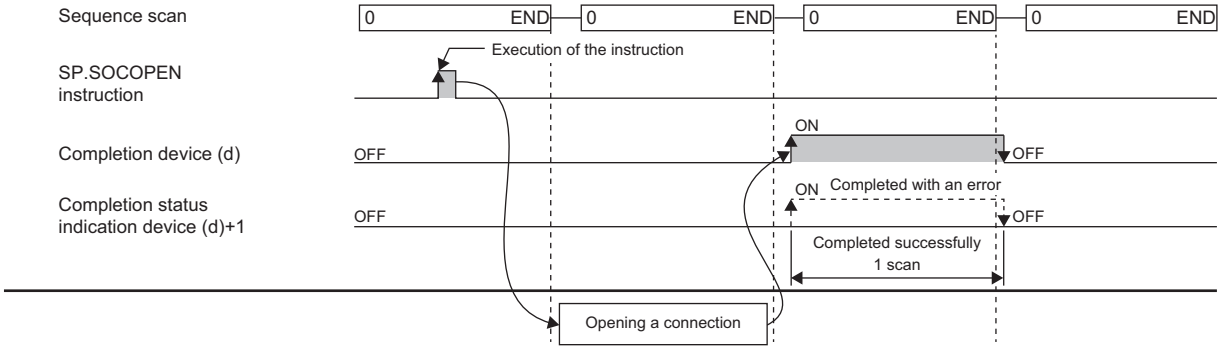
• Completion status indication device (d)+1

This device turns on or off depending on the completion status of the SP.SOCOPEN instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the SP.SOCOPEN instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the SP.SOCOPEN 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.

Operation error

Error code (SD0)	Description
3405H	The connection number specified by (s1) is a value other than 1 to 16.

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.

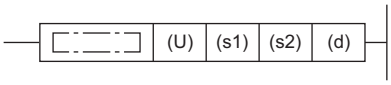
📖MELSEC iQ-R Ethernet User's Manual (Application)

Closing a connection

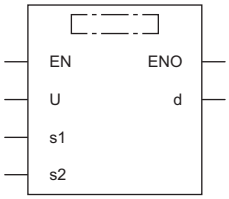
SP.SOCCLOSE

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This instruction closes the specified connection.

Ladder	ST
	ENO:=SP_SOCCLOSE(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SOCCLOSE	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit signed binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—

■Control data

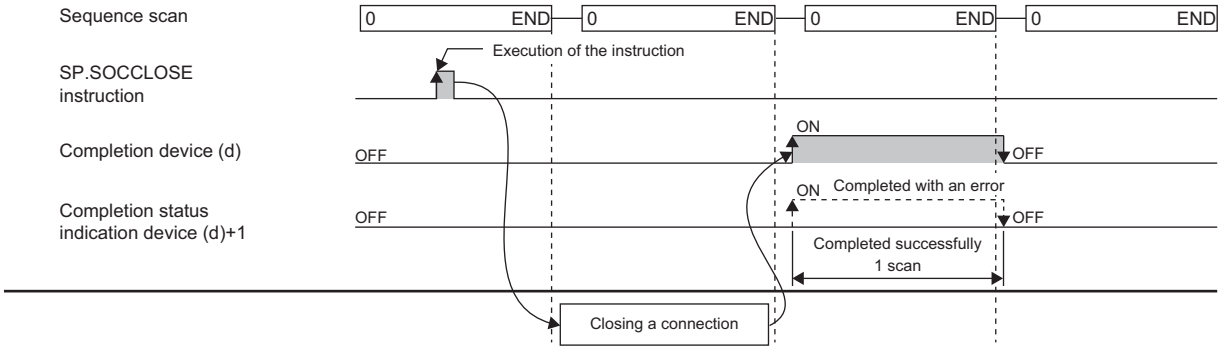
Operand: (s2)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	System area	—	—	—
+1	Completion status	The completion status is stored upon completion of the instruction. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System

Processing details

- This instruction performs close processing for the connection specified by (s1). The setting value used for open processing is selected by (s2)+0.
- The execution status and the completion status of the SP.SOCCLOSE instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

• Completion device (d)
This device turns on during END processing of the scan where the SP.SOCCLOSE 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 the SP.SOCCLOSE instruction.
When completed successfully: The device remains off.
When completed with an error: The device turns on during END processing of the scan where the SP.SOCCLOSE instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the SP.SOCCLOSE instruction.



Operation error

Error code (SD0)	Description
3405H	The connection number specified by (s1) is a value other than 1 to 16.

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.
📖MELSEC iQ-R Ethernet User's Manual (Application)

Point

When a connection waiting for opening the SP.SOCCLOSE instruction is specified in TCP Passive mode, a successful completion occurs when the SP.SOCOPEN or SP.SOCCLOSE instruction is issued and the connection is closed.

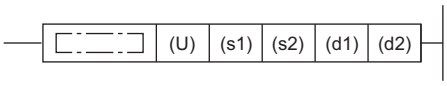
24.2 Socket Communications Instructions

Reading receive data during the END processing

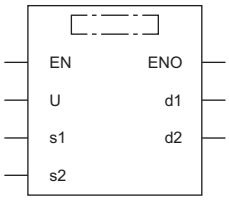
SP.SOCRCV

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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:=SP_SOCRCV(EN,U,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SOCRCV	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit signed binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 2)
(d1)	Start device for storing the receive data	—	Word	ANY16*1
(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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(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

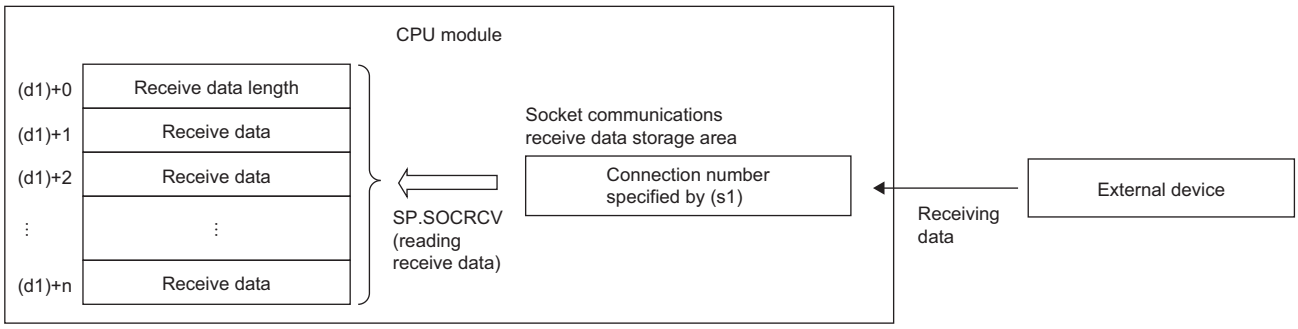
*1 The received data is stored in units of bytes sequentially from lower bytes. When an odd number of bytes of data is received, the last receive data is stored in the lower byte of the last data storage area.

Point

- When the SP.SOCRCV instruction is executed, receive data is read from the socket communications receive data area during END processing. For this reason, executing the SP.SOCRCV 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

- The SP.SOCRCV 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 the SP.SOCRCV instruction can be checked with the completion device (d2) and the completion status indication device (d2)+1.

• Completion device (d2)

This device turns on during END processing of the scan where the SP.SOCRCV instruction completes, and turns off during the next END processing.

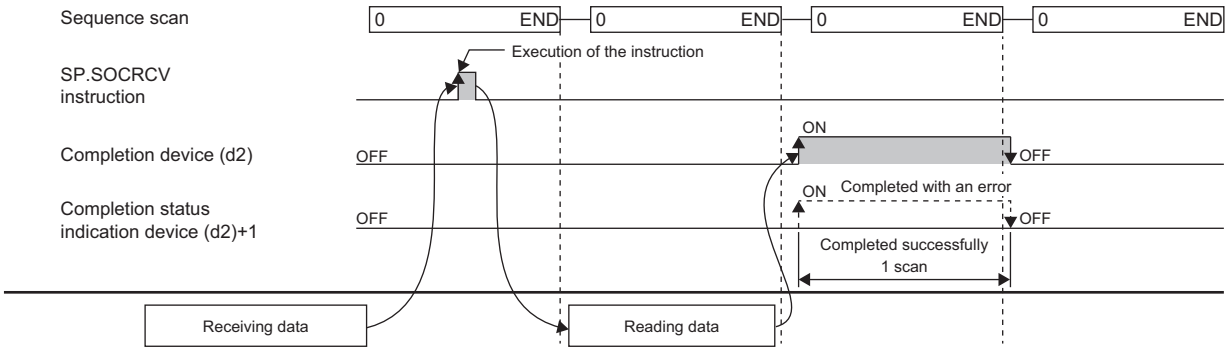
• Completion status indication device (d2)+1

This device turns on or off depending on the completion status of the SP.SOCRCV instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the SP.SOCRCV instruction completes, and turns off during the next END processing.

- The following figure shows the execution timing of the SP.SOCRCV instruction.



Precautions

When reading receive data from the same connection, do not use this command together with the S.SOCRCVS instruction.

Operation error

Error code (SD0)	Description
2820H	The amount of data received exceeds the relevant setting area in the device/label memory in the receive data storage device.
3405H	The connection number specified by (s1) is a value other than 1 to 16.

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.

MELSEC iQ-R Ethernet User's Manual (Application)

Point

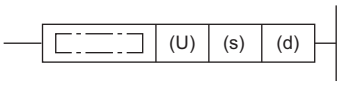
- To avoid receiving too much amount of data, the SP.SOCRMODE instruction can be used to set the size of receive data to limit the amount of receive data.
- By connecting the completion device of the SP.SOCRCV instruction to the execution instruction through a normally closed contact, data can be read continuously even when it is received continuously.

Reading receive data when the instruction is executed

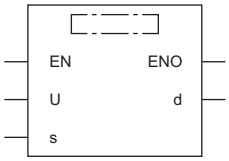
S.SOCRCVS

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This instruction reads the receive data of the specified connection from the socket communications receive data area.

Ladder	ST
	ENO:=S_SOCRCVS(EN,U,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
S.SOCRCVS	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s)	Connection number	1 to 16	16-bit unsigned binary	ANY16
(d)	Start device for storing the receive data	—	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 (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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s)	—	—	○	—	—	—	—	○	○	—	—	—
(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

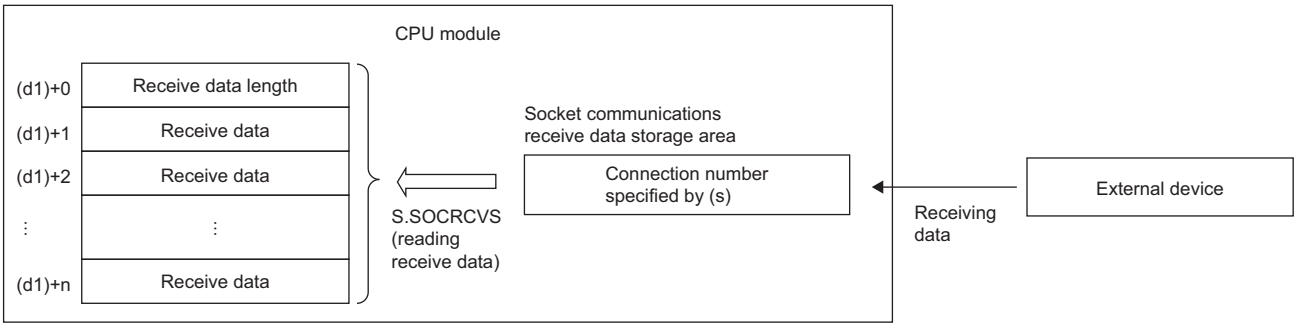
*1 The received data is stored in units of bytes sequentially from lower bytes. When an odd number of bytes of data is received, the last receive data is stored in the lower byte of the last data storage area.

Point

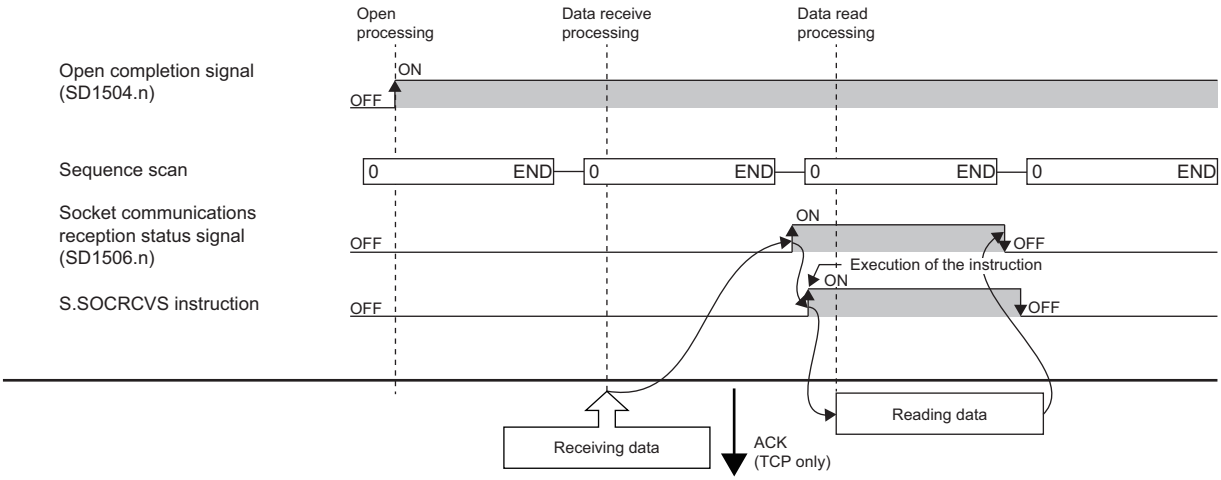
- The default receive data size is 2046 bytes. To receive 2047 bytes of data or more, change the receive data size using the SP.SOCRMODE instruction.
- When an odd number of bytes of data is received, invalid data is stored in the upper byte of the device where the last receive data is stored.

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 the S.SOCRCVS instruction.



Precautions

When reading receive data from the same connection, do not use this command together with the SP.SOCRCV instruction.

Operation error

Error code (SD0)	Description
2820H	The amount of data received exceeds the relevant setting area in the device/label memory in the receive data storage device.
3405H	The connection number specified by (s) is a value other than 1 to 16.

Point

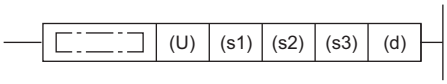
To avoid receiving too much amount of data, the SP.SOCRMODE instruction can be used to set the size of receive data to limit the amount of receive data.

Sending data

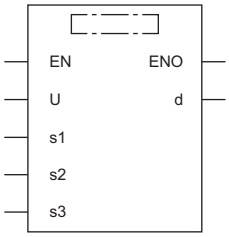
SP.SOCSND

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This instruction sends the data to the external device of the specified connection.

Ladder	ST
	ENO:=SP_SOCSND(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SOCSND	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit signed binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 2)
(s3)	Start device for storing the send data	—	Word	ANY16*1
(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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(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

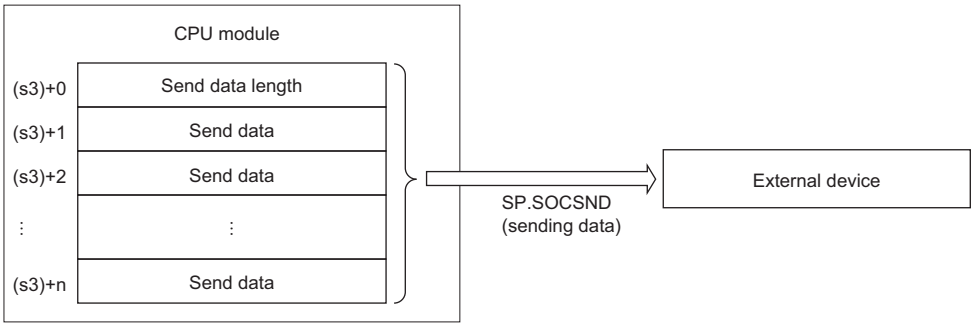
*1 The send data is sent in units of bytes sequentially from lower bytes. When an odd number of bytes of data is received, the last send data is stored in the lower byte of the last data storage area.

Point

When TCP is used, the send data length should be equal to or less than the maximum window size (TCP 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 the SP.SOCSND instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

• Completion device (d)

This device turns on during END processing of the scan where the SP.SOCSND instruction completes, and turns off during the next END processing.

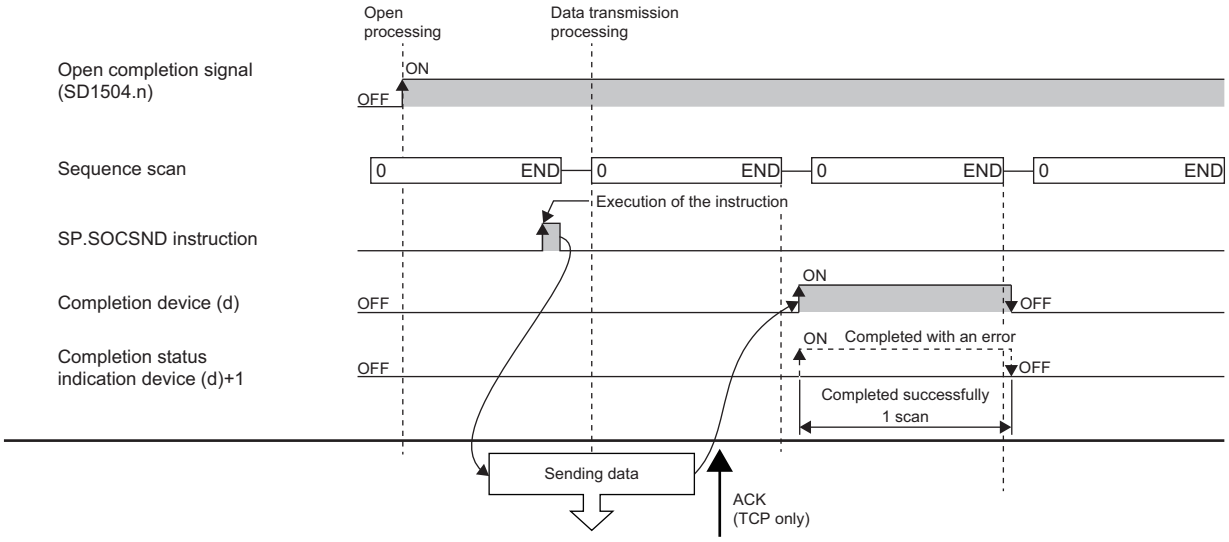
• Completion status indication device (d)+1

This device turns on or off depending on the completion status of the SP.SOCSND instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the SP.SOCSND instruction completes, and turns off during the next END processing.

• The following figure shows the timing of receive processing using the SP.SOCSND instruction.



Even after the completion device turns on, data may be sent continuously. Check the completion of the send processing on the receiving side.

Operation error

Error code (SD0)	Description
3405H	The connection number specified by (s1) is a value other than 1 to 16.

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.

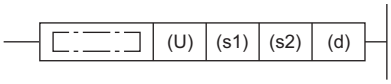
📖MELSEC iQ-R Ethernet User's Manual (Application)

Reading connection information

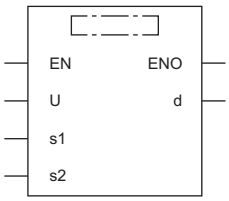
SP.SOCCINF

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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:=SP_SOCCINF(EN,U,s1,s2,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SOCCINF	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit unsigned binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(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

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. b15 b14 … b10 b9 b8 … b0 (d)+4 (3) 0 (4) (2) (1) 0 (1) Communication method (protocol) (b8) • 0: TCP/IP • 1: UDP/IP (2) Socket communications function procedure (b9) • 1: No procedure (fixed) (3) Open method (b15, b14) • 00: Active open or UDP/IP • 10: Unpassive open • 11: Fullpassive open (4) Predefined protocol setting (b10)*1 • 0: Do not use the predefined protocol support function. (Use the socket communications function.) • 1: Use the predefined protocol support function.	—	System

*1 The setting can be used only for the following models. Note that the applicable firmware version varies depending on models.
· R00CPU, R01CPU, R02CPU: There are no restrictions on the version.
· Rn(EN)CPU other than the above: "29" or later
· RnPCPU: "15" or later

Processing details

Reads the connection information of the connection specified by (s1).

Operation error

Error code (SD0)	Description
3405H	The connection number specified by (s1) is a value other than 1 to 16.

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.

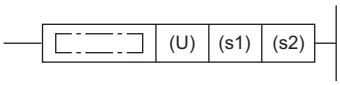
📖MELSEC iQ-R Ethernet User's Manual (Application)

Changing the communication target (UDP/IP)

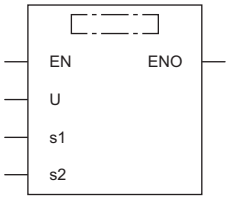
SP.SOCCSET

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This instruction changes the communication target IP address and port number of the specified connection.

Ladder	ST
	ENO:=SP_SOCCSET(EN,U,s1,s2);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SOCCSET	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit unsigned binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(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/IP communications, this instruction changes the communication target IP address and port number of the connection specified by (s1).

Point

- Using the SP.SOCCSET instruction allows the user to change the communication destination without closing the connection.
- If the SP.SOCCSET instruction is executed while there is data in the receive data area, the instruction is validated after the SP.SOCRCV or S.SOCRCVS dedicated instruction is executed. If the SP.SOCCSET instruction is executed while there is no data in the receive data area, the instruction is validated soon after it is executed.

Precautions

Do not use the SP.SOCCSET instruction to change the communication destination during execution of the SP.SOCSND instruction.

Operation error

Error code (SD0)	Description
3405H	The connection number specified by (s1) is a value other than 1 to 16.

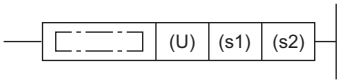
When completed with an error, an error code is stored in the completion status (s2)+1.
For the error code stored in the completion status (s2)+1, refer to the following.
📖MELSEC iQ-R Ethernet User's Manual (Application)

Changing the receive mode

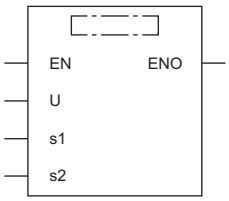
SP.SOCRMODE

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This instruction changes the TCP receive mode and receive data size for the specified connection (invalid for UDP communications connections).

Ladder	ST
	ENO:=SP_SOCRMODE(EN,U,s1,s2);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SOCRMODE	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit signed binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(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 receive mode*1	Store the TCP receive mode. • 0: TCP standard receive mode • 1: TCP 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 communications connections.

Processing details

- This instruction changes the TCP receive mode and receive data size for the connection (other than a UDP communications connection) specified by (s1).
- For TCP connections, the function enables the mode specified by (s2)+2.

■TCP standard receive mode

Upon receipt of data, the instruction stores the data in the socket communications receive data area and turns on SD1506 (socket communications receive status signal).

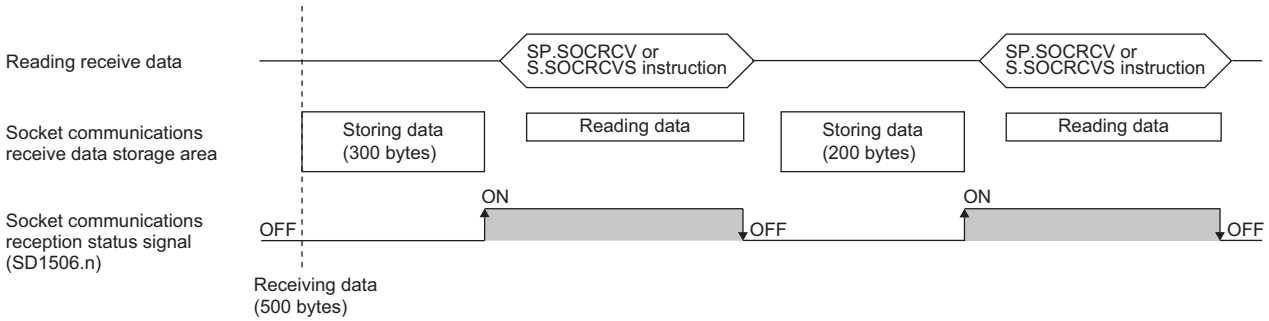
If the received data exceeds the specified receive data size, the excess data becomes the next receive data.

If data is received later before data is read from the socket communications receive data area using the SP.SOCRCV or S.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 SP.SOCRCV or S.SOCRCVS instruction, the instruction stores the data in the socket communications receive data area and turns on SD1506 (socket communications receive status signal).

Ex.

When 500 bytes of data is received while the receive data size is set to 300 bytes



■TCP 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 (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 (socket communications receive status signal) turns on.

If the received data exceeds the specified receive data size, the excess data becomes the next receive data.

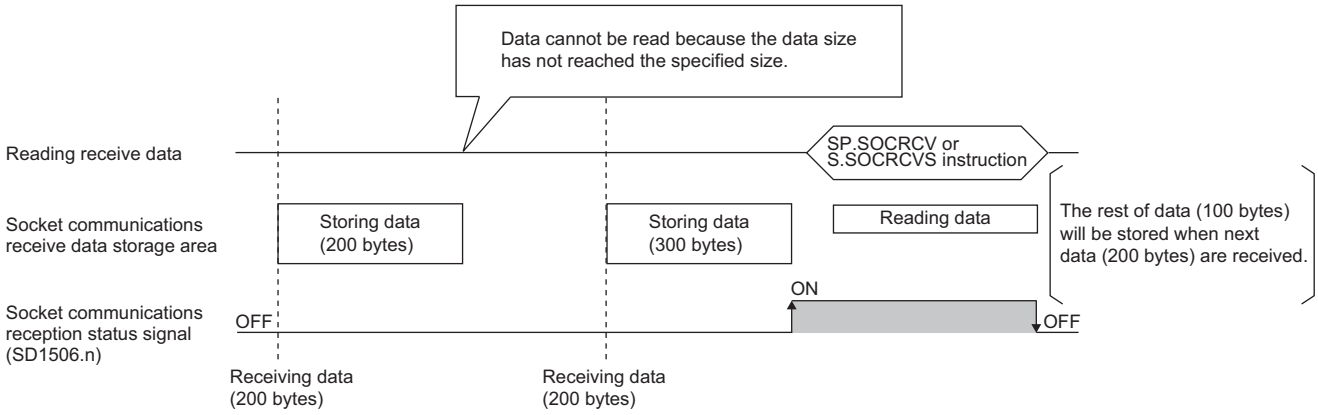
If data is received later before data is read from the socket communications receive data area using the SP.SOCRCV or S.SOCRCVS instruction, it is stored in the receive data area in the OS.

If the receive data area in the OS contains data when data is read from the socket communications receive data area using the SP.SOCRCV or S.SOCRCVS instruction, the instruction stores the data in the socket communications receive data area, but does not turn on SD1506 (socket communications receive status signal) if the data has not reached the specified receive data size.

Thereafter, data reception is repeated until the received data reaches the receive data size. When it reaches the receive data size, SD1506 (socket communications receive status signal) turns 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 SP.SOCRCV or S.SOCRCVS instruction needs a 1024-word area by default. Specifying the receive data size in 1024 words or less 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 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 fixed-length receive mode enables data to be correctly divided and received.

- The size of the receive data to be read once by the SP.SOCRCV or S.SOCRCVS instruction is specified in (s2)+3. In the case of UDP, if the received data exceeds the specified receive data size, the excess data becomes the next receive data.

Operation error

Error code (SD0)	Description
3405H	The connection number specified by (s1) is a value other than 1 to 16.

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.

MELSEC iQ-R Ethernet User's Manual (Application)

Point

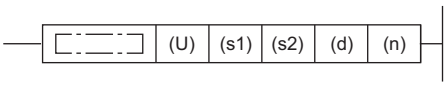
When the receive status signal does not turn on in TCP fixed-length receive mode, the data received as of the current time can be read with the SP.SOCRDATA instruction to check whether the data sent from the external device is missing.

Reading socket communications receive data

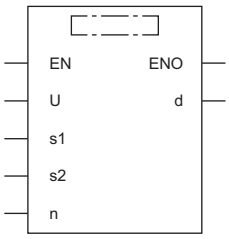
S(P).SOCRDATA

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

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:=S_SOCRDATA(EN,U,s1,s2,n,d); ENO:=SP_SOCRDATA(EN,U,s1,s2,n,d);

FBD/LD



■Execution condition

Instruction	Execution condition
S.SOCRDATA	
SP.SOCRDATA	

Setting data

■Descriptions, ranges, and data types

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit signed binary	ANY16
(s2)	Start device where control data is stored	Refer to the control data.	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	○	—	—	—
(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

- 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 SP.SOCRCV or S.SOCRCVS instruction is executed can be changed.
- After issuing the S(P).SOCRDATA instruction to check the data to be received this time and issuing the SP.SOCRMODE instruction to specify the size of the data to be received next time, the SP.SOCRCV or S.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
3405H	The connection number specified by (s1) is a value other than 1 to 16.
	The value of the device specified by (n) exceeds 5120.

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.
📖MELSEC iQ-R Ethernet User's Manual (Application)

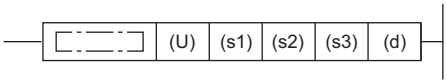
24.3 Predefined Protocol Support Function Instruction

Executing the registered protocols

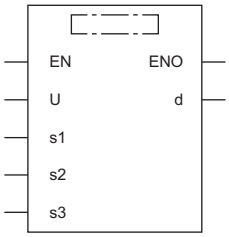
SP.ECPRTCL

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This instruction executes the protocol that has been set by the predefined protocol support function.

Ladder	ST
	ENO:=SP_ECPRTCL(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.ECPRTCL	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Connection number	1 to 16	16-bit unsigned binary	ANY16
(s2)	Number of protocols to be executed continuously	1 to 8	16-bit unsigned binary	ANY16
(s3)	Start device where control data is stored	Refer to the control data.	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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	○	—	○*1	—	—	—	—	○*1	○	—	—	—
(s2)	○	—	○*1	—	—	—	—	○*1	○	—	—	—
(s3)	○	—	○*1	—	—	—	—	○*1	—	—	—	—
(d)	○*1	—	○	—	—	—	—	○*1	—	—	—	—

*1 A local device and a file register which is set for each program cannot be used.

■Control data

Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Resulting number of executed protocols	The number of protocols executed by the SP.ECPRTCL 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 "receive 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 "receive 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 "receive 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 "receive 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

Operand: (s3)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+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 "receive 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 "receive 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 "receive 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
+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 "receive 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). (📖MELSEC iQ-R Ethernet User's Manual (Application))
- The execution status and the completion status of the SP.ECPRTCL instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

• Completion device (d)

This device turns on during END processing of the scan where the SP.ECPRTCL instruction completes, and turns off during the next END processing.

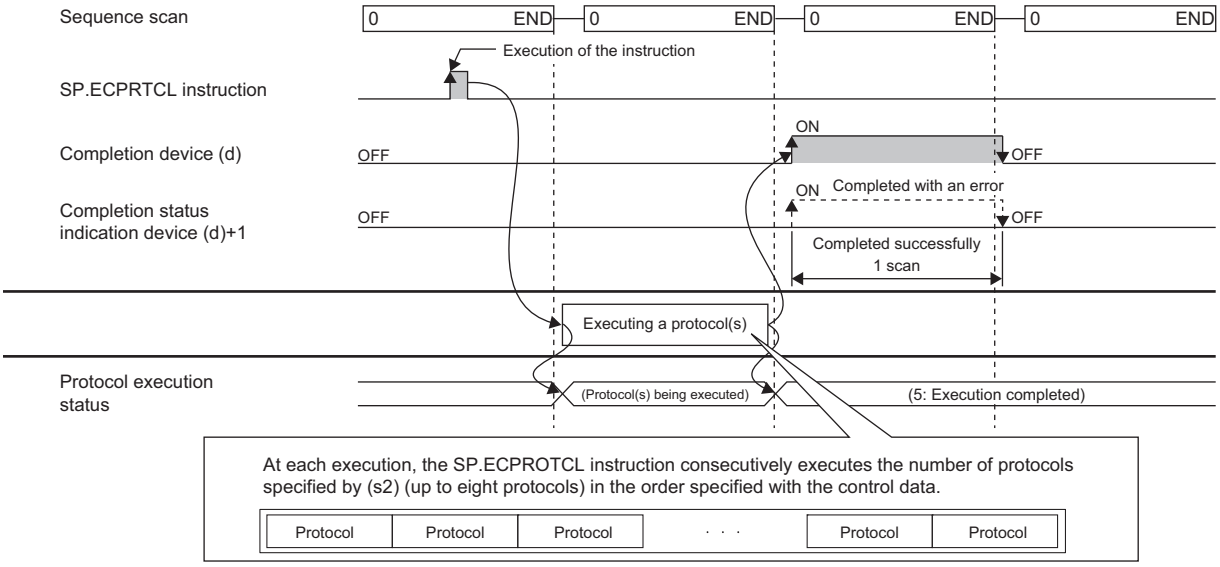
• Completion status indication device (d)+1

This device turns on or off depending on the completion status of the SP.ECPRTCL instruction.

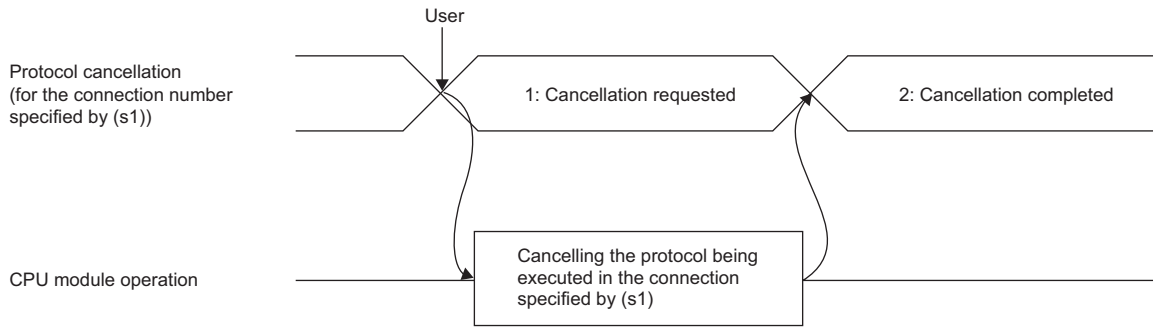
When completed successfully: The device remains off.

When completed with an error: The device turns on during END processing of the scan where the SP.ECPRTCL 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 SP.ECPRTCL instruction execution timing.



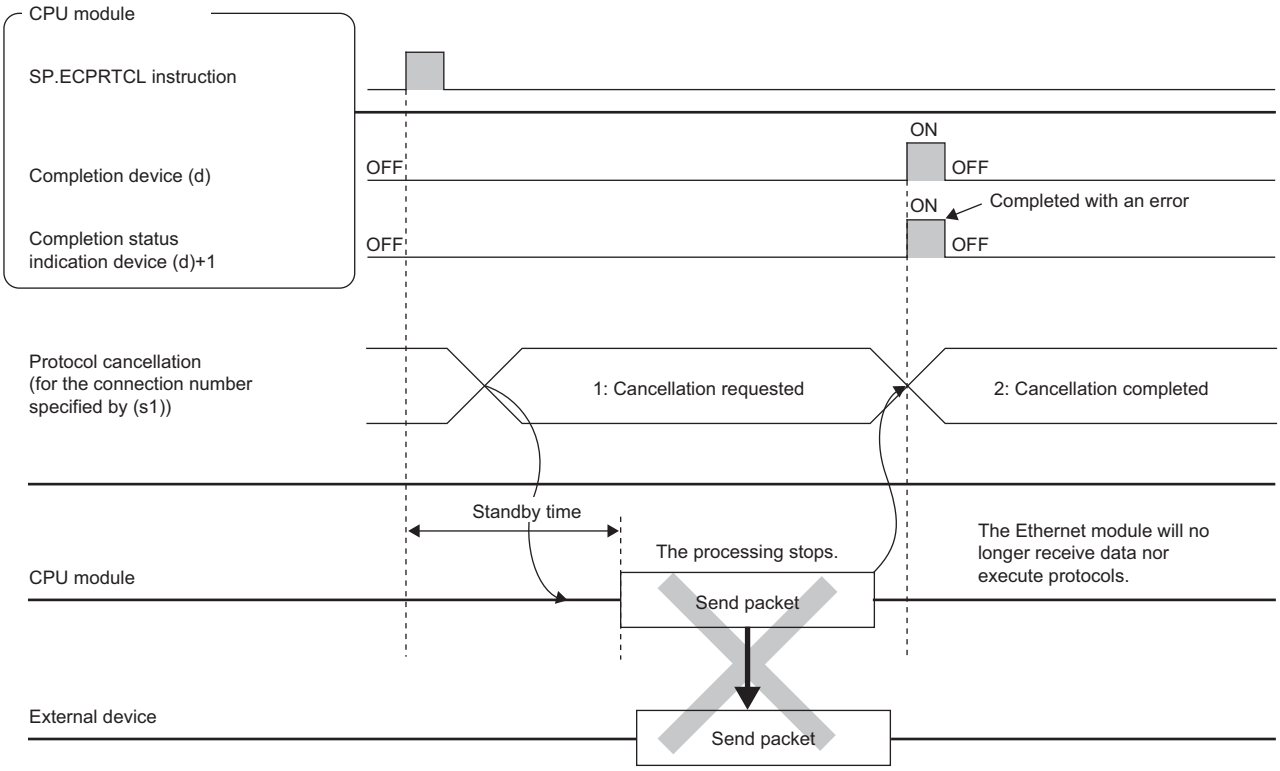
- Protocol execution can be canceled by setting a protocol cancel request. The protocol cancel request is specified in the predefined protocol support function execution status check area (Un\G350 to Un\G669). (📖MELSEC iQ-R Ethernet User's Manual (Application))



- The following figure shows the protocol cancel operations from time to time.

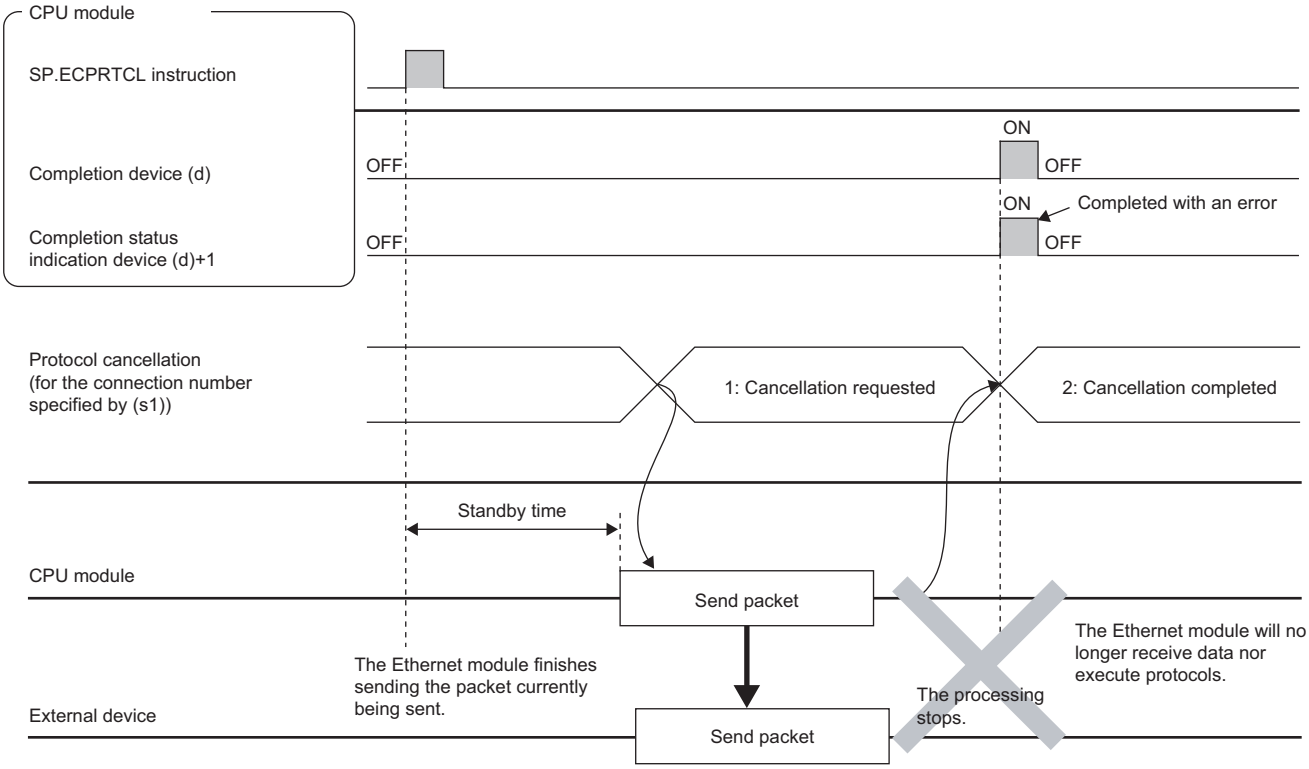
• If a cancel request is issued before transmission

The following figure shows the operation when the protocol execution status is "1: Waiting for transmission".



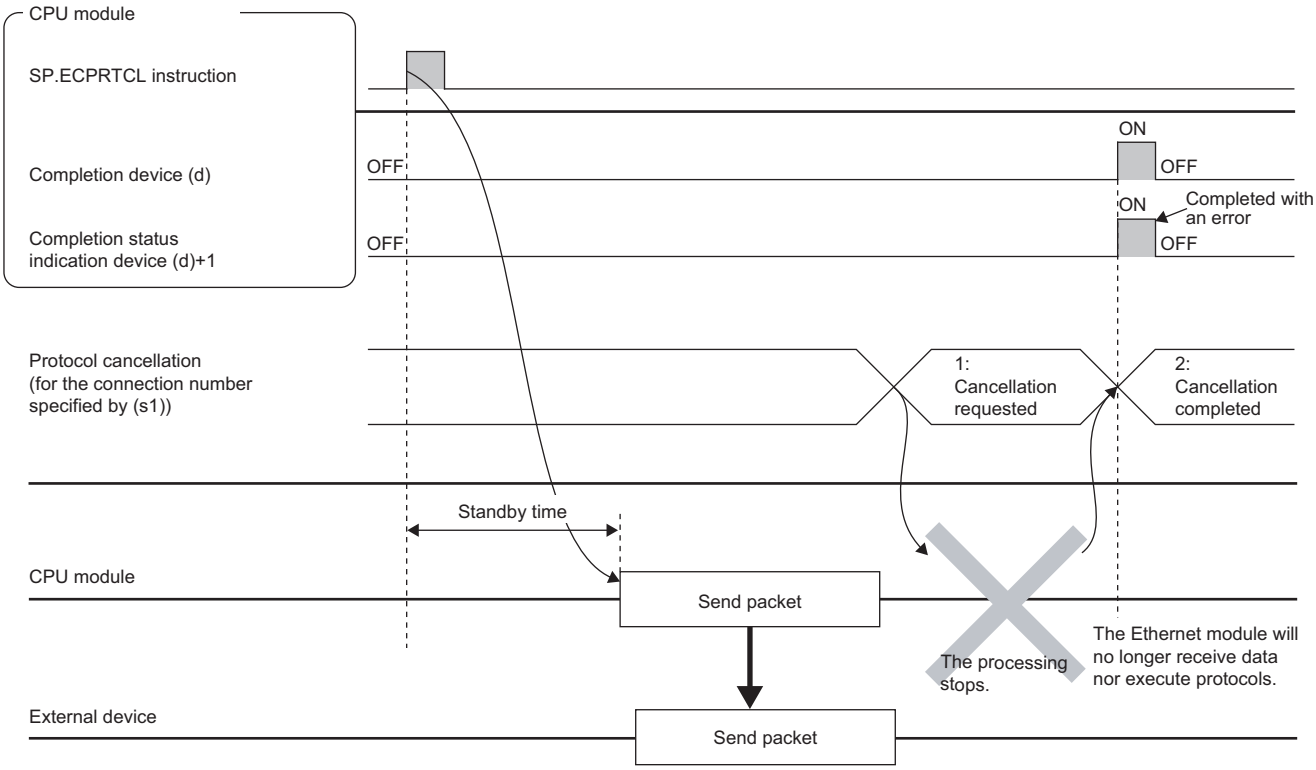
• If a cancel request is issued before completion of transmission

The following figure shows the operation when transmission has not been completed while the protocol execution status is "2: Sending".



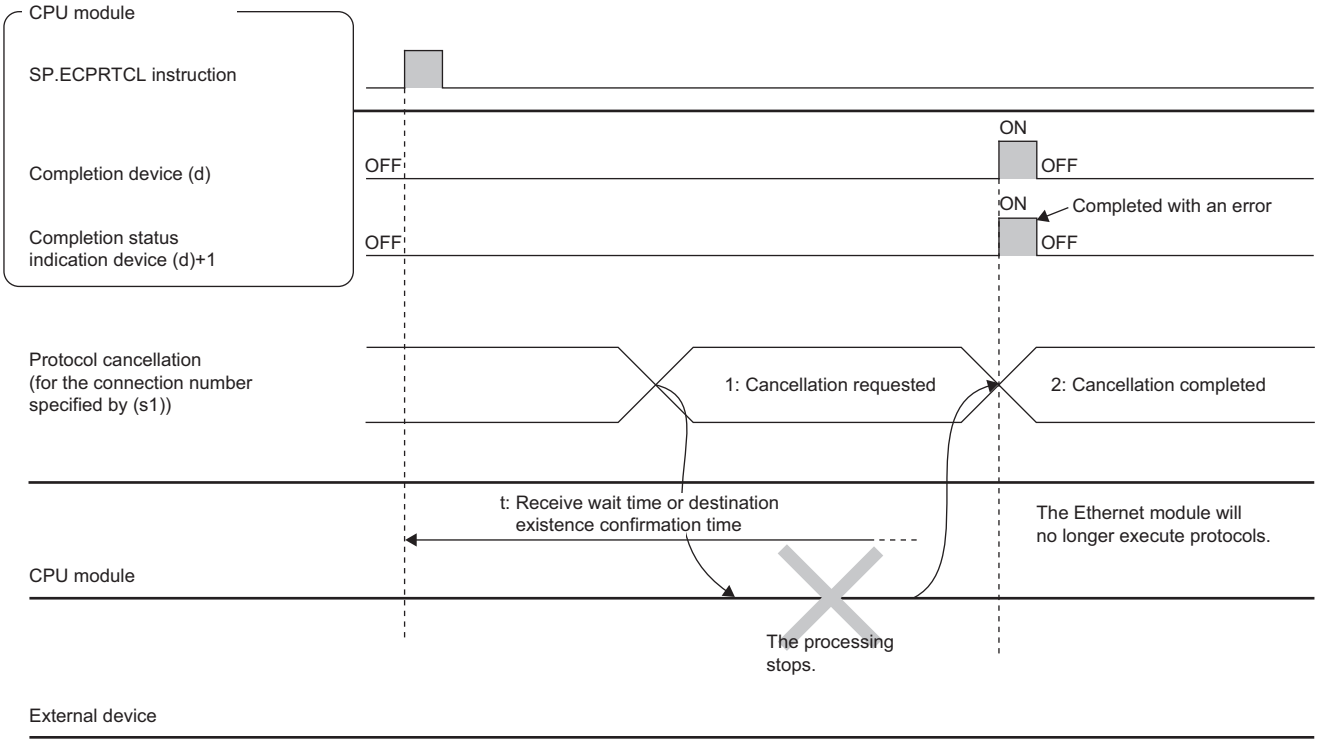
• If a cancel request is issued upon completion of transmission

The following figure shows the operation when transmission has been completed while the protocol execution status is "2: Sending".



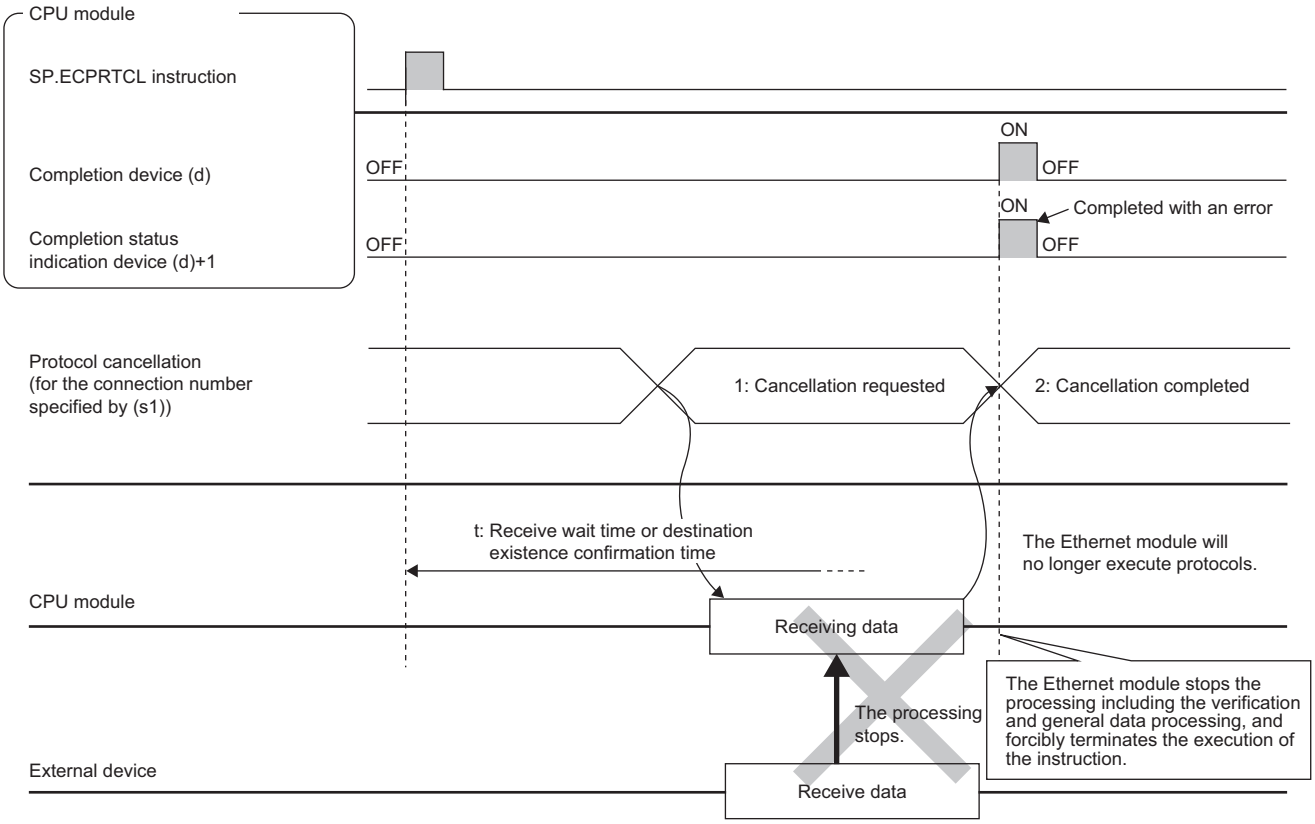
• If a cancel request is issued while waiting for reception

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
3405H	(s1) is not a value in the range from 1 to 16.

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.

📖MELSEC iQ-R Ethernet User's Manual (Application)

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
  - The SP.ECPRTCL 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 CPU module 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 CPU module 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 CPU module periodically checks for a cancel request. For this reason, it may take time until cancel processing is performed after a cancel request is issued.
- The SP.ECPRTCL instruction itself does not open/close a connection and therefore the SP.SOCOPEN/SP.SOCCLOSE instructions need to be used to open/close the connection.

☞ Page 1174 SP.SOCOPEN,Page 1177 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", the SP.ECPRTCL instruction is not completed until the data specified in the protocol setting is received.

24.4 SLMP Frame Send Instruction

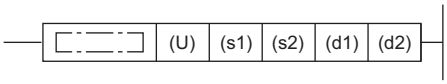
Sending an SLMP frame

SP.SLMPSND

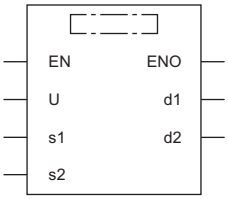
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "17" or later support this instruction. Use an engineering tool with version "1.020W" or later.
- The RnPCPU with firmware version "13" or later supports this instruction. Use an engineering tool with version "1.040S" or later.

This instruction sends SLMP messages to the SLMP-compatible device.

Ladder	ST
	ENO:=SP_SLMPSND(EN, U, s1, s2, d1, d2)

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SLMPSND	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Start device where control data is stored	☞ Page 1207 Control data	Word	ANY16*1
(s2)	Start device where a request frame is stored	☞ Page 1209 Request frame	Word	ANY16*1
(d1)	Start device for storing a response frame	☞ Page 1209 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 (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□\□, U3E□\(H)G□	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 or not. • 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	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~FFFFFFEH (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 • 03D0H: Control system CPU • 03D1H: Standby system CPU • 03D2H: System A CPU • 03D3H: System B CPU	03FFH, 03E0H to 03E3H, 03D0H to 03D3H	User

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+9	Request destination multidrop station number	0000H (fixed)	0000H	User
+10	Number of resends	The device becomes effective when the execution type specified by (s1)+0 is "1: With arrival check". ■Before instruction execution Specify the number of resends to be performed if the instruction is not completed within the monitoring time specified by (s1)+11. • 0 to 15 (times) ■At completion of instruction The number of resends performed (result) is stored. • 0 to 15 (times)	0 to 15	User/ system
+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. b15 b8 b7 b0 3 | 4 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. b15 b8 b7 b0 1 | 2 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 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 Process CPU (redundant mode), the serial numbers are held separately in the system A and system B.
For the serial number, refer to the following.
📖SLMP Reference Manual

■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	Execution results for the request data are set. (Some commands do not return response data.)	—	System

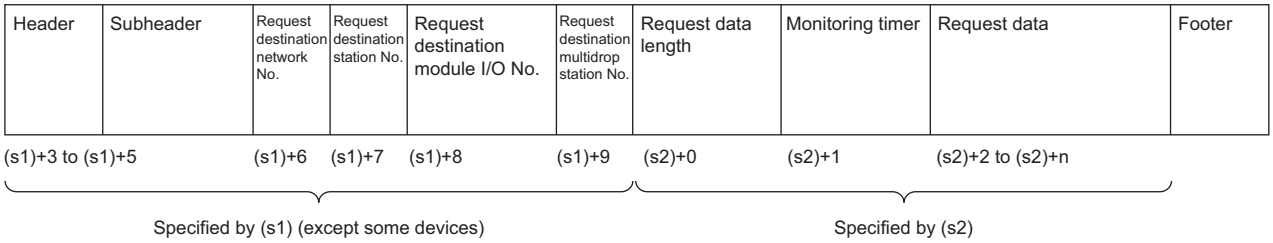
*1 The response data is stored in units of bytes sequentially from lower bytes. When an odd number of bytes of response data is received, the last response data is stored in the lower byte of the last data storage area.

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).

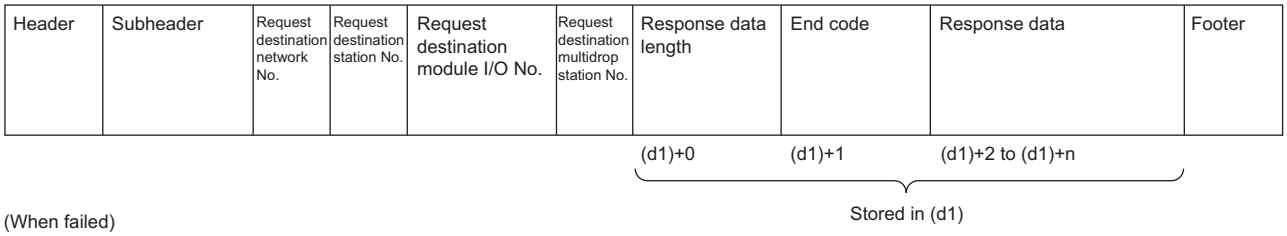
The following figures show the request data and the response data in normal/abnormal end.

[Request data]

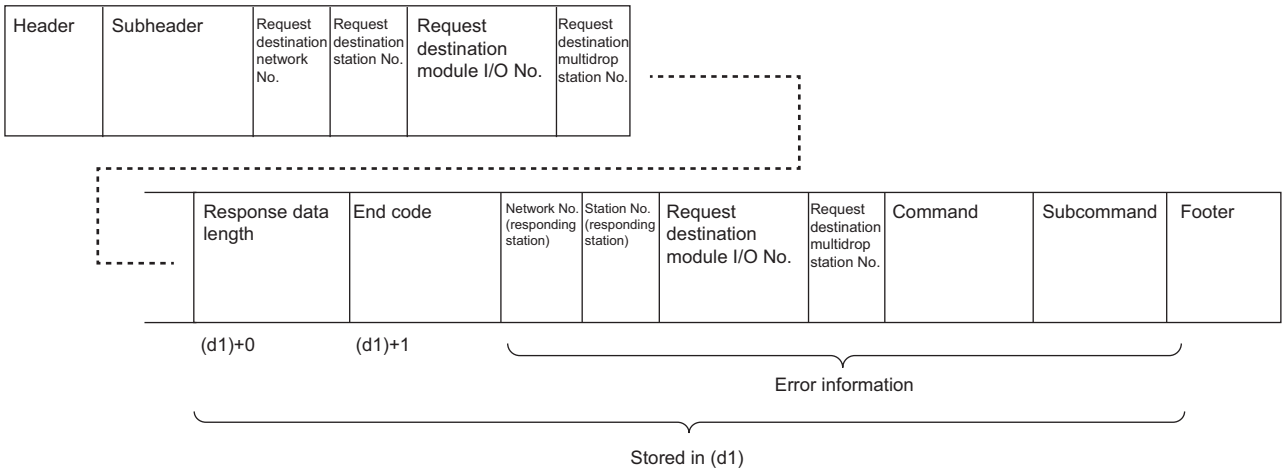


[Response data]

(When completed)



(When failed)



- The SP.SLMPSND instruction communicates using UDP. Set the external device to use UDP.
- The SP.SLMPSND 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 the SP.SLMPSND instruction can be checked with the completion device (d2)+0 and the completion status indication device (d2)+1.

• Completion device (d2)+0

This device turns on during END processing of the scan where the SP.SLMPSND 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 the SP.SLMPSND instruction.

When completed successfully: The device remains off.

When completed with an error: The device turns on during the END processing of the scan where the SP.SLMPSND instruction completes, and turns off during the next END processing.

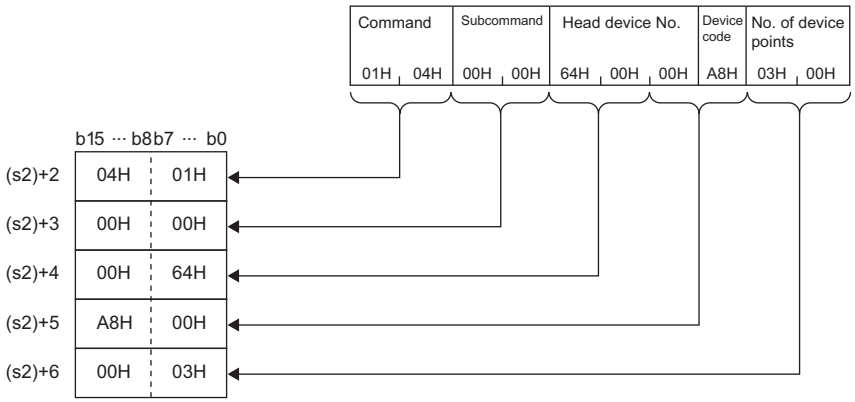
- When an odd number of bytes of response data is received, invalid data is stored in the upper byte of the device where the last response data is stored.
- For the Process CPU (redundant mode), the system A and system B communicate using separate IP addresses.

- The following figures show the request data for sending "Read (command: 0401H)" (reading in units of words) and the response data in normal end.

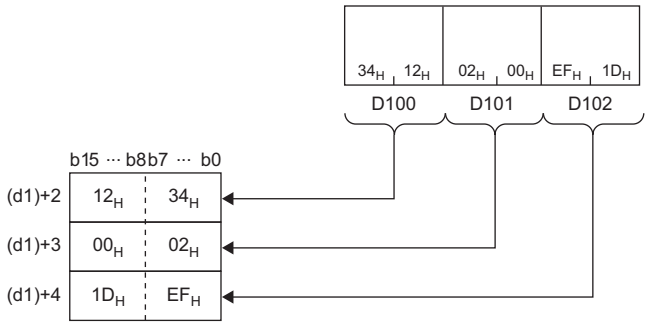
Ex.

Reading the value in D100 to D102

[Request data]

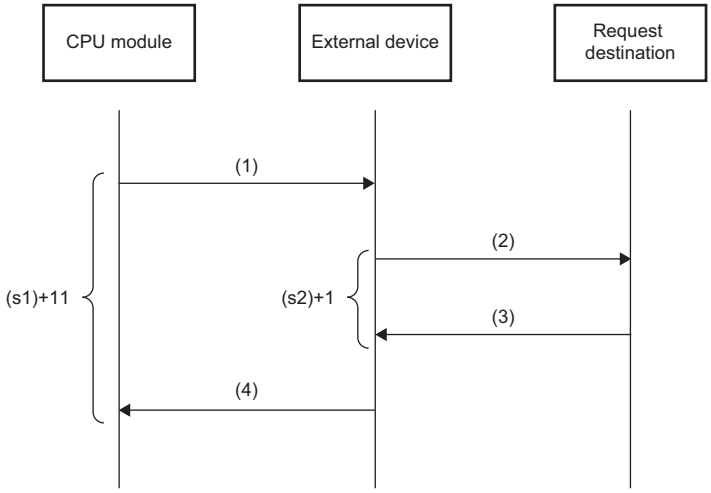


[Response data]



Precautions

- When executing multiple SP.SLMPSND instructions concurrently, be careful not to overlap the channels of the SP.SLMPSND instructions. Multiple SP.SLMPSND instructions specifying the same channels cannot be used concurrently. When the execution conditions of the SP.SLMPSND instructions in the same channel are satisfied in the same sequence scan, only the SP.SLMPSND instruction that has been executed first is enabled and the subsequent SP.SLMPSND instructions are not executed. In addition, any subsequent SP.SLMPSND instruction of the same channel setting as the SP.SLMPSND instruction being executed is not executed. If the CPU module does not execute the processing of the SP.SLMPSND instruction, SM699 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

The SP.SLMPSND instruction is successfully completed even if the target device returns an abnormal response. When the SP.SLMPSND 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
3405H	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.

MELSEC iQ-R Ethernet User's Manual (Application)

24.5 File Transfer Function Instructions

Sending FTP client files

SP.FTPPUT

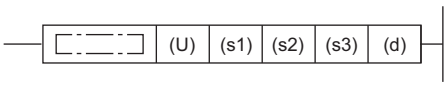
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "22" or later support this instruction. Use an engineering tool with version "1.025B" or later.
- The RnPCPU with firmware version "13" or later supports this instruction. Use an engineering tool with version "1.040S" or later.

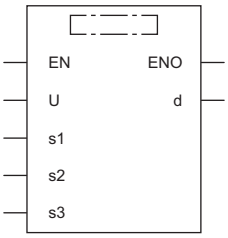
This instruction sends files in the CPU module, which are specified by (s2), to the folder path of the FTP server, which is specified by (s3).

Note that the FTP server can be specified by (s1) as well. *1

*1 When specifying the FTP server by the control data of file transfer function instructions, check the version of the CPU module used. (MELSEC iQ-R Programmable Controller CPU Module User's Manual)

Ladder	ST
	ENO:=SP_FTPPUT(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.FTPPUT	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Start device where control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 4 or 172)*3
(s2)	Name of files stored in the CPU module (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 "_".

*3 The necessary number of array elements depends on the FTP server specification setting of control data.
·When b15 of (s1)+0 is 0: Number of elements is 4
·When b15 of (s1)+0 is 1: Number of elements is 172

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(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 (3) 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 (FTPCLI_I.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 (3) FTP server specification setting (b15) Set whether to specify the FTP server by control data. When Do not specify is set, the FTP server specified with the module parameter is used. When Specify is set, the FTP server specified with (s1)+4 to 171 is used. • 0: Do not specify (the setting of (s1)+4 to 171 is not required) • 1: Specify (the setting of (s1)+4 to 171 is required)	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 the SP.FTPPUT instruction is stored.	—	System
+3	Number of transferred files	The number of transferred files is stored.	—	System

The following table shows the control data that needs to be specified when the FTP server specification setting ((s1)+0, Application setting area, b15) is 1: Specify.

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+4	FTP server specification	Set an FTP server specification method. • 0000H: FTP server address • 0001H: FTP server name	0000H, 0001H	User
+5 to +133	FTP server name	Set an FTP server name (domain name) with an ASCII character string of the following range: • 1 to 256 characters (one-byte alphanumeric characters, one-byte symbols) The first character is stored in the lower byte of (s1)+5, and the second character in the upper byte of (s1)+5. 00H is stored immediately after the last character. (The characters until 00H are recognized as a character string.) Note that this setting is not required when specifying "0000H: FTP server address" for FTP server specification of (s1)+4. b15 … b8 b7 … b0 (s1)+5: ASCII code (2nd character) / ASCII code (1st character) (s1)+6: ASCII code (4th character) / ASCII code (3rd character) (s1)+7: ASCII code (6th character) / ASCII code (5th character) (s1)+8: ASCII code (8th character) / ASCII code (7th character) ⋮ (s1)+132: ASCII code (256th character) / ASCII code (255th character) (s1)+133: 00H / 00H Null character (end of string)	1 to 256 characters (one-byte alphanumeric characters, one-byte symbols)	User

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+134 +135	FTP server address	Set an IP address of the FTP server within the following range. • 000.000.000.001 to 223.255.255.254 The first octet is stored in the upper byte of (s1)+135, the second octet in the lower byte of (s1)+135, the third octet in the upper byte of (s1)+134, and the fourth octet in the lower byte of (s1)+134. Note that this setting is not required when specifying "0001H: FTP server name" for FTP server specification of (s1)+4. b15 b8 b7 b0 (s1)+134: (Third octet) (Fourth octet) (s1)+135: (First octet) (Second octet)	000.000.000.001 to 223.255.255.254	User
+136 to +152	Login name	Set a login name for accessing the FTP server with an ASCII character string of the following range: • 1 to 32 characters (one-byte alphanumeric characters, one-byte symbols) The first character is stored in the lower byte of (s1)+136, and the second character in the upper byte of (s1)+136. 00H is stored immediately after the last character. (The characters until 00H are recognized as a character string.) b15 b8 b7 b0 (s1)+136: ASCII code (2nd character) ASCII code (1st character) (s1)+137: ASCII code (4th character) ASCII code (3rd character) (s1)+138: ASCII code (6th character) ASCII code (5th character) (s1)+139: ASCII code (8th character) ASCII code (7th character) ⋮ (s1)+151: ASCII code (32nd character) ASCII code (31st character) (s1)+152: 00H 00H Null character (end of string)	1 to 32 characters (one-byte alphanumeric characters, one-byte symbols)	User
+153 to +169	Password	Set a password for logging in to the FTP server with an ASCII character string of the following range: 0 to 32 characters (one-byte alphanumeric characters, one-byte symbols) The first character is stored in the lower byte of (s1)+153, and the second character in the upper byte of (s1)+153. 00H is stored immediately after the last character. (The characters until 00H are recognized as a character string.) b15 b8 b7 b0 (s1)+153: ASCII code (2nd character) ASCII code (1st character) (s1)+154: ASCII code (4th character) ASCII code (3rd character) (s1)+155: ASCII code (6th character) ASCII code (5th character) (s1)+156: ASCII code (8th character) ASCII code (7th character) ⋮ (s1)+168: ASCII code (32nd character) ASCII code (31st character) (s1)+169: 00H 00H Null character (end of string)	0 to 32 characters (one-byte alphanumeric characters, one-byte symbols)	User
+170	Connection method	Set a connection method for the FTP server. • 0000H: PORT mode • 0001H: PASV mode	0000H, 0001H	User
+171	Port number	Set an FTP server control port number within the following range. • 1 to 65535 (0001H to FFFFH)	1 to 65535 (0001H to FFFFH)	User

Processing details

- When the FTP server is specified with the module parameter (b15 of (s1)+0 is "0: Do not specify"), this instruction sends files in the CPU module specified by (s2) to the folder path of the FTP server specified by (s3). The CPU module opens a connection with the FTP server set in the module parameters ("FTP Client Settings") at execution of the instruction, and closes a connection after sending files. For details on the parameter setting, refer to the following.

📖MELSEC iQ-R Ethernet User's Manual (Application)

- When specifying the FTP server with the control data of the SP.FTPPUT instruction (b15 of (s1)+0 is "1: Specify"), specify the server by (s1)+4 to 171.

· Specify the FTP server name, login name, and password in (s1)+4 to 171 using an ASCII character string, and store 00H immediately after the last character. The setting examples are shown below.

Ex.

When setting "SERV001" for FTP server name ((s1)+5~). The start device of the control data (s1) is D0.

	b15 b8	b7 b0
(s1)+5	45H (E)	53H (S)
(s1)+6	56H (V)	52H (R)
(s1)+7	30H (0)	30H (0)
(s1)+8	00H	31H (1)

[Input in the program]

When setting "USER001" for Login name ((s1)+136~). The start device of the control data (s1) is D0.

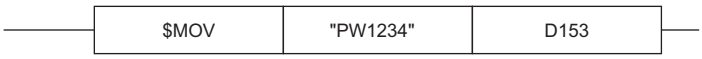
	b15 b8	b7 b0
(s1)+136	53H (S)	55H (U)
(s1)+137	52H (R)	45H (E)
(s1)+138	30H (0)	30H (0)
(s1)+139	00H	31H (1)

[Input in the program]

When setting "PW1234" for Password ((s1)+153~). The start device of the control data (s1) is D0.

	b15 b8	b7 b0
(s1)+153	57H (W)	50H (P)
(s1)+154	32H (2)	31H (1)
(s1)+155	34H (4)	33H (3)
(s1)+156	00H	00H

[Input in the program]

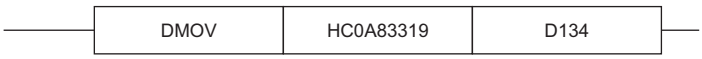


· Set the FTP server address in (s1)+134 to 135. Each part of the IP address, which is separated by a dot, should be represented as a single byte in hexadecimal format. The setting example is shown below.

Ex.

	b15 b8	b7 b0
(s1)+134	33H (51)	19H (25)
(s1)+135	C0H (192)	A8H (168)

[Input in the program]



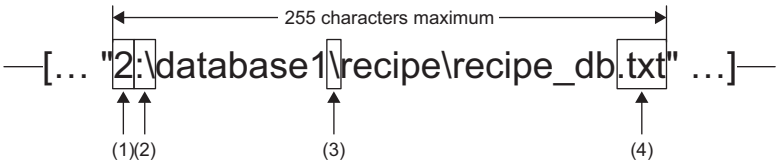
· The following FTP server has been verified to operate correctly when specified with the control data of the SP.FTPPUT instruction.

FTP server	Applicable operating system
Microsoft® Internet Information Services (IIS)	Microsoft® Windows® 11 Microsoft® Windows® 10

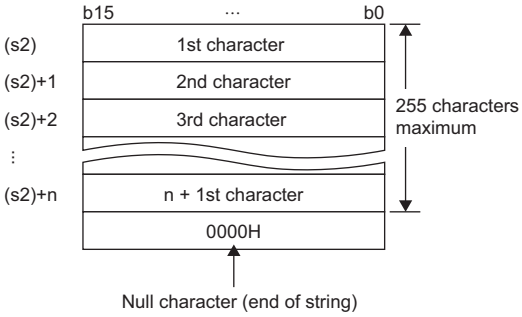
· When specifying the FTP server name, the DNS setting of the module parameter is required. (📖MELSEC iQ-R Ethernet User's Manual (Application))

- The total number of files to be transferred by the SP.FTPPUT instruction is stored in (s1)+2, and the number of transferred files is stored in (s1)+3.

- Specify the transfer source drive number (2 to 4) of the CPU module, the folder path where the files are stored, and the file name (including an extension) in (s2) in Unicode string. 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)Drive numbers that can be specified are 2 to 4. Specify 3 or 4 for the R00CPU.
(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.



- Wild card characters (*, ?) can be used in the file name or the extension specified in (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.)

Wild card characters do not recognize periods.

Using wild card characters in the following ways results in an error.

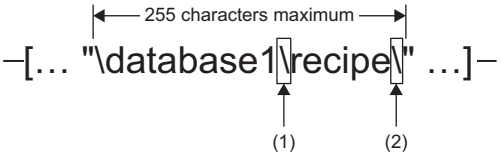
- Two or more asterisks '*' are used in a file name (before the period) or an extension. (Example: *abc*.txt)
- An asterisk '*' and a question mark '?' are used in a file name (before the period) or an extension. (Example: *ab?. txt)

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 the SP.FTPPUT instruction, the transfer processing will be stopped upon detection of the error, and the rest of 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.
- Specify the folder path of the transfer destination FTP server in (s3) in Unicode string. The specified folder path shall be a relative 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. 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.



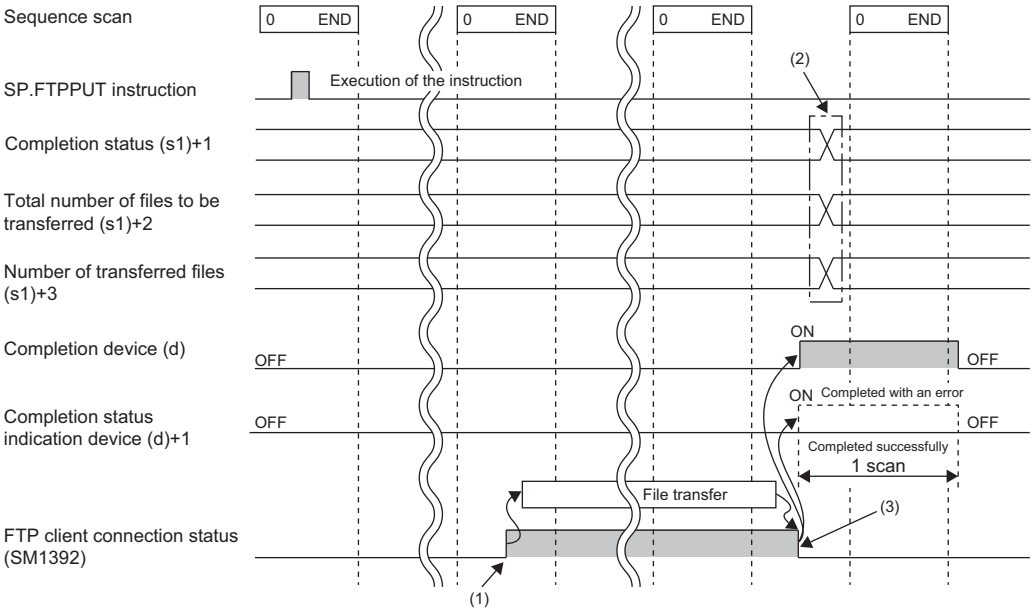
(1)Use one-byte '\' or '/' as a delimiter to specify the boundary of the folder path. *1
(2)The delimiter at the end of string can be omitted.

*1 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 CPU module 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 send is 4G bytes.
- The execution status and the completion status of the SP.FTPPUT instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

• Completion device (d)
This device turns on during END processing of the scan where the SP.FTPPUT 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 the SP.FTPPUT instruction.
When completed successfully: The device remains off.
When completed with an error: The device turns on during END processing of the scan where the SP.FTPPUT 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 the SP.FTPPUT instruction.



(1)SM1392 turns on during the END processing after the CPU module 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 CPU module is connected to the FTP server, and SM1392 turns off when disconnected.
- SM753 (File access in progress) turns on while the SP.FTPPUT instruction is being executed. While SM753 is on, the SP.FTPPUT instruction cannot be executed. If executed, no processing is performed.
- If the SP.FTPPUT instruction is executed while the SP.FTPPUT or SP.FTPGET instruction is being executed, the instruction is ignored and not executed until the currently executing instruction completes. When the instruction is ignored, SM699 (Dedicated instruction skip flag) turns on.
- In the following cases, the instruction completes with an error: when there is no free space in the transfer destination; or when b3 (Temporary file create setting) of (s1)+0 is set to 0 (Create), but there is not enough free space for storing transfer-target files and a temporary file (same size as transfer-target files) in the transfer destination.
- Even though the operating status of the CPU module is switched from RUN to STOP during the file transfer processing, the CPU module continues the processing until completed.
- For the Process CPU (redundant mode), the system A and system B communicate using separate IP addresses.

Precautions

- If a cable is disconnected, power is shut off, or the CPU module 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.
- When b2 (Transfer completion file delete setting) of (s1)+0 is set to 1 (Delete), note the following.

Item	Description
When files in the CPU module are transferred	Files required for the CPU module to operate are also deleted. If deleted, operations of the CPU module 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, the SP.FTPPUT instruction completes with an error. Note, however, that the 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 b3 (Temporary file create setting) of (s1)+0 is set to 0 (Create), a temporary file of 12 characters (FTPCLI_I.TMP) will be created in the transfer destination. Therefore, set the folder path so that the total number of characters in the folder path and the temporary file does not exceed the maximum path length supported by the FTP server.
- 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 CPU module, 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.
- The Process CPU (redundant mode) may not be able to open a connection with the FTP server in the new control system immediately after system switching has done during the file transfer processing, because the connection with the old control system is still not closed at the FTP server. In this case, keep an attempt to execute the SP.FTPPUT instruction until the connection with the FTP server becomes open.

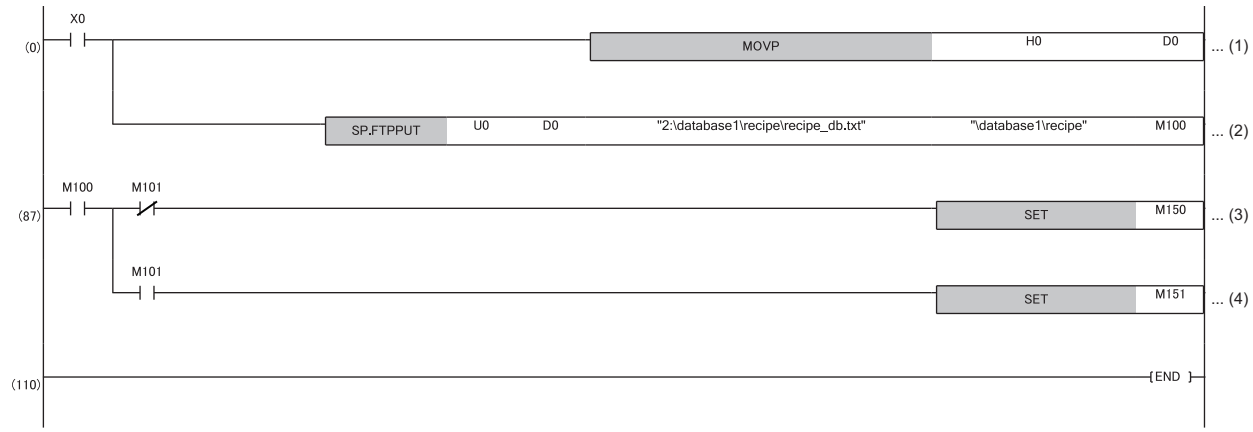
Operation error

Error code (SD0)	Description
2820H	There is no NULL code (0000H) in each setting area in the device/label memory in device specified by (s2), (s3) and later.
3405H	The number of characters in the string specified by (s2) exceeds 255.
	The number of characters in the path specified by (s2), 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.
	The drive number specified by (s2) is out of range.
	When b15 of Application setting area of (s1)+0 is on, the data stored in (s1)+4 to 171 is out of range. • The value in (s1)+4 is other than the valid values (0000H, 0001H). • No character is specified in (s1)+5 to (s1)+133. Or 00H is not stored in the one-byte area immediately after the last character (256 characters maximum). • The values in (s1)+134 to (s1)+135 are out of the following range. (000.000.000.001 to 223.255.255.254) • No character is specified in (s1)+136 to (s1)+152. Or 00H is not stored in the one-byte area immediately after the last character (32 characters maximum). • For the characters specified in (s1)+153 to (s1)+169, 00H is not stored in the one-byte area immediately after the last character (32 characters maximum). • The value in (s1)+170 is other than the valid values (0000H, 0001H). • The value in (s1)+171 is out of the following range. (1 to 65535)
3426H	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).
3430H	The SP.FTPPUT instruction was executed without setting FTP client parameters. (Except when the FTP server specification setting ((s1)+0, Application setting area) is set to Specify.)

■ Program example

• When using the FTP server specified in the module parameter ("FTP Client Settings")

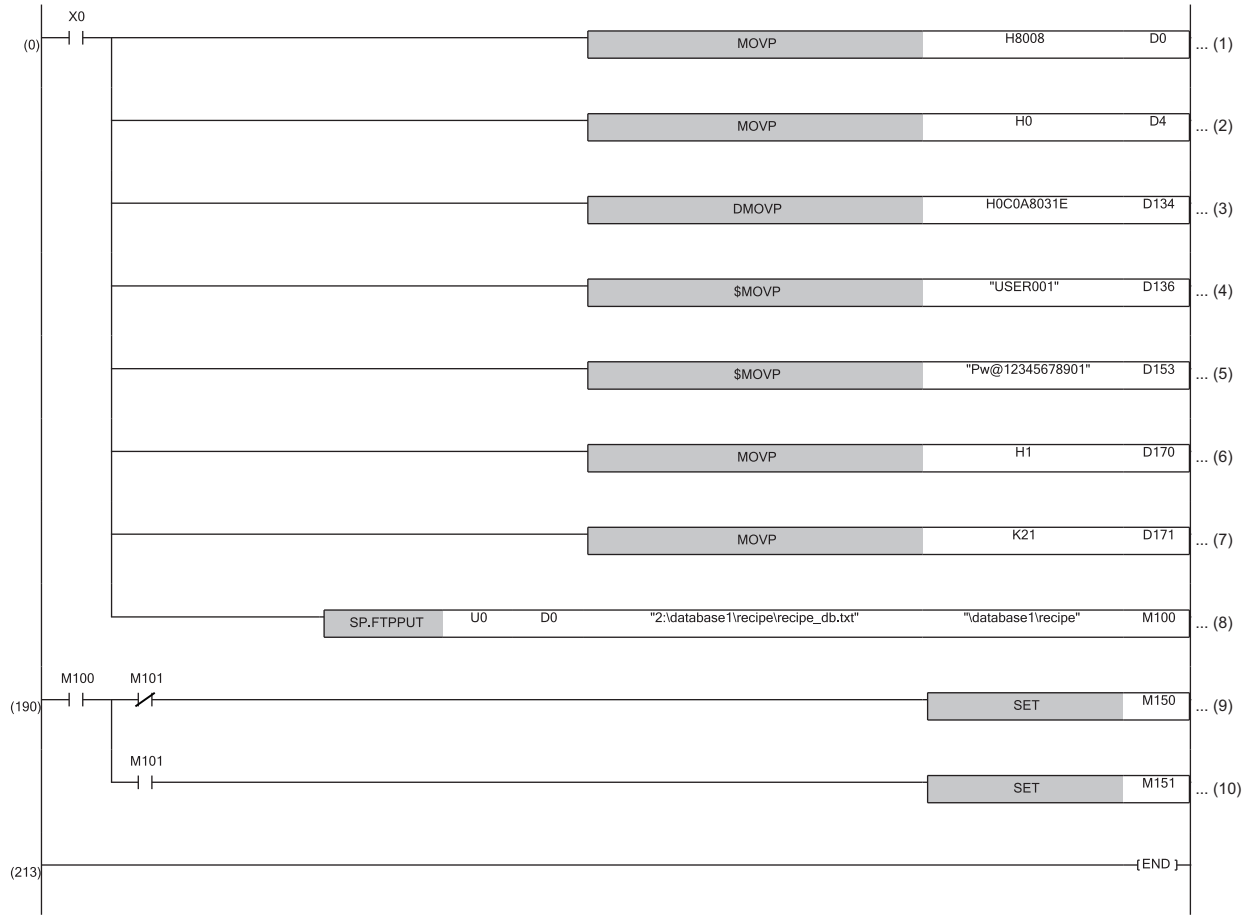
The program that sends the file stored in the CPU module drive 2 "\database1\recipe\recipe_db.txt" to the FTP server "\database1\recipe\" when X0 turns on



(1) Sets Application setting area.
(2) Executes the SP.FTPPUT instruction.
(3) Displays Completed successfully.
(4) Displays Completed with an error.

• When specifying the FTP server with the control data of the SP.FTPPUT instruction

The program that sends the file stored in the CPU module drive 2 "\database1\recipe\recipe_db.txt" to the "\database1\recipe\" of the FTP server address "192.168.3.30" when X0 turns on



(1) Sets Application setting area.
(2) Sets an FTP server.
(3) Sets an FTP server address.
(4) Sets a login name.
(5) Sets a password.
(6) Sets a connection method for the FTP server.
(7) Sets a port number
(8) Executes the SP.FTPPUT instruction.
(9) Displays Completed successfully.
(10)Displays Completed with an error.

Retrieving FTP client files

SP.FTPGET

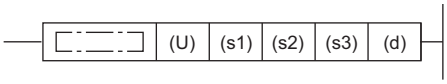
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "22" or later support this instruction. Use an engineering tool with version "1.025B" or later.
- The RnPCPU with firmware version "13" or later supports this instruction. Use an engineering tool with version "1.040S" or later.

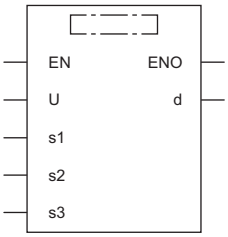
This instruction retrieves files on the FTP server, which are specified by (s2), to the folder path of the CPU module, which is specified by (s3).

Note that the FTP server can be specified by (s1) as well. *1

*1 When specifying the FTP server by the control data of file transfer function instructions, check the version of the CPU module used. (📖MELSEC iQ-R Programmable Controller CPU Module User's Manual)

Ladder	ST
	ENO:=SP_FTPGET(EN,U,s1,s2,s3,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.FTPGET	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(U)	Dummy	—	String	ANYSTRING_SINGLE
(s1)	Start device where control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 4 or 172)*3
(s2)	Name of files stored in the FTP server (transfer source)*1	—	Unicode string*2	ANYSTRING_DOUBLE
(s3)	Folder path of the CPU module (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 "_".

*3 The necessary number of array elements depends on the FTP server specification setting of control data.
·When b15 of (s1)+0 is 0: Number of elements is 4
·When b15 of (s1)+0 is 1: Number of elements is 172

■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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U)	—	—	○	—	—	—	—	○	—	—	○	○
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	○	—
(s3)	—	—	○	—	—	—	—	○	—	—	○	—
(d)	○	—	○*1	—	—	—	—	○	—	—	—	—

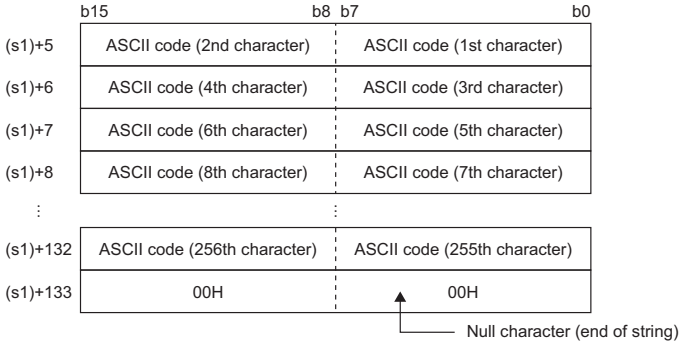
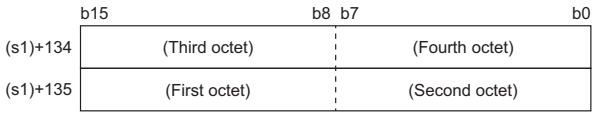
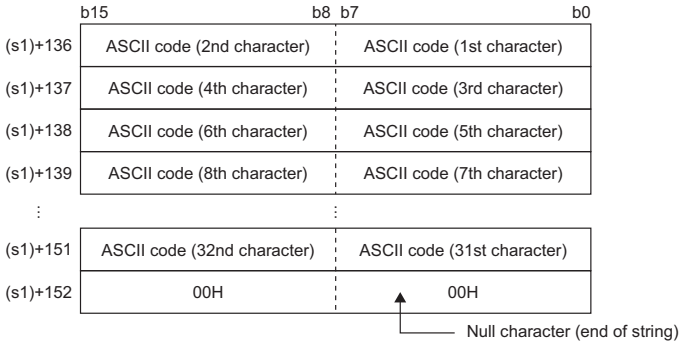
*1 T, ST, C, and FD cannot be used.

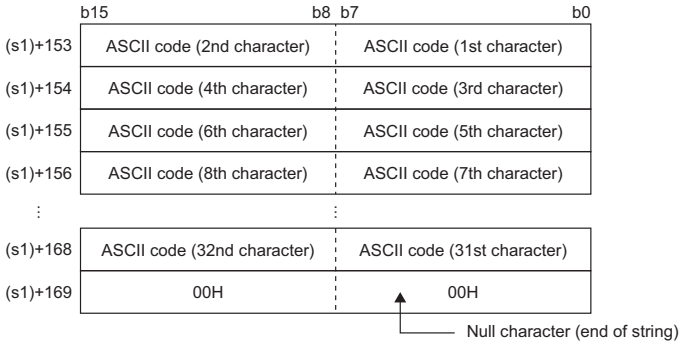
■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Application setting area	b15 … b3 b2 b1 b0 (3) 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 (FTPCLI_I.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 (3) FTP server specification setting (b15) Set whether to specify the FTP server by control data. When Do not specify is set, the FTP server specified with the module parameter is used. When Specify is set, the FTP server specified with (s1)+4 to 171 is used. • 0: Do not specify (the setting of (s1)+4 to 171 is not required) • 1: Specify (the setting of (s1)+4 to 171 is required)	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 the SP.FTPGET instruction is stored.	—	System
+3	Number of transferred files	The number of transferred files is stored.	—	System

The following table shows the control data that needs to be specified when the FTP server specification setting ((s1)+0, Application setting area, b15) is 1: Specify.

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+4	FTP server specification	Set an FTP server specification method. • 0000H: FTP server address • 0001H: FTP server name	0000H, 0001H	User

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+5 to +133	FTP server name	Set an FTP server name (domain name) with an ASCII character string of the following range: • 1 to 256 characters (one-byte alphanumeric characters, one-byte symbols) The first character is stored in the lower byte of (s1)+5, and the second character in the upper byte of (s1)+5. 00H is stored immediately after the last character. (The characters until 00H are recognized as a character string.) Note that this setting is not required when specifying "0000H: FTP server address" for FTP server specification of (s1)+4.  	1 to 256 characters (one-byte alphanumeric characters, one-byte symbols)	User
+134 +135	FTP server address	Set an IP address of the FTP server within the following range. • 000.000.000.001 to 223.255.255.254 The first octet is stored in the upper byte of (s1)+135, the second octet in the lower byte of (s1)+135, the third octet in the upper byte of (s1)+134, and the fourth octet in the lower byte of (s1)+134. Note that this setting is not required when specifying "0001H: FTP server name" for FTP server specification of (s1)+4.  	000.000.000.001 to 223.255.255.254	User
+136 to +152	Login name	Set a login name for accessing the FTP server with an ASCII character string of the following range: • 1 to 32 characters (one-byte alphanumeric characters, one-byte symbols) The first character is stored in the lower byte of (s1)+136, and the second character in the upper byte of (s1)+136. 00H is stored immediately after the last character. (The characters until 00H are recognized as a character string.)  	1 to 32 characters (one-byte alphanumeric characters, one-byte symbols)	User

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+153 to +169	Password	Set a password for logging in to the FTP server with an ASCII character string of the following range: 0 to 32 characters (one-byte alphanumeric characters, one-byte symbols) The first character is stored in the lower byte of (s1)+153, and the second character in the upper byte of (s1)+153. 00H is stored immediately after the last character. (The characters until 00H are recognized as a character string.)  	0 to 32 characters (one-byte alphanumeric characters, one-byte symbols)	User
+170	Connection method	Set a connection method for the FTP server. • 0000H: PORT mode • 0001H: PASV mode	0000H, 0001H	User
+171	Port number	Set an FTP server control port number within the following range. • 1 to 65535 (0001H to FFFFH)	1 to 65535 (0001H to FFFFH)	User

Processing details

- When the FTP server is specified with the module parameter (b15 of (s1)+0 is "0: Do not specify"), this instruction retrieves files on the FTP server specified by (s2) to the folder path of the CPU module specified by (s3). The CPU module opens a connection with the FTP server set in the module parameters ("FTP Client Settings") at execution of the instruction, and closes a connection after retrieving files. For details on the parameter setting, refer to the following.

📖MELSEC iQ-R Ethernet User's Manual (Application)

- When specifying the FTP server with the control data of the SP.FTPGET instruction (b15 of (s1)+0 is "1: Specify"), specify the server by (s1)+4 to 171.

· Specify the FTP server name, login name, and password in (s1)+4 to 171 using an ASCII character string, and store 00H immediately after the last character. The setting examples are shown below.

Ex.

When setting "SERV001" for FTP server name ((s1)+5~). The start device of the control data (s1) is D0.

	b15 b8	b7 b0
(s1)+5	45H (E)	53H (S)
(s1)+6	56H (V)	52H (R)
(s1)+7	30H (0)	30H (0)
(s1)+8	00H	31H (1)

[Input in the program]

When setting "USER001" for Login name ((s1)+136~). The start device of the control data (s1) is D0.

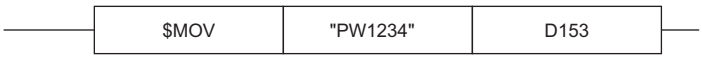
	b15 b8	b7 b0
(s1)+136	53H (S)	55H (U)
(s1)+137	52H (R)	45H (E)
(s1)+138	30H (0)	30H (0)
(s1)+139	00H	31H (1)

[Input in the program]

When setting "PW1234" for Password ((s1)+153~). The start device of the control data (s1) is D0.

	b15 b8	b7 b0
(s1)+153	57H (W)	50H (P)
(s1)+154	32H (2)	31H (1)
(s1)+155	34H (4)	33H (3)
(s1)+156	00H	00H

[Input in the program]

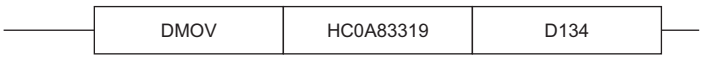


Set the FTP server address in (s1)+134 to 135. Each part of the IP address, which is separated by a dot, should be represented as a single byte in hexadecimal format. The setting example is shown below.

Ex.

	b15 b8	b7 b0
(s1)+134	33H (51)	19H (25)
(s1)+135	C0H (192)	A8H (168)

[Input in the program]

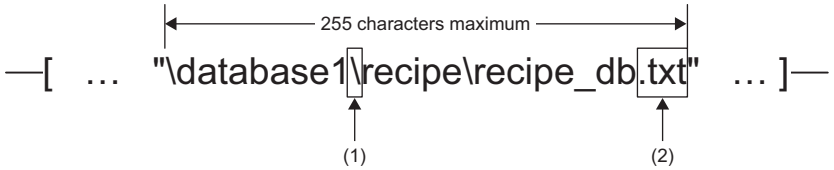


· The following FTP server has been verified to operate correctly when specified with the control data of the SP.FTPGET instruction.

FTP server	Applicable operating system
Microsoft® Internet Information Services (IIS)	Microsoft® Windows® 11 Microsoft® Windows® 10

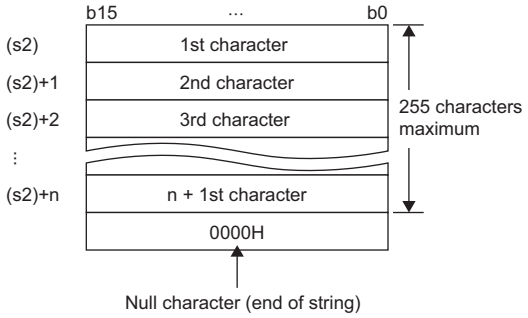
· When specifying the FTP server name, the DNS setting of the module parameter is required. (📖MELSEC iQ-R Ethernet User's Manual (Application))

- The total number of files to be transferred by the SP.FTPGET instruction is stored in (s1)+2, and the number of transferred files is stored in (s1)+3.
- Specify the folder path where the transfer source files are stored on the FTP server, and the file name (including an extension) in (s2) in Unicode string. The maximum number of characters used in a file path is 255. The specified file path shall be a relative path from home directory of the FTP server. Use one-byte '\' or '/' as a delimiter to specify the boundary of the folder path or the file name.*1



(1)Use one-byte '\' or '/' as a delimiter of the folder path or file.*1
(2)The specified file name should include an extension.

*1 Note that '\' cannot be used as a delimiter for some FTP servers.



- Wild card characters (*, ?) can be used in the file name or the extension specified in (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.)

Wild card characters do not recognize periods.

Using wild card characters in the following ways results in an error.

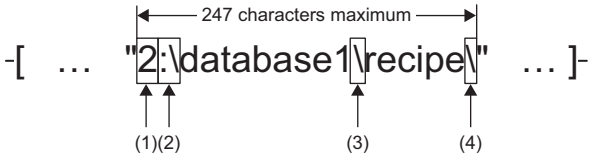
- Two or more asterisks '*' are used in a file name (before the period) or an extension. (Example: *abc*.txt)
- An asterisk '*' and a question mark '?' are used in a file name (before the period) or an extension. (Example: *ab?. txt)

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 retrieved during execution of the SP.FTPGET instruction, the transfer processing will be stopped upon detection of the error, and the rest of 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 only a file name is specified by (s2), the CPU module 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 CPU module 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 CPU module. 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.



(1)Drive numbers that can be specified are 2 to 4. Specify 4 for the R00CPU.
(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.

- 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 4G bytes.
- The execution status and the completion status of the SP.FTPGET instruction can be checked with the completion device (d) and the completion status indication device (d)+1.

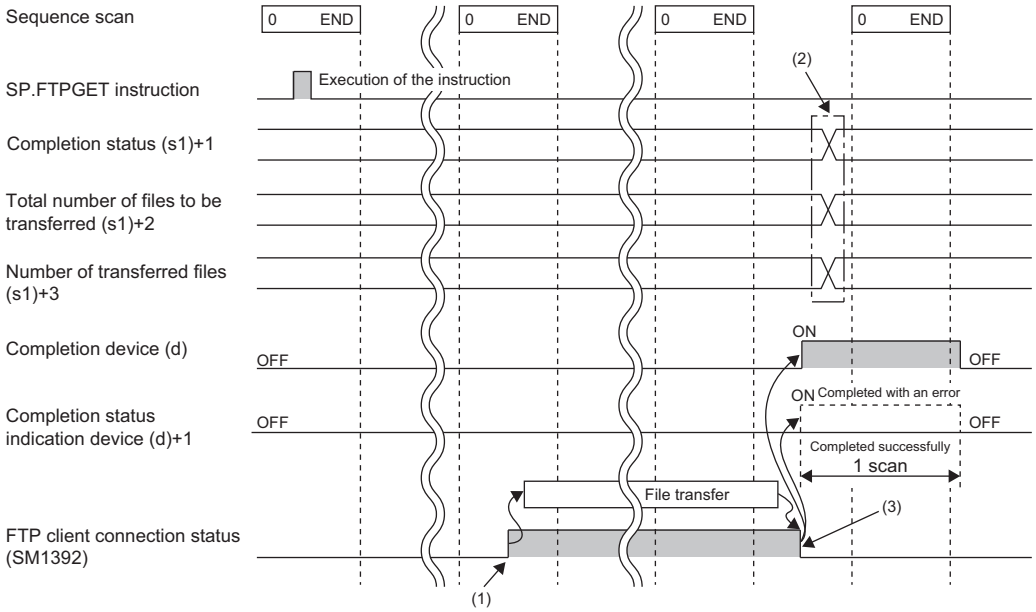
- Completion device (d)

This device turns on during END processing of the scan where the SP.FTPGET 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 the SP.FTPGET instruction.
When completed successfully: The device remains off.
When completed with an error: The device turns on during END processing of the scan where the SP.FTPGET 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 the SP.FTPGET instruction.



(1)SM1392 turns on during the END processing after the CPU module 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 CPU module is connected to the FTP server, and SM1392 turns off when disconnected.
• SM753 (File access in progress) turns on while the SP.FTPGET instruction is being executed. While SM753 is on, the SP.FTPGET instruction cannot be executed. If executed, no processing is performed.
• If the SP.FTPGET instruction is executed while the SP.FTPPUT or SP.FTPGET instruction is being executed, the instruction is ignored and not executed until the currently executing instruction completes. When the instruction is ignored, SM699 (Dedicated instruction skip flag) turns on.
• In the following cases, the instruction completes with an error: when there is no free space in the transfer destination; or when b3 (Temporary file create setting) of (s1)+0 is set to 0 (Create), but there is not enough free space for storing transfer-target files and a temporary file (same size as transfer-target files) in the transfer destination.
• Even though the operating status of the CPU module is switched from RUN to STOP during the file transfer processing, the CPU module continues the processing until completed.
• For the Process CPU (redundant mode), the system A and system B communicate using separate IP addresses.

Precautions

- If a cable is disconnected, power is shut off, or the CPU module is reset during the file transfer processing, transfer files again. In those cases, unnecessary files will be deleted in the following ways.

Unnecessary file	Delete operation
Temporary file (FTPCLI_I.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, the SP.FTPGET 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 CPU module to 243 (242 when a delimiter at the end of the string is omitted). Since a temporary file of 12 characters (FTPCLI_I.TMP) is created in the transfer destination, set the 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 CPU module.
- 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.)
- The Process CPU (redundant mode) may not be able to open a connection with the FTP server in the new control system immediately after system switching has done during the file transfer processing, because the connection with the old control system is still not closed at the FTP server. In this case, keep an attempt to execute the SP.FTPGET instruction until the connection with the FTP server becomes open.

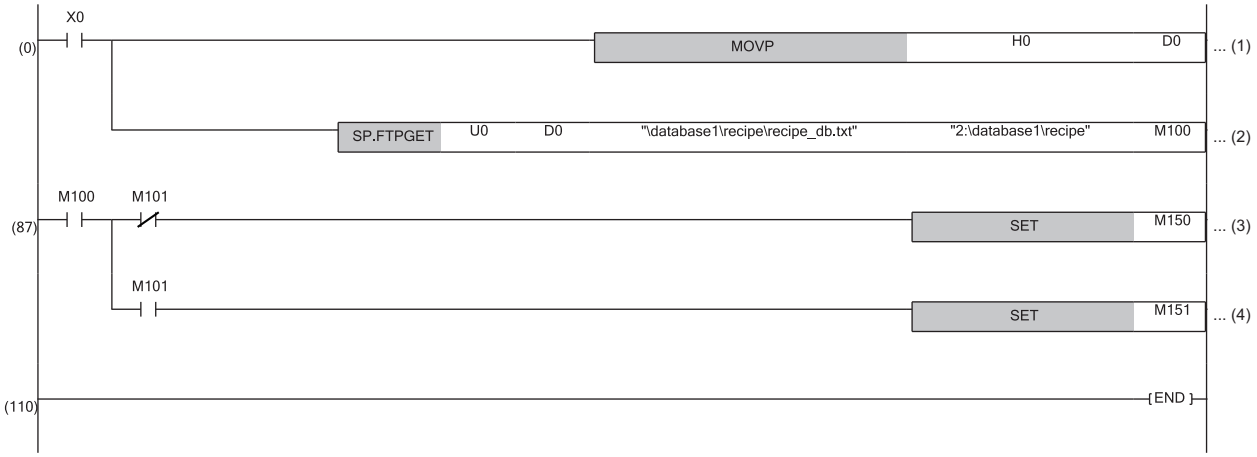
Operation error

Error code (SD0)	Description
2820H	There is no NULL code (0000H) in each setting area in the device/label memory in device specified by (s2), (s3) and later.
3405H	The number of characters in the string specified by (s2) exceeds 255.
	The number of characters in the path specified by (s2), 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.
	The drive number specified by (s3) is out of range.
	When b15 of Application setting area of (s1)+0 is on, the data stored in (s1)+4 to 171 is out of range. • The value in (s1)+4 is other than the valid values (0000H, 0001H). • No character is specified in (s1)+5 to (s1)+133. Or 00H is not stored in the one-byte area immediately after the last character (256 characters maximum). • The values in (s1)+134 to (s1)+135 are out of the following range. (000.000.000.001 to 223.255.255.254) • No character is specified in (s1)+136 to (s1)+152. Or 00H is not stored in the one-byte area immediately after the last character (32 characters maximum). • For the characters specified in (s1)+153 to (s1)+169, 00H is not stored in the one-byte area immediately after the last character (32 characters maximum). • The value in (s1)+170 is other than the valid values (0000H, 0001H). • The value in (s1)+171 is out of the following range. (1 to 65535)
3426H	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).
3430H	The SP.FTPGET instruction was executed without setting FTP client parameters. (Except when the FTP server specification setting ((s1)+0, Application setting area) is set to Specify.)

■ Program example

• When using the FTP server specified in the module parameter ("FTP Client Settings")

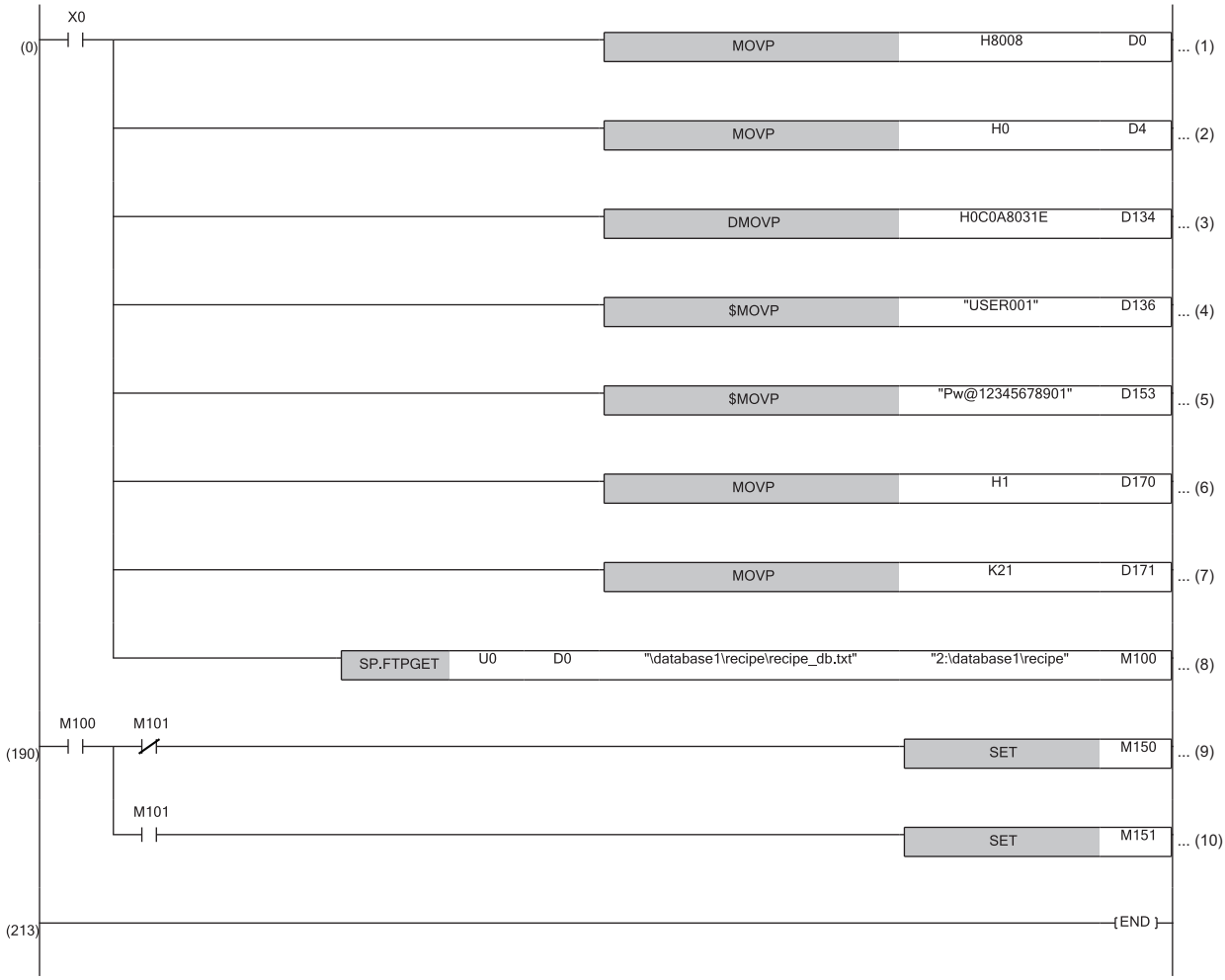
The program that retrieves the file "\database1\recipe\recipe_db.txt" stored in the FTP server "\database1\recipe\" to the CPU module drive 2 "\database1\recipe\" when X0 turns on.



(1) Sets Application setting area.
(2) Executes the SP.FTPGET instruction.
(3) Displays Completed successfully.
(4) Displays Completed with an error.

• When specifying the FTP server with the control data of the SP.FTPGET instruction

The program that retrieves the file "\database1\recipe\recipe_db.txt" stored in the FTP server address "192.168.3.30" "\database1\recipe\" to the CPU module drive 2 "\database1\recipe\" when X0 turns on.



(1) Sets Application setting area.
(2) Sets an FTP server.
(3) Sets an FTP server address.
(4) Sets a login name.
(5) Sets a password.
(6) Sets a connection method for the FTP server.
(7) Sets a port number
(8) Executes the SP.FTPGET instruction.
(9) Displays Completed successfully.
(10)Displays Completed with an error.

25 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 1234 Overview	Page 1245 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 1249 Overview	Page 1258 PID Control Instructions (Inexact Differential), Page 1270 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

25.1 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

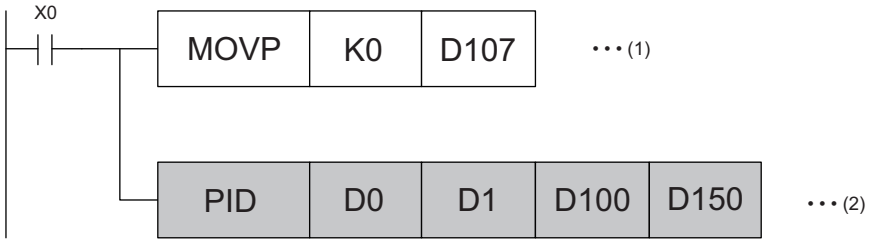
The maximum error of the sampling time (T_S) is -(1 operation cycle + 1ms) to +(1 operation cycle).

- 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 1241 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	

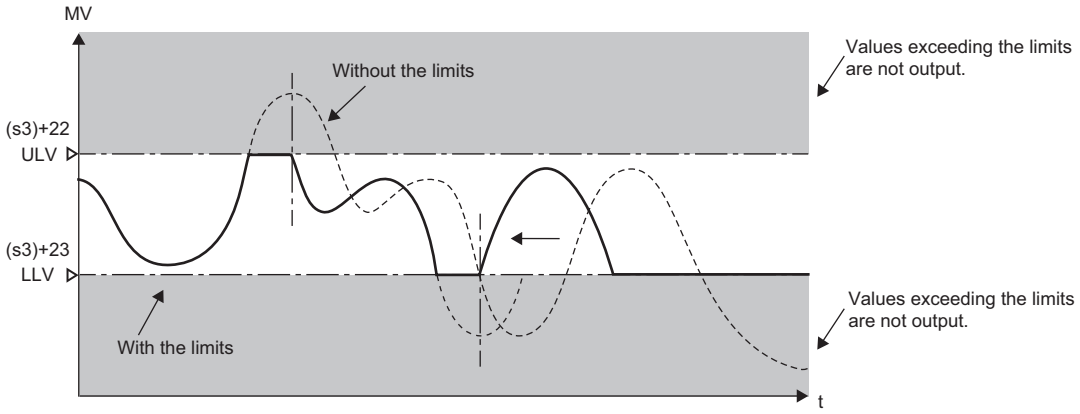
- Output variation (bit 2 of (s3)+1)

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

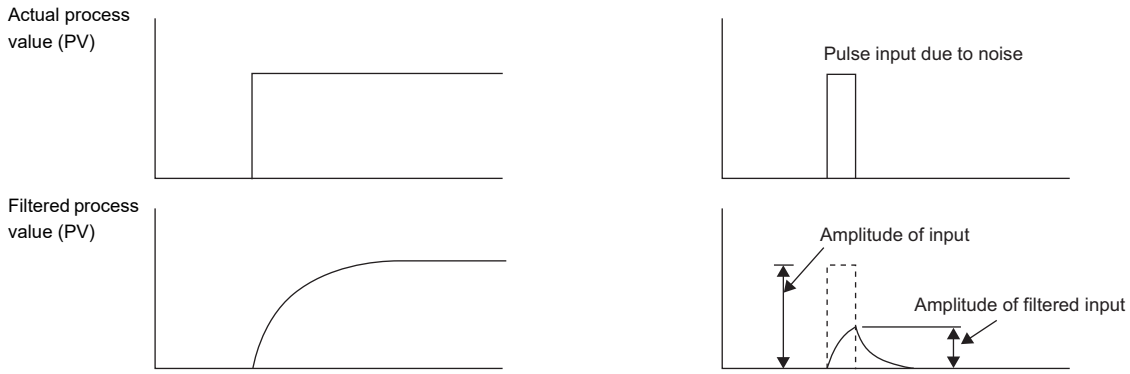
Setting range: 0 to 99 [%]

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.

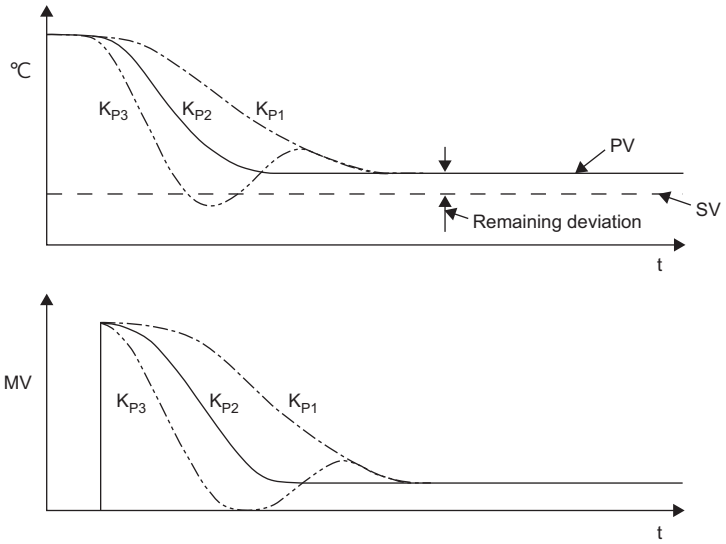
Manipulated value (MV) = Proportional gain (K_P) × Deviation (EV)

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)

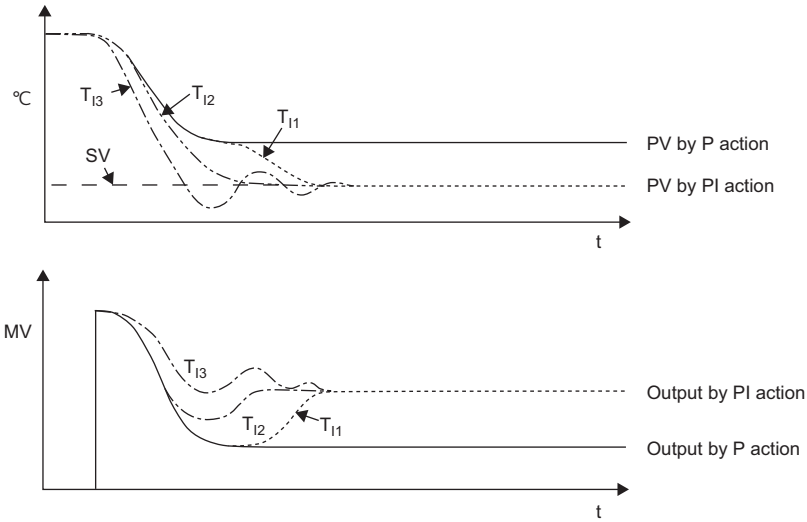
PID control: Integral action

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.

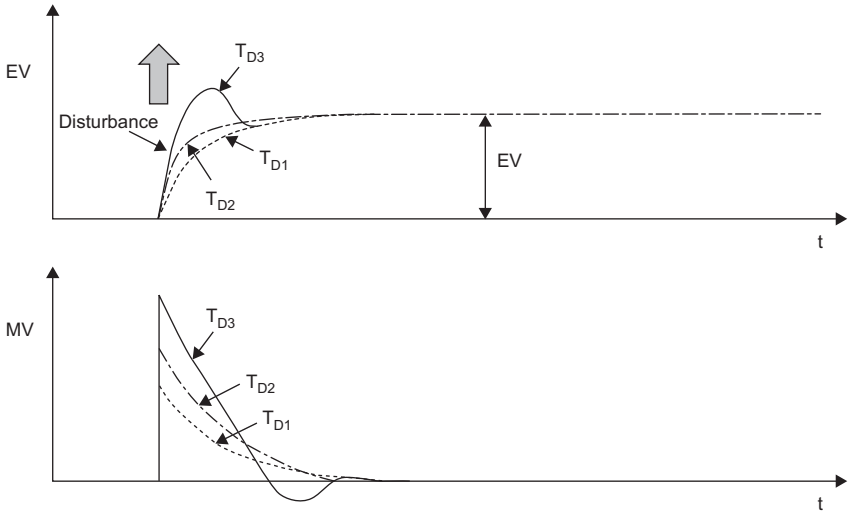
Derivative time (T_D): (s3)+6

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.

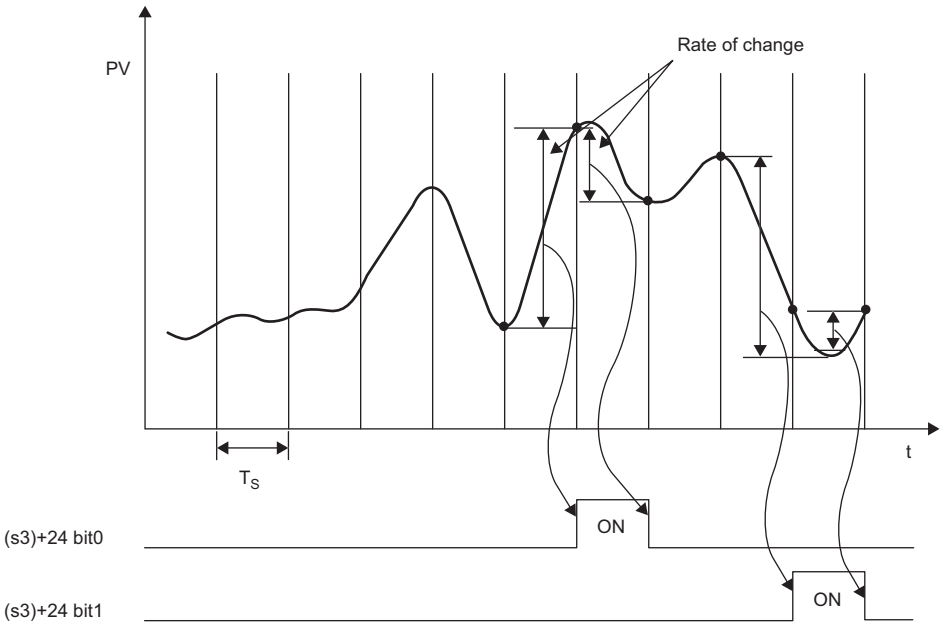


Derivative time (T_D): $T_{D3} > T_{D2} > T_{D1}$
EV: Deviation
MV: Manipulated value
t: Time

Alarm output flag: (s3)+24

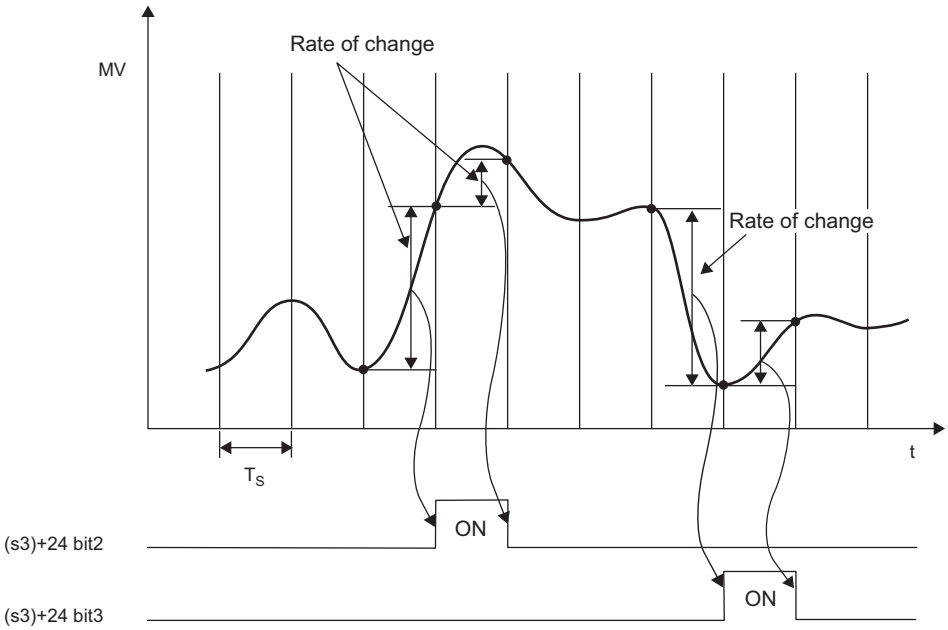
When the set input/output variation is exceeded, bits of (s3)+24 turn on as an alarm flag immediately after execution of the PID operation instruction.

- 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 1611 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 1612 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 1000 ms 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.

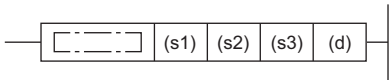
25.2 PID Operation Instruction

PID

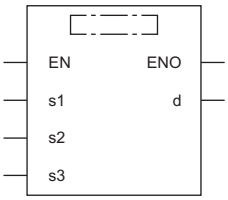
RnCPU | RnENCPU | ~~RnPCPU (Process)~~ | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU and RnENCPU (excluding the R00CPU, R01CPU, and R02CPU) with firmware version "17" or later support this instruction. (Use an engineering tool with version "1.020W" or later.)

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 1234 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) ❶ The device areas of 25 points starting from the device specified in (s3) are used when all of the bits 1, 2, and 5 of (s3)+1 (Action setting (ACT)) are not 0. ❷ The device areas of 20 points starting from the device specified in (s3) are occupied when all of the bits 1, 2, and 5 of (s3)+1 (Action setting (ACT)) are 0.	❶ 25 points ❷ 20 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	Enabled when the bit 1 of (s3)+1 (Action direction (ACT)) is 1.
+21*1	Input variation (decrease) alarm setting value		0 to 32767	Enabled when the bit 1 of (s3)+1 (Action direction (ACT)) is 1.

Operand: (s3)				
Device	**Item**		**Description**	**Remarks**
+22*1	Output variation (increase) alarm setting value		0 to 32767	Enabled when the bit 2 of (s3)+1 (Action direction (ACT)) is 1, and the bit 5 is 0.
	Output upper limit setting value		-32768 to 32767	Enabled when the bit 2 of (s3)+1 (Action direction (ACT)) is 0, and the bit 5 is 1.
+23*1	Output variation (decrease) alarm setting value		0 to 32767	Enabled when the bit 2 of (s3)+1 (Action direction (ACT)) is 1, and the bit 5 is 0.
	Output lower limit setting value		-32768 to 32767	Enabled when the bit 2 of (s3)+1 (Action direction (ACT)) is 0, and the bit 5 is 1.
+24*1	Alarm output	Bit 0	0: Input variation (increase) not exceeded 1: Input variation (increase) exceeded	Enabled when the bit 1 of (s3)+1 (Action direction (ACT)) is 1, or the 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).	Occupied when the bit 6 of (s3)+1 (Action direction (ACT)) is 1 (limit cycle method).
+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)		-5 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.

Error code (SD0)	Description
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.

26 PID CONTROL INSTRUCTIONS

There are two types of instructions for PID control.

- PID OPERATION INSTRUCTIONS
- PID CONTROL INSTRUCTIONS

For how to use or compare them, refer to the following.

☞ Page 1234 PID operation instruction and PID control instruction

26.1 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.

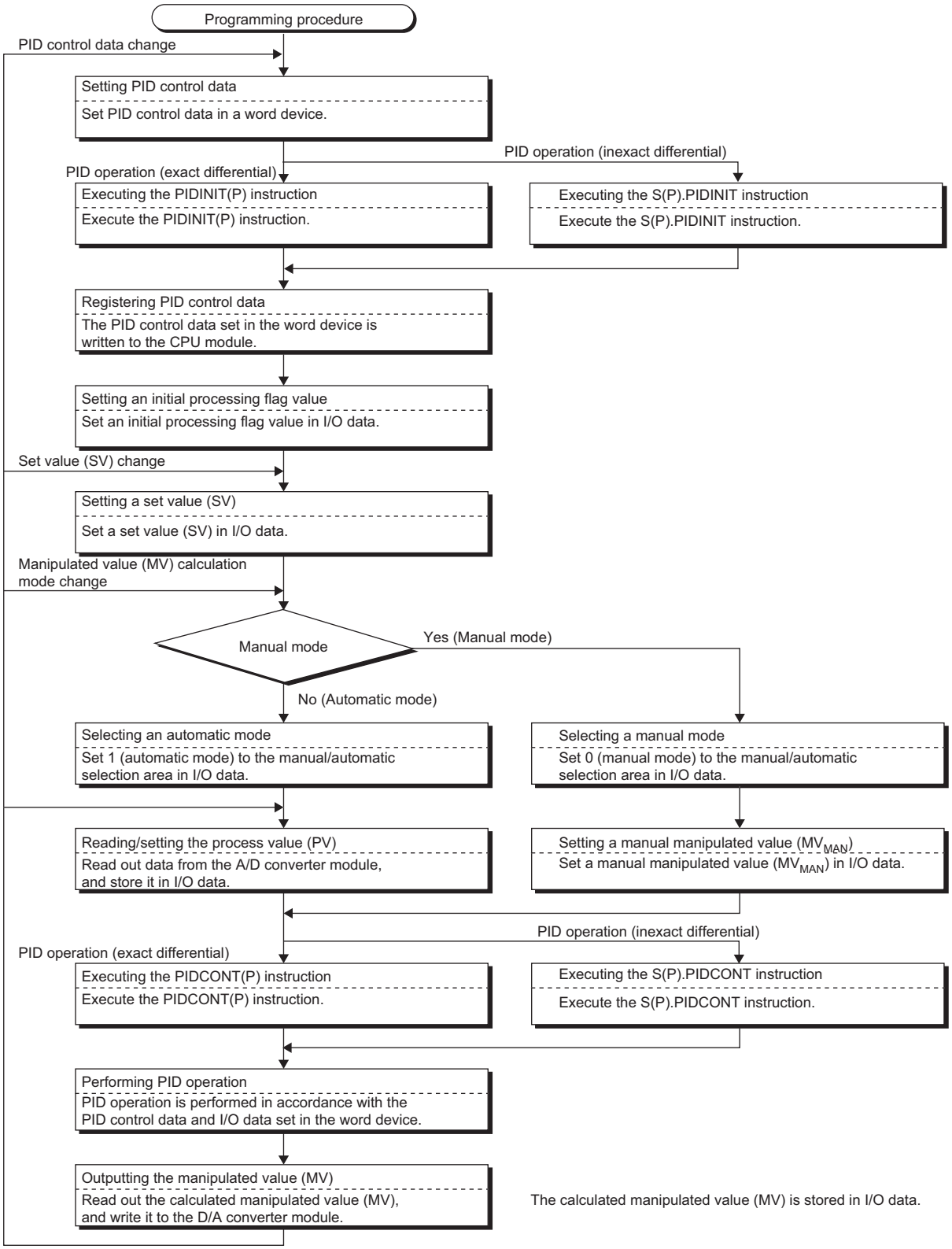
Operation method

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 CPU module 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 1260 S(P).PIDINIT
- Exact differential: Page 1272 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	
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.

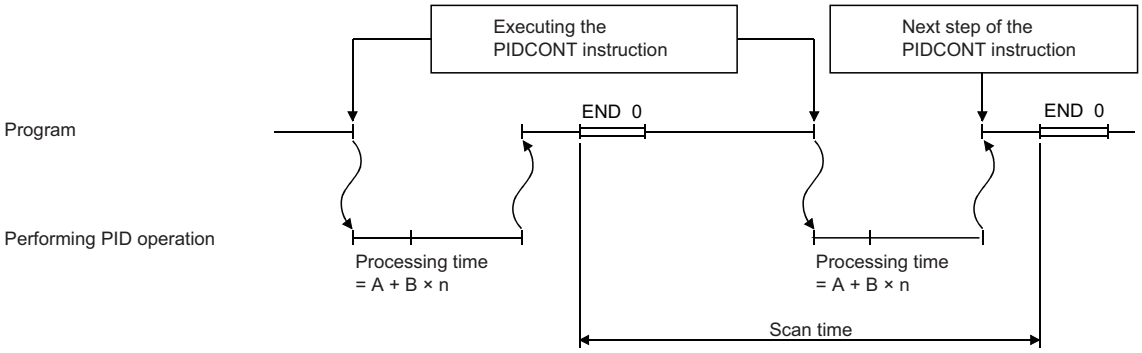
*1 When the PID limit is restricted
*2 When the PID limit is not restricted
*3 For PID control (inexact differential) only

■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.

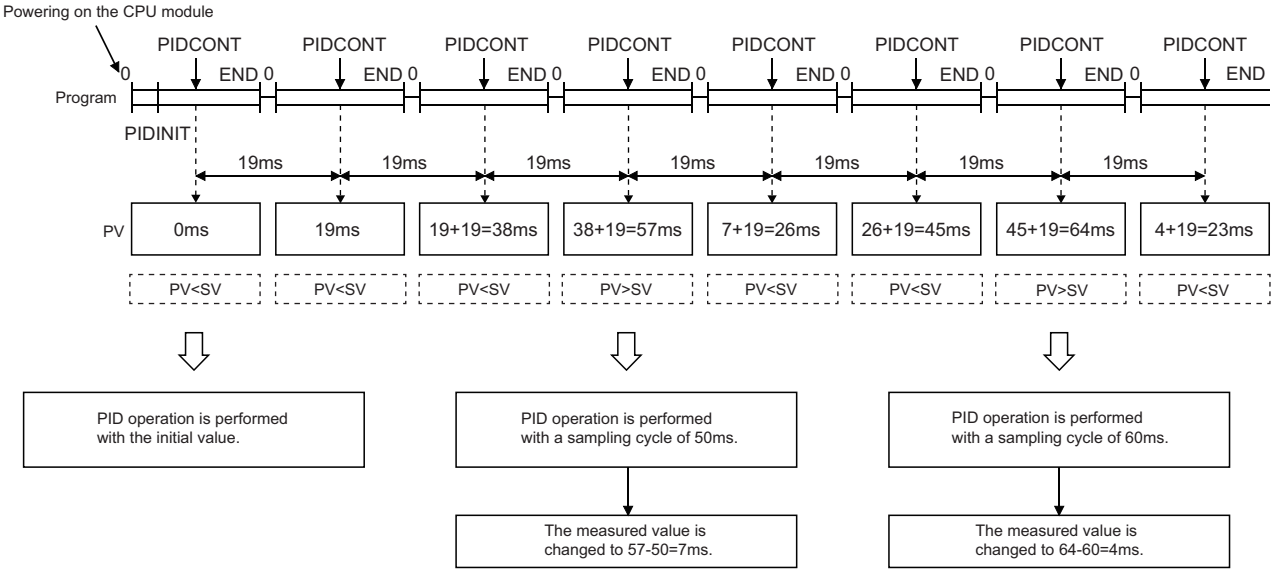
■Sampling period

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 1263 S(P).PIDCONT
- Exact differential: Page 1274 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 CPU module is reset. b15 ··· b0 (2) (1) (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 1255 I/O data)

Point

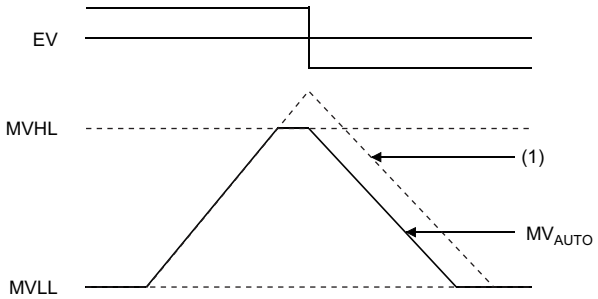
PID control is performed in each mode as described below.

- Automatic mode: The CPU module controls the target with the MV calculated by PID operation using the PID control instructions.
- Manual mode: The CPU module 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	b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 b0: Loop 1 b1: Loop 2 ⋮ b14: Loop 15 b15: Loop 16
SD795	SD793	b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0 b0: Loop 17 b1: Loop 18 ⋮ b14: Loop 31 b15: Loop 32

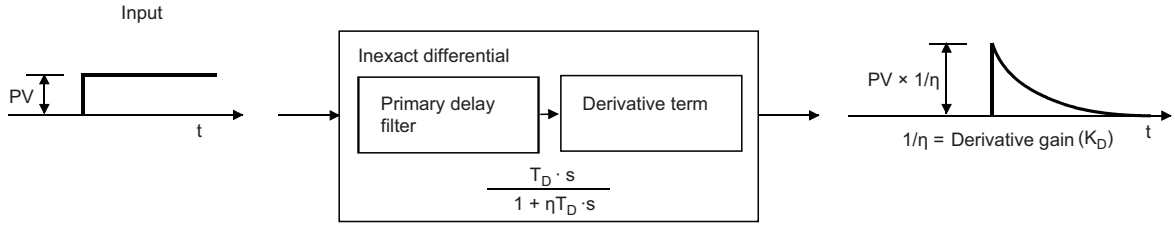
0: PID limit restricted (default)

1: PID limit not restricted

26.2 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

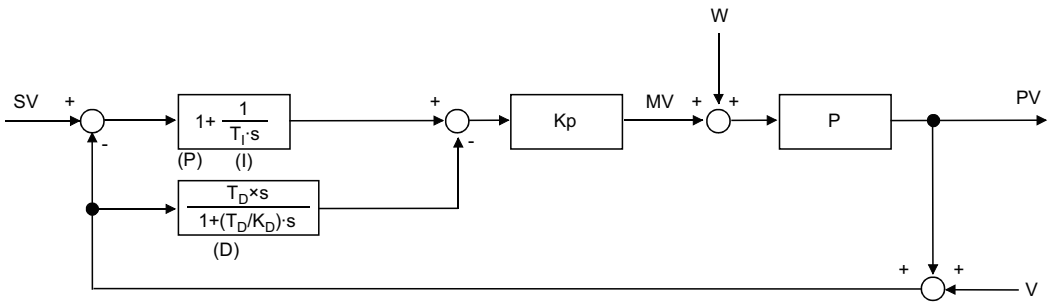


The following table summarizes the performance specifications of PID control instructions for inexact differential.

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 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
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 filter coefficient is not set to the input data, the PV_{fn} will be same as the process value of the input data.

$PV_{fn} = PV_n + \alpha(PV_{fn-1} - PV_n)$

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 1255 I/O data)

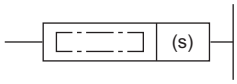
Registering the PID control data to the CPU module

S(P).PIDINIT

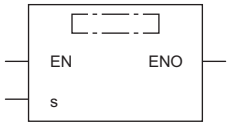
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

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 CPU 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□\□, U3E□\(H)G□	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 by (s) and later altogether in the CPU module to enable PID control. (☞ Page 1252 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
3405H	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.

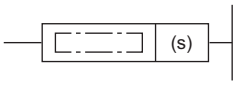
Performing PID operation

S(P).PIDCONT

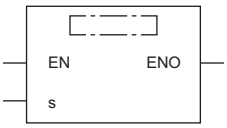
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (MELSEC iQ-R CPU Module User's Manual (Application))

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□\□, U3E□\(H)G□	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 1255 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)

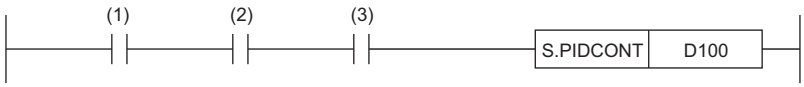
*1 $m = (n-1) \times 23 + 10$

- 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
3405H	The value of the data set in the I/O data area specified by (s) is out of the setting range.
3422H	The S(P).PIDINIT instruction is not executed before the S(P).PIDCONT instruction.

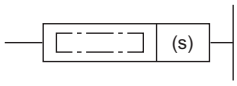
Stopping the operation of specified loop number

S(P).PIDSTOP

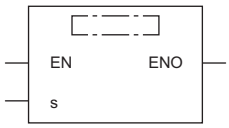
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

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□\□, U3E□\(H)G□	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
3405H	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.
3422H	The S(P).PIDINIT and S(P).PIDCONT instructions are not executed before the S(P).PIDSTOP instruction.

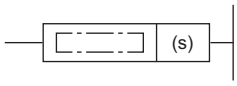
Starting the operation of specified loop number

S(P).PIDRUN

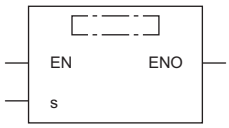
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (MELSEC iQ-R CPU Module User's Manual (Application))

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□\□, U3E□\(H)G□	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
3405H	The loop number specified by (s) does not exist.
	(s) is outside the range from 1 to 32.
3422H	The S(P).PIDINIT and S(P).PIDCONT instructions are not executed before the S(P).PIDRUN instruction.

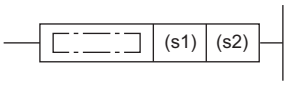
Changing the parameters of specified loop number

S(P).PIDPRMW

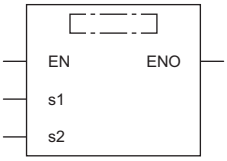
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

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□\□, U3E□\(H)G□	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.

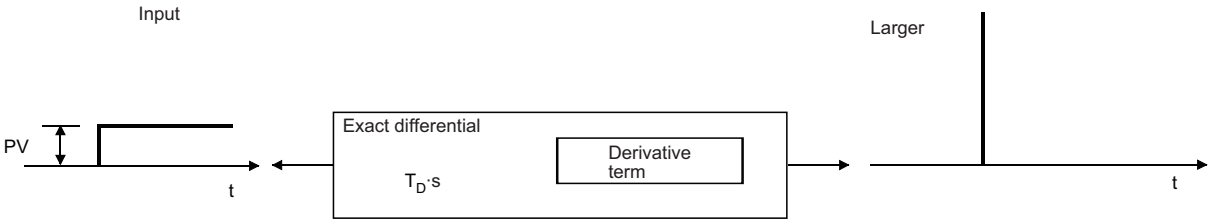
PID control data	
(s2)+0	Operational expression selection
(s2)+1	Sampling period (T_S)
(s2)+2	Proportional constant (K_P)
(s2)+3	Integral constant (T_I)
(s2)+4	Derivative constant (T_D)
(s2)+5	Filter coefficient (α)
(s2)+6	Manipulated value lower limit (MVLL)
(s2)+7	Manipulated value upper limit (MVHL)
(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
3405H	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.
3422H	The S(P).PIDINIT instruction is not executed before the S(P).PIDPRMW instruction.

26.3 PID Control Instructions (Exact Differential)

Exact differential is PID control that uses the input of a differential term as it is.

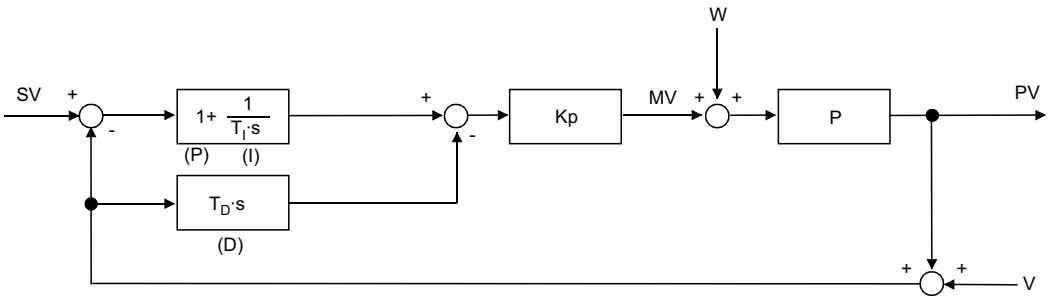


The following table summarizes the performance specifications of PID control instructions for exact differential.

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 two times before (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 filter coefficient is not set to the input data, the PV_{fn} will be same as the process value of the input data.

$PV_{fn} = PV_n + \alpha(PV_{fn-1} - PV_n)$

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 1255 I/O data)

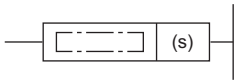
Registering the PID control data to the CPU module

PIDINIT(P)

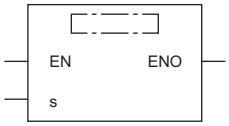
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (MELSEC iQ-R CPU Module User's Manual (Application))

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 CPU 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□\□, U3E□\(H)G□	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 by (s) and later altogether in the CPU module to enable PID control. (☞ Page 1252 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
3405H	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)

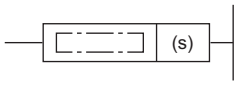
Performing PID operation

PIDCONT(P)

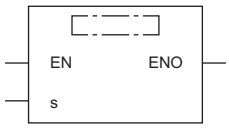
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (MELSEC iQ-R CPU Module User's Manual (Application))

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□\□, U3E□\(H)G□	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 1255 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	Automatic 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)	Automatic 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)

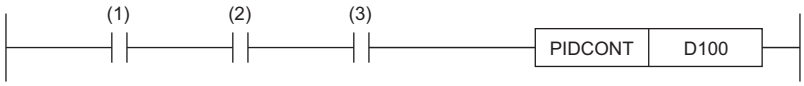
*1 $m = (n-1) \times 18 + 10$

- The number of device points used for I/O data setting is calculated by the following formula.

Number of device points = $10 + 18 \times n$ (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
3405H	The value of the data set in the I/O data area specified by (s) is out of the setting range.
3422H	The PIDINIT(P) instruction is not executed before the PIDCONT(P) instruction.

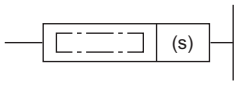
Stopping the operation of specified loop number

PIDSTOP(P)

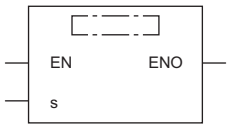
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

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□\□, U3E□\(H)G□	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
3405H	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.
3422H	The PIDINIT(P) and PIDCONT(P) instructions are not executed before the PIDSTOP(P) instruction.

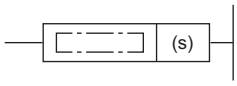
Starting the operation of specified loop number

PIDRUN(P)

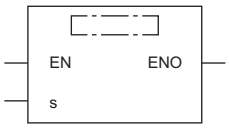
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

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□\□, U3E□\(H)G□	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
3405H	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.
3422H	The PIDINIT(P) and PIDCONT(P) instructions are not executed before the PIDRUN(P) instruction.

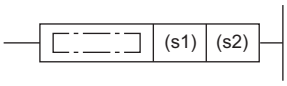
Changing the parameters of specified loop number

PIDPRMW(P)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

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□\□, U3E□\(H)G□	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
3405H	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.
3422H	The PIDINIT(P) instruction is not executed before the PIDPRMW(P) instruction.

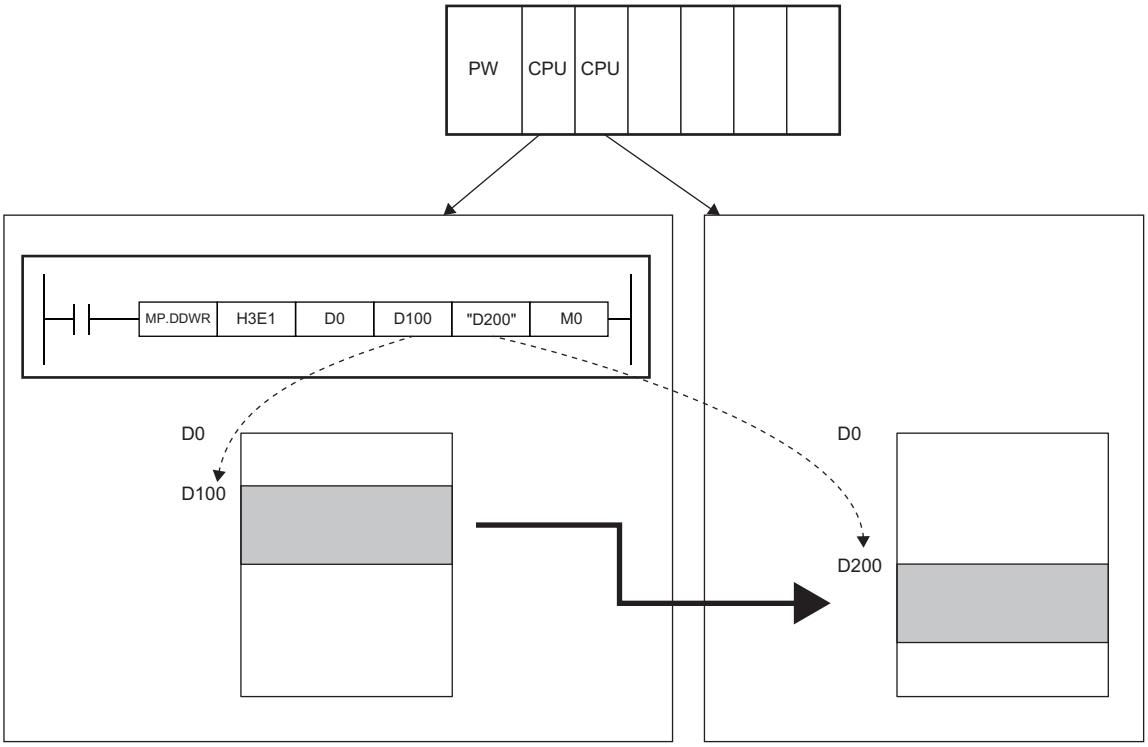
27 MULTIPLE CPU DEDICATED INSTRUCTIONS

27.1 Another CPU Module Access Instructions

Overview

The host CPU module read or write device data from or to another CPU module by using another CPU module access instructions.

The following figure shows the operation for writing data from CPU No.1 to CPU No.2 by using another CPU module access instruction.



The following table lists another CPU module access instructions.

Instruction symbol	Description	Application
D(P).DDRD	Loads the device data of another CPU module to the device of the host CPU module.	Use these instructions to read or write data at the timing set by the fixed scan communication function.
D(P).DDWR	Writes the device data of the host CPU module to the device of another CPU module.	
M(P).DDRD	Loads the device data of another CPU module to the device of the host CPU module.	Use these instructions to read or write data at the timing of each CPU module.
M(P).DDWR	Writes the device data of the host CPU module to the device of another CPU module.	

Setting parameters

To use the D(P).DDRD or D(P).DDWR instruction, the fixed scan communication function of the system parameters needs to be set.

Readable/writable devices

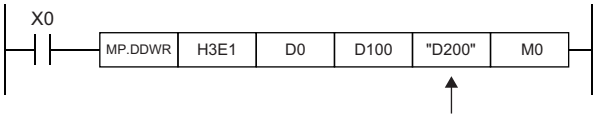
The following table lists the devices that can be read from or written to another CPU module by using another CPU module access instructions.

Classification	Type	Device name*1	Target device setting	Condition
User device System device	Bit device	X, Y, M, L, B, F, SB, SM	Available	Satisfy the following conditions. • Four digits are specified by 16 bits. • The start bit device is a multiple of 16 (10H).
	Word device	T, ST, C, D, W, SW, SD	Available	None
File register	Word device	R, ZR	Available	None
Indirect specification		—	Not available	None

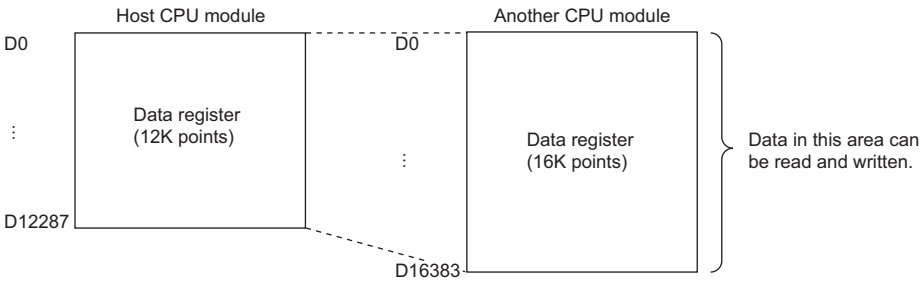
*1 Index modification (Z, ZZ representation) using the index register by the string specification can be performed. Another CPU module is accessed with a value which is index-modified by the value of the index register of the host CPU module.
For example, "K4M0Z0" with Z0=16 causes M0+16=M16, causing K4M16 to access another CPU module.
Similarly, "ZR0ZZ0" with Z0, Z1=100000 causes ZR0+10000=ZR100000, causing ZR100000 to access another CPU module.

Device specification method and readable/writable ranges

Specify the device of another CPU module with character strings.
Directly specify "D200", the write-target device number in another CPU module.



The string specification enables writing to or reading from every range of the device in another CPU module. For example, when the data register of the host CPU has 12K points while the data register of another CPU module has 16K points, 16K points of data can be written to or read from the head of the data register of another CPU module.



Point

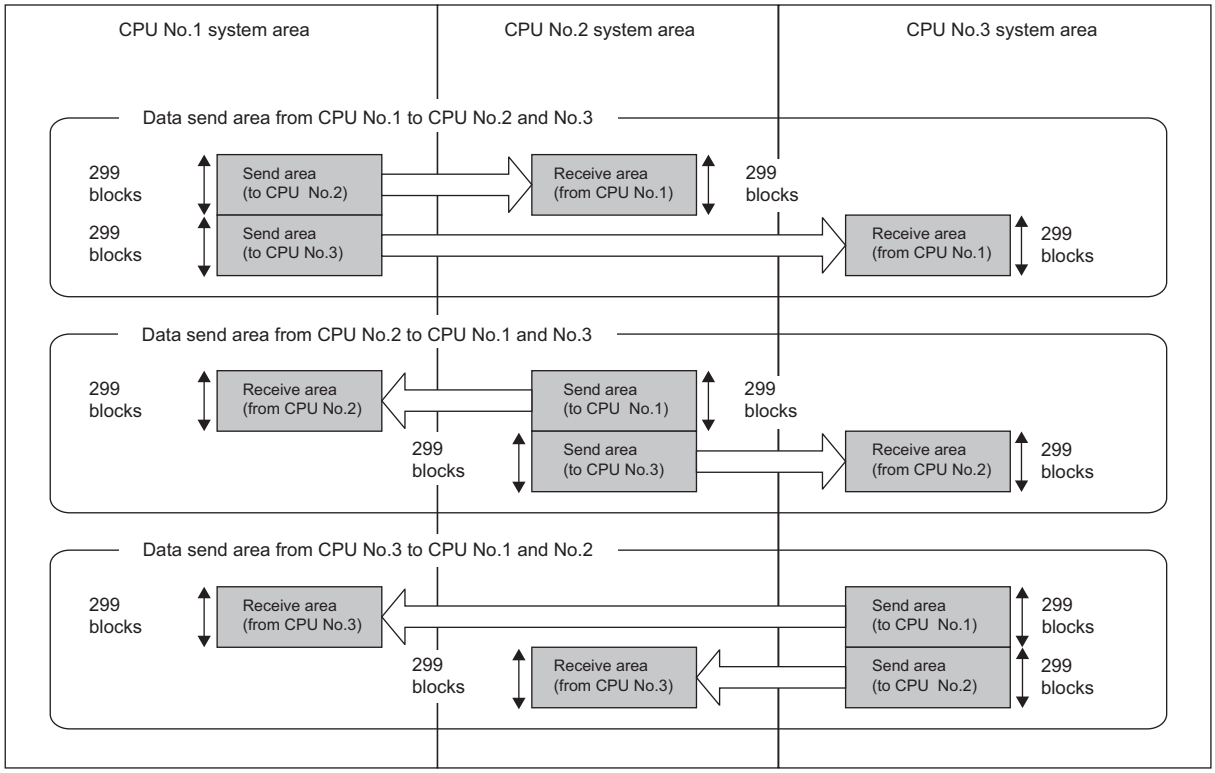
- Even if "0" is added to higher places of a device number, the device is processed the same as when it is not added. For example, "D1" and "D0001" are each processed as D1.
- Device numbers are not case-sensitive in terms of processing. For example, "D1" and "d1" are each processed as D1.
- Note that if a device not existing in another CPU module is specified by a character string, the instruction will be completed with an error.

Number of available blocks

Another CPU module access instructions use the system area in minimum units of blocks, each consisting of 16 words. The following table lists the numbers of blocks available for another CPU module access instructions.

Number of CPU modules	Maximum number of blocks
2	599
3	299
4	199

The following figure shows how blocks are used in a multiple CPU system consisting of three CPU modules.



Maximum number of data points that can be read or written

The maximum number of data points that can be read or written by an instruction depends on the number of CPU modules in a multiple CPU system configuration.

Number of CPU modules	Maximum number of data points that can be read	Maximum number of data points that can be written
2 modules	8192 point	8192 point
3 modules	4096 point	4096 point
4 modules	2048 point	2048 point

Number of blocks used by instructions

The number of blocks used by instructions depends on the number of read/write data points. The following table lists the numbers of blocks used by instructions.

Reading/ writing	Number of blocks	Example
Read	Number of blocks used by instructions = (21 + number of read data points)÷16	• When the number of read data points is 100 Number of blocks used by instructions = (21 + 100)÷16 = 7 [blocks]
Write	Number of blocks used by instructions = (19 + number of write data points)÷16	• When the number of write data points is 100 Number of blocks used by instructions = (19 + 100)÷16 = 7 [blocks]

Simultaneous execution of another CPU module access Instructions

Another CPU module access instructions can be executed simultaneously within the range of the following expression.

Conditions under which another CPU module access instructions can be executed simultaneously
[Number of blocks available for each CPU module]≥[total number of blocks used by concurrently executed instructions]

If executing another CPU module access instruction causes the number of blocks used by the CPU module access instructions to exceed the total number of blocks in the system area, the instruction is not executed (no processing) in the relevant scan and is executed in the next scan.
Note, however, that this instruction is completed with an error if the number of empty block in the system area is less than the value specified in SD796 to SD799 (maximum number of blocks for multiple CPU dedicated instructions) when the instruction is executed.
The table below shows whether another CPU module access instruction can be executed when the number of empty blocks in the system area is less than the number of blocks used by another CPU module access instructions or the value set in SD796 to SD799.

Size relationship between the value set in SD*3 and number of empty blocks*2	Size relationship between the number of blocks used by instruction*1 and number of empty blocks	
	Number of blocks used by instruction ≤ and number of empty blocks	Number of blocks used by instruction > number of empty blocks
Value set in SD ≤ number of empty blocks	Executed	Not executed (non-processing)
Value set in SD > number of empty blocks	Completed with an error	

*1 Number of blocks used by another CPU module access instructions
*2 Number of empty blocks in the system area
*3 Value set in SD796 to SD799

Interlock applied when another CPU module access instructions are used

Special relay SM796 to SM799 is used for interlocking among another CPU module access instructions.
When executing multiple another CPU module access instructions concurrently, use SM796 to SM799 for interlocking among these instructions.

Point

When using SM796 to SM799, specify the maximum numbers of blocks of the instructions used by individual CPU modules in SD796 to SD799. For example, when the maximum number of blocks used by another CPU module access instructions executed for CPU module No.3 is 5, specify 5 in SD798.
When the number of blocks specified in any of SD796 to SD799 is exceeded, the relevant special relay (SM796 to SM799) turns on.

Precautions

- Execute the D(P).DDWR, M(P).DDWR, D(P).DDRD, or M(P).DDRD instruction while the read/write target CPU module is on. If the instruction is executed while the target CPU is not on, the instruction performs no processing.
- After the D(P).DDWR, M(P).DDWR, D(P).DDRD, or M(P).DDRD instruction is executed, do not change the device range specified in the setting data before the completion device is turned on; otherwise, the completion status and completion device data can no longer be stored in the system.
- SB/SW and SM/SD include the system information area. When writing data with the D(P).DDWR or M(P).DDWR instruction, be careful not to overwrite the system information area.
- If the number of blocks used by the instruction to be executed is greater than the value set in SD796 to SD799, the instruction may not be executed (terminated abnormally) even if it is interlocked with SM796 to SM799.
- Set SD796 to SD799 before executing the instruction for the corresponding CPU module. (It is recommended to set them in the first scan after the CPU module runs.)

Reading device data from another CPU module

D(P).DDRD, M(P).DDRD

- D(P).DDRD

RnCPU | RnENCPU | RnPCPU (Process) | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

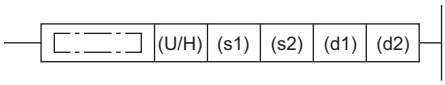
- The RnENCPU with firmware version "25" or later supports these instructions. Use an engineering tool with version "1.032J" or later.
- The RnSFCPU (standard) with firmware version "17" or later supports these instructions. Use an engineering tool with version "1.055H" or later.

- M(P).DDRD

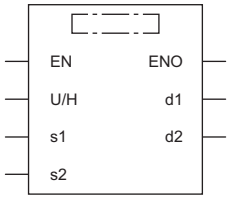
RnCPU | RnENCPU | RnPCPU (Process) | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

The RnENCPU with firmware version "25" or later supports these instructions. Use an engineering tool with version "1.032J" or later.

These instructions read device data from another CPU module in a multiple CPU system.

Ladder	ST
	ENO:=D_DDRD(EN,U/H,s1,s2,d1,d2); ENO:=DP_DDRD(EN,U/H,s1,s2,d1,d2); ENO:=M_DDRD(EN,U/H,s1,s2,d1,d2); ENO:=MP_DDRD(EN,U/H,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
D.DDRD M.DDRD	
DP.DDRD MP.DDRD	

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 another CPU module	3E0H to 3E3H	16-bit unsigned binary	ANY16
(s1)	Start device of host CPU module where the control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 2)
(s2)	Start device of another CPU module where the data to be read is stored	—	String	ANYSTRING_SINGLE
(d1)	Start device of host CPU module for storing the data that has been read	—	Word	ANY16*1
(d2)	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 (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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U/H)	○	—	○	—	—	—	—	○	○	—	—	○
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	—	—	—	—	—	—	—	—	○	—
(d1)	○	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—

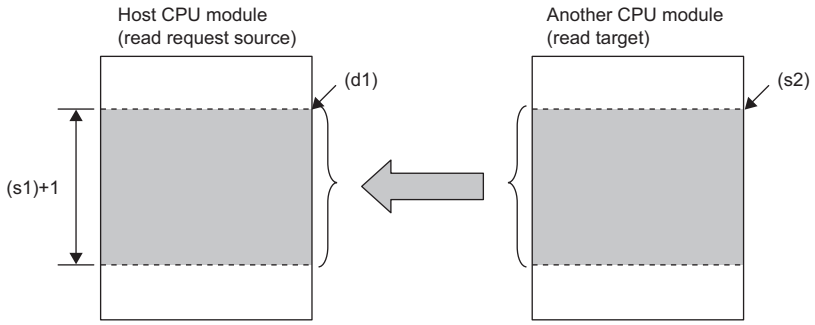
■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Completion status	The completion status is stored. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+1	Number of read data points	Specify the number of read data points in units of words.	1 to 8192*1	User

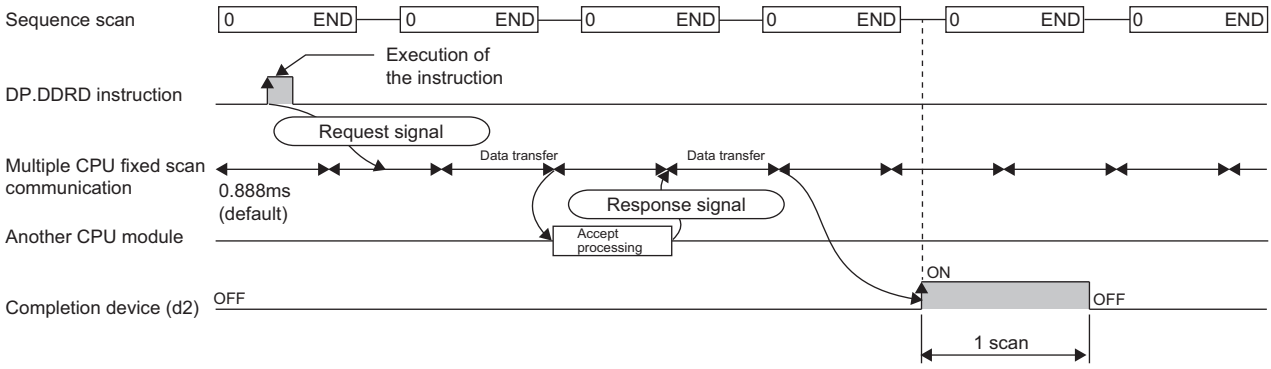
*1 This is the maximum setting range in a multiple CPU system consisting of two CPU modules.
It may be less than 8192 because the number of data points that can be read varies depending on the system configuration. (☞ Page 1281 Another CPU Module Access Instructions)

Processing details

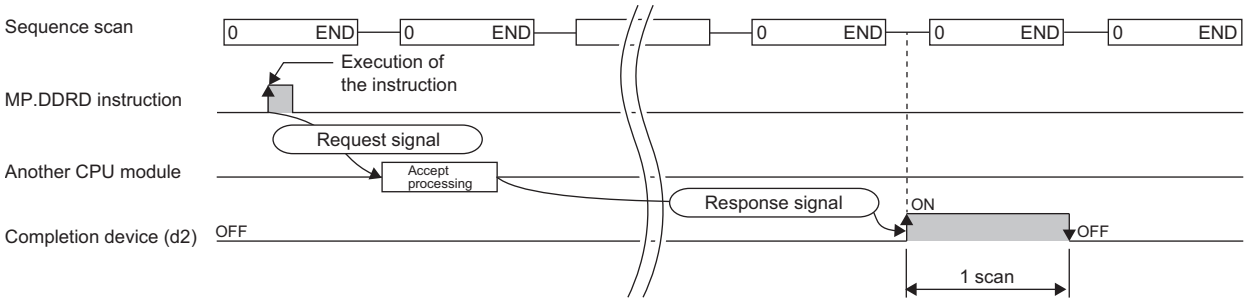
- In a multiple CPU system, these instructions read the data in the device specified by (d1) in the host CPU module, by the number of read data points specified by (s1)+1, and store it in the device specified by (d1) and later in another CPU module (U/H).



- The following figure shows an outline of operation of the D(P).DDRD and M(P).DDRD instructions.
  - Outline of operation of the D(P).DDRD instructions



• Outline of operation of the M(P).DDRD instructions



- The execution of the D(P).DDRD or M(P).DDRD instruction and whether it has been completed normally or with an error can be checked with the completion device (d2) or completion status indication device (d2)+1.

• Completion device (d2)

The completion device turns on in END processing of the scan performed upon completion of the D(P).DDRD or M(P).DDRD instruction and turns off in the next END processing.

• Completion status indication device (d2)+1

The completion device turns on or off depending on the completion status of the D(P).DDRD or M(P).DDRD instruction.

When completed normally: Unchanged from off.

When completed with an error: Turns on in END processing of the scan performed upon completion of the D(P).DDRD or M(P).DDRD instruction and turns off in the next END processing.

When completed with an error, an error code is stored in the device (completion status) specified by (s1)+0.

- The number of blocks used by instructions depends on the number of read data points. For the number of blocks used by instructions, refer to the following.

☞ Page 1281 Another CPU Module Access Instructions

- For the specifiable target devices in the read target CPU module, refer to the following.

☞ Page 1281 Another CPU Module Access Instructions

- If an instruction is executed while the system area has no empty block, it is completed with an error. Completion with an error can be prevented by setting the number of blocks used by instructions in SD796 to SD799 and using SM796 to SM799 as interlocks.

Operation error

Error code (SD0)	Description
2800H	The start I/O number (first three digits in four-digit hexadecimal representation) of the specified CPU module is out of the range, 3E0H to 3E3H.
2801H	An invalid another CPU module is specified. • A reserved CPU module is specified. • An unmounted CPU module is specified.
2802H	Another CPU module does not support the D(P).DDRD and M(P).DDRD instructions.
2803H	The host CPU module is specified as another CPU module.
2810H	A CPU module which cannot execute the instruction is specified as another CPU module.
3404H	An invalid character string is used to specify a device.
3405H	The number of read data points specified by (s1)+1 is out of the range from 0 to 8192.*2
3440H	The D(P).DDRD instruction is executed with the inter-CPU fixed-scan communication disabled.
3441H	The specified number of data points exceeds the size of the system area that can be used by each CPU module.

*2 This is the maximum setting range in a multiple CPU system consisting of two CPU modules.
It may be less than 8192 because the number of data points that can be read varies depending on the system configuration. (☞ Page 1281 Another CPU Module Access Instructions)

Error code ((s1)+0)	Description
0010H	The instruction request to the target CPU module exceeds the allowable value. (There is not empty block in the system area.)
1001H	The device of another CPU module specified by (s2) cannot be used by another CPU module. Alternatively, it is out of the device range.
1081H	The number of read data points that has been set by the D(P).DDRD or M(P).DDRD instruction is 0.

Writing device data to another CPU module

D(P).DDWR, M(P).DDWR

- D(P).DDWR

RnCPU | RnENCPU | RnPCPU (Process) | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

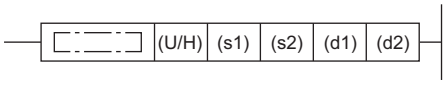
The RnENCPU with firmware version "25" or later supports these instructions. Use an engineering tool with version "1.032J" or later.

- M(P).DDWR

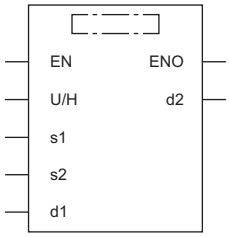
RnCPU | RnENCPU | RnPCPU (Process) | ~~RnPCPU (Redundant)~~ | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

The RnENCPU with firmware version "25" or later supports these instructions. Use an engineering tool with version "1.032J" or later.

These instructions write device data to another CPU module in a multiple CPU system.

Ladder	ST
	ENO:=D_DDWR(EN,U/H,s1,s2,d1,d2); ENO:=DP_DDWR(EN,U/H,s1,s2,d1,d2); ENO:=M_DDWR(EN,U/H,s1,s2,d1,d2); ENO:=MP_DDWR(EN,U/H,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
D.DDWR M.DDWR	
DP.DDWR MP.DDWR	

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 another CPU module	3E0H to 3E3H	16-bit unsigned binary	ANY16
(s1)	Start device of host CPU module where the control data is stored	Refer to the control data.	Word	ANY16_ARRAY (Number of elements: 2)
(s2)	Start device of host CPU module where the write data is stored	—	Word	ANY16*1
(d1)	Start device of another CPU module for storing the written data	—	String	ANYSTRING_SINGLE
(d2)	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 (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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(U/H)	○	—	○	—	—	—	—	○	○	—	—	○
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	○	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	—	—	—	—	—	—	—	—	○	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—

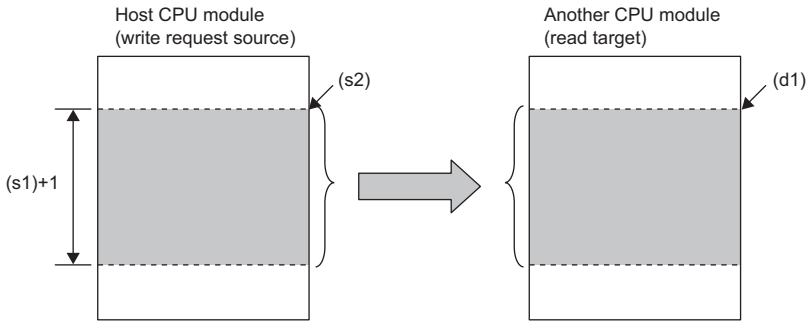
■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Completion status	The completion status is stored. • 0000H: Completed successfully • Other than 0000H: Completed with an error (error code)	—	System
+1	Number of write data points	Specify the number of write data points in units of words.	1 to 8192*1	User

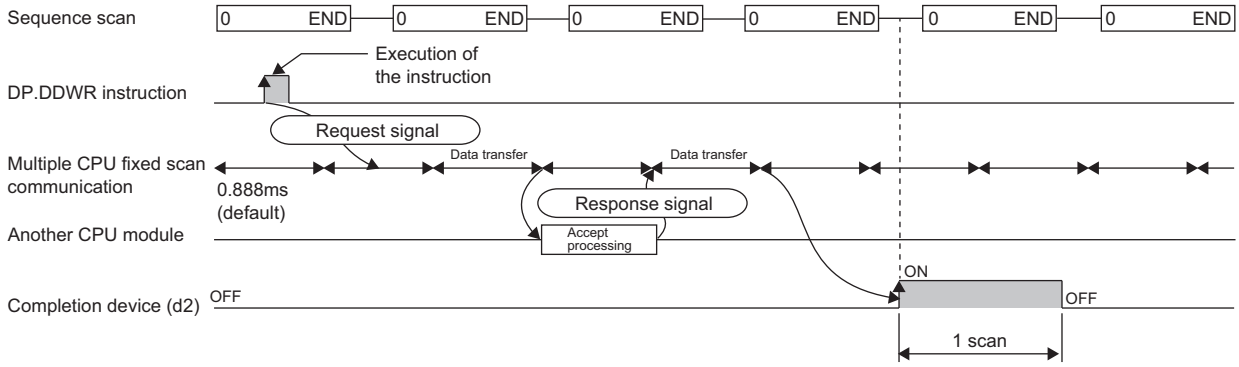
*1 This is the maximum setting range in a multiple CPU system consisting of two CPU modules.
It may be less than 8192 because the number of data points that can be written varies depending on the system configuration. (☞ Page 1281 Another CPU Module Access Instructions)

Processing details

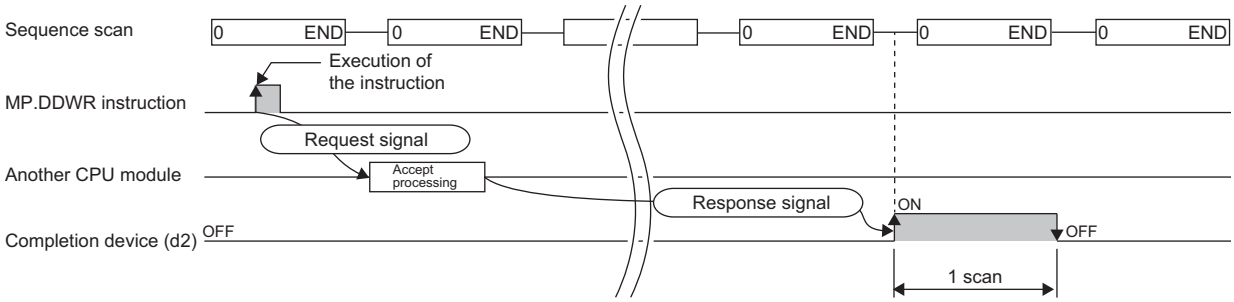
- In a multiple CPU system, these instructions read the data in the device specified by (s2) in the host CPU module, by the number of write data points specified by (s1)+1, and store it in the device specified by (d1) and later in another CPU module (U/H).



- The following figure shows an outline of operation of the D(P).DDWR and M(P).DDWR instructions.
  - Outline of operation of the D(P).DDWR instructions



• Outline of operation of the M(P).DDWR instructions



- The execution of the D(P).DDWR or M(P).DDWR instruction and whether it has been completed normally or with an error can be checked with the completion device (d2) or completion status indication device (d2)+1.

• Completion device (d2)

The completion device turns on in END processing of the scan performed upon completion of the D(P).DDWR or M(P).DDWR instruction and turns off in the next END processing.

• Completion status indication device (d2)+1

The completion device turns on or off depending on the completion status of the D(P).DDWR or M(P).DDWR instruction.

When completed normally: Unchanged from off.

When completed with an error: Turns on in END processing of the scan performed upon completion of the D(P).DDWR or M(P).DDWR instruction and turns off in the next END processing.

When completed with an error, an error code is stored in the device (completion status) specified by (s1)+0.

- The number of blocks used by instructions depends on the number of write data points. For the number of blocks used by instructions, refer to the following.

☞ Page 1281 Another CPU Module Access Instructions

- For the specifiable target devices in the write target CPU module, refer to the following.

☞ Page 1281 Another CPU Module Access Instructions

- If an instruction is executed while the system area has no empty block, it is completed with an error. Completion with an error can be prevented by setting the number of blocks used by instructions in SD796 to SD799 and using SM796 to SM799 as interlocks.

Operation error

Error code (SD0)	Description
2800H	The start I/O number (first three digits in four-digit hexadecimal representation) of the specified CPU module is out of the range, 3E0H to 3E3H.
2801H	An invalid another CPU module is specified. • A reserved CPU module is specified. • An unmounted CPU module is specified.
2802H	Another CPU module does not support the D(P).DDWR and M(P).DDWR instructions.
2803H	The host CPU module is specified as another CPU module.
2810H	A CPU module which cannot execute the instruction is specified as another CPU module.
3404H	An invalid character string is used to specify a device.
3405H	The number of write data points specified by (s1)+1 is out of the range from 0 to 8192.*2
3440H	The D(P).DDWR instruction is executed with the inter-CPU fixed-scan communication disabled.
3441H	The specified number of data points exceeds the size of the system area that can be used by each CPU module.

*2 This is the maximum setting range in a multiple CPU system consisting of two CPU modules.
It may be less than 8192 because the number of data points that can be written varies depending on the system configuration. (☞ Page 1281 Another CPU Module Access Instructions)

Error code ((s1)+0)	Description
0010H	The instruction request to the target CPU module exceeds the allowable value. (There is not empty block in the system area.)
1001H	The device of another CPU module specified by (d1) cannot be used by another CPU module. Alternatively, it is out of the device range.
1080H	The number of write data points that has been set by the D(P).DDWR or M(P).DDWR instruction is 0.
1090H	The destination device of the D(P).DDWR or M(P).DDWR instruction disables device data writing from outside the CPU module.

28 SFC PROGRAM INSTRUCTIONS

Point

This chapter describes the instructions used in SFC programs. For details on SFC programs, refer to the following.

MELSEC iQ-R Programming Manual (Program Design)

28.1 SFC Control Instructions

Checking the status of a step

LD, LDI, AND, ANI, OR, ORI [S□/BL□\S□]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "12" or later support these instructions. Use an engineering tool with version "1.015R" or later.
- The Process CPU (process mode) with firmware version "03" or later supports these instructions. Use an engineering tool with version "1.020W" or later.
- The Process CPU (redundant mode) with firmware version "18" or later supports these instructions. Use an engineering tool with version "1.050C" or later.

- LD: Normally open contact, LDI: Normally closed contact

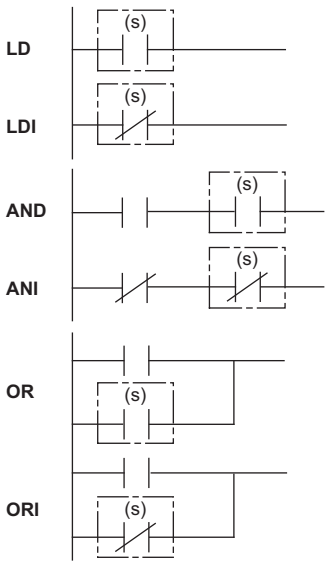
These instructions output the status (active or inactive) of the specified step as the operation result.

- AND: Normally open contact series connection, ANI: Normally closed contact series connection

These instructions perform an AND operation between the status (active or inactive) of the specified step and the previous operation result(s), 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 status (active or inactive) of the specified step and the previous operation result(s), 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 (BL□\S□)
	X, Y, M, L, SM, F, B, S, 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	$	
(s)	○*1	—	—	—	—	—	—	—	—	—	—	○

*1 Only S can be used.

Processing details

- These instructions check whether the specified step in the specified block is active or not.
- The status (on or off) of each contact will be as follows depending on the status (active or inactive) of the specified step.

Status of the specified step	Contact of the normally open contact instruction	Contact of the normally closed contact instruction
Active	On	Off
Inactive	Off	On

- The following table summarizes specification methods of steps.

Program		Specification method
SFC program	Specifying a step in current block	Use S□.
	Specifying a step in another block	Use BL□\S□
Sequence program		Use BL□\S□ When a block No. is not specified, specify a block No. by using the BRSET instruction. If a block No. is not specified by using the BRSET instruction, the target block will be block 0.

- Specify the block No. and the step No. within the following range.

CPU module	Block No.	Step No.
R00CPU, R01CPU, R02CPU	0 to 127	0 to 127
Other CPU modules	0 to 319	0 to 511

- If the block No. or the step No. specified is out of range, both of the normally open contact and normally closed contact turn off.
- Execute the instruction only when an SFC program exists (SM320 is on) and SM321 is on.
- If the instruction is executed while no SFC program exists (SM320 is off) or SM321 is off and both of the specified block No. and step No. are within the range, the normally open contact instruction turns off and the normally closed contact instruction turns on.

Precautions

- The status (on or off) of the contact of a step specified by S□, which is specified without a block No., on the sequence program, cannot be monitored on the engineering tool. However, the operation is performed. If the contact is on in the CPU module, coil output turns on.

Operation error

There is no operation error.

Checking the status of a block

LD, LDI, AND, ANI, OR, ORI [BL□]

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "12" or later support these instructions. Use an engineering tool with version "1.015R" or later.
- The Process CPU (process mode) with firmware version "03" or later supports these instructions. Use an engineering tool with version "1.020W" or later.
- The Process CPU (redundant mode) with firmware version "18" or later supports these instructions. Use an engineering tool with version "1.050C" or later.

- LD: Normally open contact, LDI: Normally closed contact

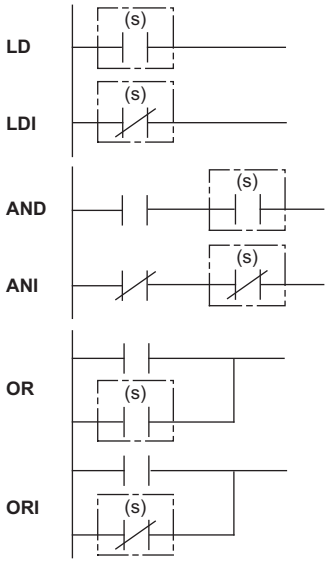
These instructions output the status (active or inactive) of the specified block as the operation result.

- AND: Normally open contact series connection, ANI: Normally closed contact series connection

These instructions perform an AND operation between the status (active or inactive) of the specified block and the previous operation result(s), 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 status (active or inactive) of the specified block and the previous operation result(s), 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 (BL□)
	X, Y, M, L, SM, F, B, S, 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	$	
(s)	—	—	—	—	—	—	—	—	—	—	—	○

Processing details

- These instructions check whether the specified block is active or not.
- The status (on or off) of each contact will be as follows depending on the status (active or inactive) of the specified block.

Status of the specified block	Contact of the normally open contact instruction	Contact of the normally closed contact instruction
Active	On	Off
Inactive	Off	On

- Specify the block No. within the following range.

CPU module	Block No.
R00CPU, R01CPU, R02CPU	0 to 127
Other CPU modules	0 to 319

- If the block No. specified is out of range, both of the normally open contact and normally closed contact turn off.
- Execute the instruction only when an SFC program exists (SM320 is on) and SM321 is on.
- If the instruction is executed while no SFC program exists (SM320 is off) or SM321 is off and the specified block No. is within the range, the normally open contact instruction turns off and the normally closed contact instruction turns on.

Operation error

There is no operation error.

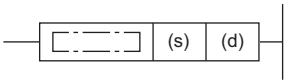
Batch-reading the status of steps

MOV(P) [K4S□/BL□\K4S□]

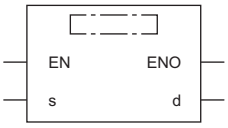
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "12" or later support these instructions. Use an engineering tool with version "1.015R" or later.
- The Process CPU (process mode) with firmware version "03" or later supports these instructions. Use an engineering tool with version "1.020W" or later.
- The Process CPU (redundant mode) with firmware version "18" or later supports these instructions. Use an engineering tool with version "1.050C" or later.

These instructions batch-read (in units of 16-bit binary data) the status (active or inactive) of steps in the specified block, and store the read data in the specified device.

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)	Device number where the transfer source data is stored	—	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

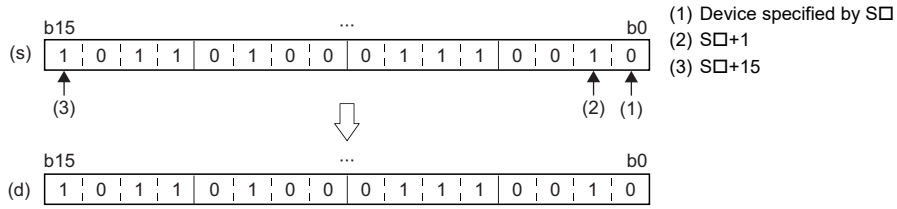
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□\S□)
	X, Y, M, L, SM, F, B, S, 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	$	
(s)	○*1	—	—	—	—	—	—	—	—	—	—	○
(d)	○	○	○	○	○	—	—	○	—	—	—	—

*1 Only S can be used.

Processing details

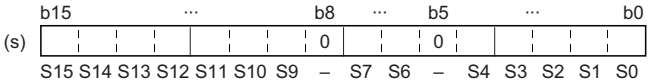
- These instructions batch-read (in units of 16-bit binary data) the status (active or inactive) of steps in the specified block.
- When a block is not specified, the status (active or inactive) of the following block is read.
  - Sequence program: Block 0
  - SFC program (within the action): Block where the instruction is executed (current block)
- The read data are stored in the device specified by (d). When the step is active, 1 is stored. When the step is inactive, 0 is stored.



- When there is a missing step No., 0 is stored in the corresponding bit.

Ex.

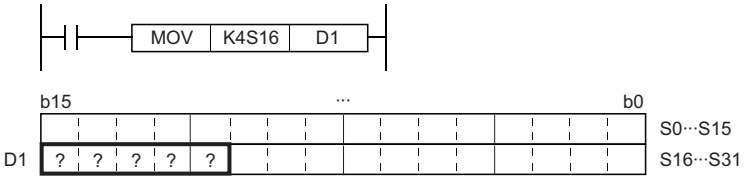
When the step No.5 and No.8 are missing in the specified block (The status of each step is stored in other bits.)



- If no block is specified and the read target range exceeds the maximum step No. in the block, undefined values are stored.

Ex.

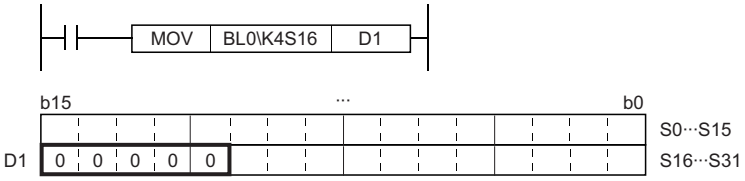
When the last step No. in the block is S26 and the status of steps (S16 to S31) are read to D1. (The status of each step is stored in other bits.)



- If the read target range exceeds the number of steps in the specified block, 0 is stored in the bits exceeding the existing step No.

Ex.

When the last step No. in the block is 26 and the status of steps (No.16 to No.31) are read to D1 (The status of each step is stored in other bits.)



- If the block No. that does not exist or does not include the read target data is specified, or if the specified block No. is correct but the non-existent step is specified, 0 is read and stored in all bits.
- If the instruction is executed while no SFC program exists, 0 is read and stored in all bits.

Operation error

Error code (SD0)	Description
2820H	When a block No. is specified, the specified block No. is out of the range.
	When a block No. is specified, the specified step No. is out of the range.

Point

Use a digit specification to specify a step.

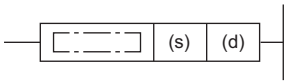
- To specify a step in the current block of an SFC program, use K4S□.
- To specify a step in another block of an SFC program, use BL□\K4S□.
- To specify a step of a sequence program, use BL□\K4S□.

DMOV(P) [K8S□/BL□\K8S□]

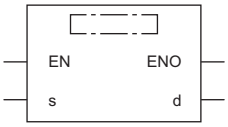
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "12" or later support these instructions. Use an engineering tool with version "1.015R" or later.
- The Process CPU (process mode) with firmware version "03" or later supports these instructions. Use an engineering tool with version "1.020W" or later.
- The Process CPU (redundant mode) with firmware version "18" or later supports these instructions. Use an engineering tool with version "1.050C" or later.

These instructions batch-read (in units of 32-bit binary data) the status (active or inactive) of steps in the specified block, and store the read data in the specified device.

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)	Device number where the transfer source data is stored	—	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

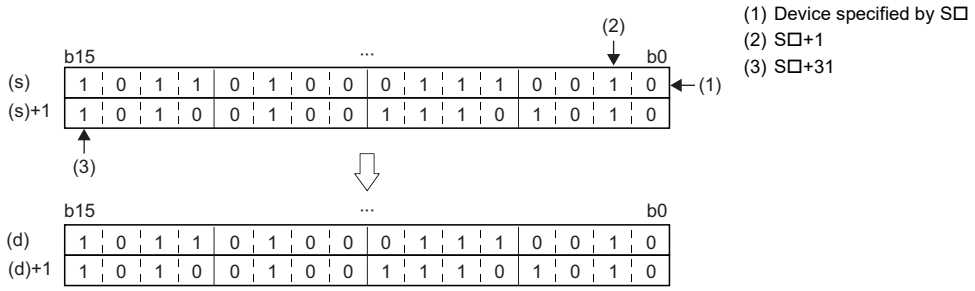
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□\S□)
	X, Y, M, L, SM, F, B, S, 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	$	
(s)	○*1	—	—	—	—	—	—	—	—	—	—	○
(d)	○	○	○	○	○	○	○	○	—	—	—	—

*1 Only S can be used.

Processing details

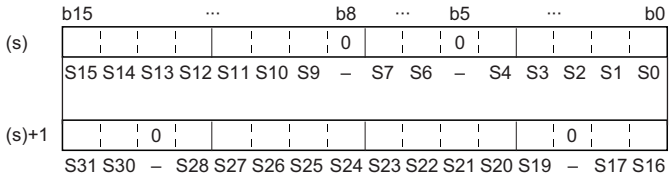
- These instructions batch-read (in units of 32-bit binary data) the status (active or inactive) of steps in the specified block.
- When a block is not specified, the status (active or inactive) of the following block is read.
  - Sequence program: Block 0
  - SFC program (within the action): Block where the instruction is executed (current block)
- The read data are stored in the device specified by (d). When the step is active, 1 is stored. When the step is inactive, 0 is stored.



- When there is a missing step No., 0 is stored in the corresponding bit.

Ex.

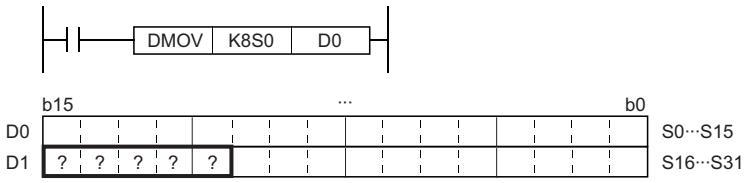
When the step No.5, 8, 18, and 29 are missing in the specified block (The status of each step is stored in other bits.)



- If no block is specified and the read target range exceeds the maximum step No. in the block, undefined values are stored.

Ex.

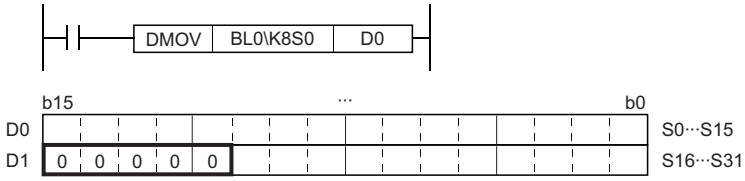
When the last step No. in the block is 26 and the status of the steps (No.0 to No.31) are read to D0 and D1 (The status of each step is stored in other bits.)



- If the read target range exceeds the number of steps in the specified block, 0 is stored in the bits exceeding the existing step No.

Ex.

When the last step No. in the block is 26 (The status of each step is stored in other bits.)



- If the block No. that does not exist or does not include the read target data is specified, or if the specified block No. is correct but the non-existent step is specified, 0 is read and stored in all bits.
- If the instruction is executed while no SFC program exists, 0 is read and stored in all bits.

Operation error

Error code (SD0)	Description
2820H	When a block No. is specified, the specified block No. is out of the range.
	When a block No. is specified, the specified step No. is out of the range.

Point

Use a digit specification to specify a step.

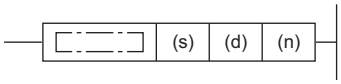
- To specify a step in the current block of an SFC program, use K8S□.
- To specify a step in another block of an SFC program, use BL□\K8S□.
- To specify a step of a sequence program, use BL□\K8S□.

BMOV(P) [K4S□/BL□\K4S□]

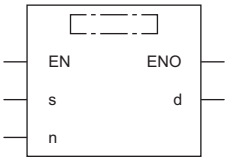
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "12" or later support these instructions. Use an engineering tool with version "1.015R" or later.
- The Process CPU (process mode) with firmware version "03" or later supports these instructions. Use an engineering tool with version "1.020W" or later.
- The Process CPU (redundant mode) with firmware version "18" or later supports these instructions. Use an engineering tool with version "1.050C" or later.

These instructions batch-read (in units of the specified number of words starting from the specified step) the status (active or inactive) of steps in the specified block.

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 number of a device where the transfer target data is stored	—	16-bit signed binary	ANY16
(d)	Transfer destination device start number	—	16-bit signed binary	ANY16
(n)	Number of transfer data points	0 to 65535	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

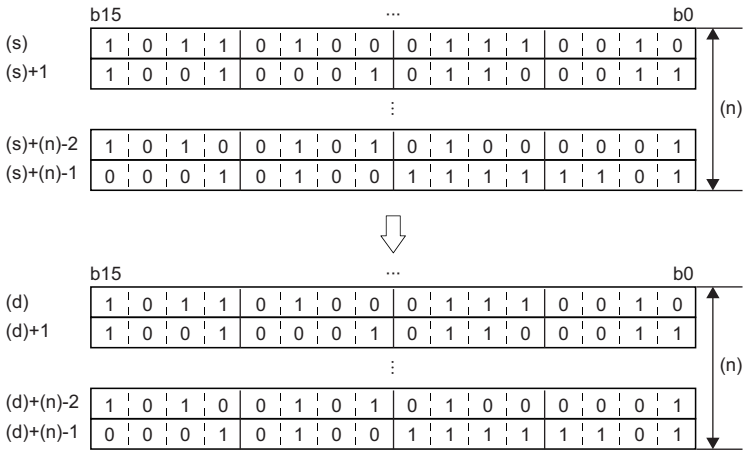
■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□\S□)
	X, Y, M, L, SM, F, B, S, 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	$	
(s)	○*1	—	—	—	—	—	—	—	—	—	—	○
(d)	○	○	○	○	—	○	—	○	—	—	—	—
(n)	○	○	○	○	○	—	—	○	○	—	—	—

*1 Only S can be used.

Processing details

- These instructions batch-read (in units of the specified number of words starting from the specified step) the status (active or inactive) of steps in the specified block.
- When a block is not specified, the status (active or inactive) of the following block is read.
  - Sequence program: Block 0
  - SFC program (within the action): Block where the instruction is executed (current block)
- The read data are stored in the device specified by (d).



- When there is a missing step No., 0 is stored in the corresponding bit.

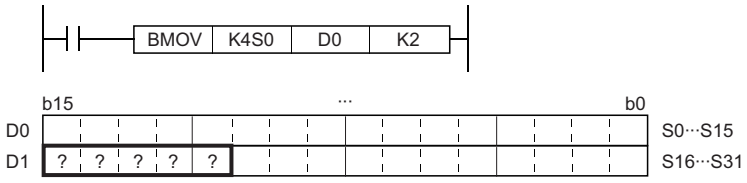
Ex.

When the step No.5 and No.8 are missing in the specified block (The status of each step is stored in other bits.)

- If no block is specified and the read target range exceeds the maximum step No. in the block, undefined values are stored.

Ex.

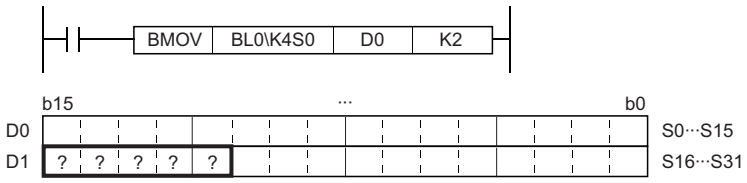
When the last step No. in the block is 26 and the status of the steps (two words from step No.0) is read to D0 and D1 (The status of each step is stored in other bits.)



- If the read target range exceeds the number of steps in the specified block, or if the non-existent step is specified as a start step, undefined values are stored.

Ex.

When the last step No. in the block is 26 and the status of the steps (two words from step No.0) is read to D0 and D1 (The status of each step is stored in other bits.)



- If the instruction is executed while no SFC program exists, or if the block No. that does not exist or does not include the read target data is specified, 0 is read and stored in all bits.

Operation error

Error code (SD0)	Description
2820H	When a block No. is specified, the specified block No. is out of the range.
	When a block No. is specified, the specified step No. is out of the range.

Point

Use a digit specification to specify a step.

- To specify a step in the current block of an SFC program, use K4S□.
- To specify a step in another block of an SFC program, use BL□\K4S□.
- To specify a step of a sequence program, use BL□\K4S□.

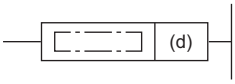
Starting a block

SET [BL□]

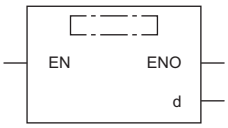
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "12" or later support this instruction. Use an engineering tool with version "1.015R" or later.
- The Process CPU (process mode) with firmware version "03" or later supports this instruction. Use an engineering tool with version "1.020W" or later.
- The Process CPU (redundant mode) with firmware version "18" or later supports this instruction. Use an engineering tool with version "1.050C" or later.

This instruction activates the specified block, and executes a step sequence starting from an initial step.

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)	Block No. to be activated (Set (on) target bit 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 (BL□)
	X, Y, M, L, SM, F, B, S, 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	$	
(d)	—	—	—	—	—	—	—	—	—	—	—	○*1

*1 The BL can be index modified.

Processing details

- This instruction activates the specified block, and executes a step sequence starting from an initial step. When there are several initial steps, all the initial steps are activated.
- If the block start/end bit of the SFC information device is set, the corresponding bit device turns on.
- If the instruction is executed to an active block, the instruction is ignored and processing will continue.
- If the instruction is executed to an SFC block for which the online program change is being executed, the instruction is ignored and processing will continue.
- For the Process CPU (redundant mode), executing the instruction in the standby system in backup mode will result in non-processing.

Operation error

Error code (SD0)	Description
31A1H	The specified block does not exist.
	The instruction is executed when no SFC program (scan execution type) exists or SM321 is off.
31A2H	The specified block No. is out of the range.

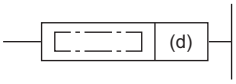
Ending a block

RST [BL□]

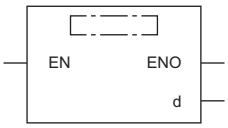
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "12" or later support this instruction. Use an engineering tool with version "1.015R" or later.
- The Process CPU (process mode) with firmware version "03" or later supports this instruction. Use an engineering tool with version "1.020W" or later.
- The Process CPU (redundant mode) with firmware version "18" or later supports this instruction. Use an engineering tool with version "1.050C" or later.

This instruction deactivates the specified block.

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)	Block No. to be deactivated (Reset (off) target bit device number)	—	Bit	ANY_ELEMENTARY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□)
	X, Y, M, L, SM, F, B, S, 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	$	
(d)	—	—	—	—	—	—	—	—	—	—	—	○*1

*1 The BL can be index modified.

Processing details

- This instruction deactivates the specified block independently.
- All the active steps are deactivated and coil outputs are turned off.
- If the block start/end bit of the SFC information device is set, the corresponding bit device turns off.
- If the instruction is executed to an inactive block, the instruction is ignored.
- For the Process CPU (redundant mode), executing the instruction in the standby system in backup mode will result in non-processing.

Operation error

Error code (SD0)	Description
31A1H	The specified block does not exist.
	The instruction is executed when no SFC program (scan execution type) exists or SM321 is off.
31A2H	The specified block No. is out of the range.

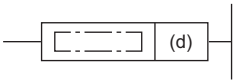
Pausing a block

PAUSE [BL□]

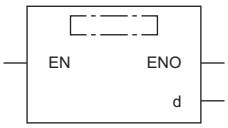
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "12" or later support this instruction. Use an engineering tool with version "1.015R" or later.
- The Process CPU (process mode) with firmware version "03" or later supports this instruction. Use an engineering tool with version "1.020W" or later.
- The Process CPU (redundant mode) with firmware version "18" or later supports this instruction. Use an engineering tool with version "1.050C" or later.

This instruction temporarily stops the step sequence in the specified block.

Ladder	ST
	ENO:=PAUSE(EN,d);

FBD/LD



■Execution condition

Instruction	Execution condition
PAUSE	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Block No. where the sequence is temporarily stopped	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□)
	X, Y, M, L, SM, F, B, S, 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	$	
(d)	—	—	—	—	—	—	—	—	—	—	—	○*1

*1 The BL can be index modified.

Processing details

- This instruction temporarily stops the step sequence in the specified block.
- If the block pause/restart bit of the SFC information device is set, the corresponding bit device turns on.
- Stop timing of each step after execution of the instruction differs depending on the setting (immediate stop or stop after transition) of the block stop mode bit of the SFC information device.

Setting of the block stop mode bit	Stop timing
Immediate stop	All the steps become inactive immediately after execution of the instruction.
Stop after transition	Steps continue operation even after the execution. After transition condition is satisfied, the operation stops at next step.

- The coil output status by using the OUT instruction at the temporary stop depends on the status of SM325.

Status of SM325	Coil output status by using the OUT instruction
On	The coil output remains the state immediately before stop.
Off	The coil output turns off at stop.

- For the Process CPU (redundant mode), executing the instruction in the standby system in backup mode will result in non-processing.

Precautions

If the sequence is stopped while SM325 is off, coil HOLD steps become inactive. The sequence cannot be restarted with the hold status. If the sequence is stopped while SM325 is on, the sequence can be restarted with the hold status.

Operation error

Error code (SD0)	Description
31A1H	The specified block does not exist.
	The instruction is executed when no SFC program (scan execution type) exists or SM321 is off.
31A2H	The specified block No. is out of the range.

Point

Operation of the PAUSE and RESTART instructions depends on the combination of the SM325 status, block stop mode bit setting of SFC information device, and step hold status. For details, refer to the following.
MELSEC iQ-R Programming Manual (Program Design)

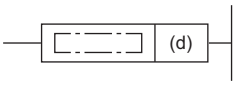
Restarting a block

RSTART [BL□]

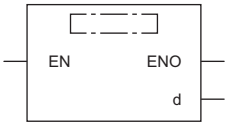
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "12" or later support this instruction. Use an engineering tool with version "1.015R" or later.
- The Process CPU (process mode) with firmware version "03" or later supports this instruction. Use an engineering tool with version "1.020W" or later.
- The Process CPU (redundant mode) with firmware version "18" or later supports this instruction. Use an engineering tool with version "1.050C" or later.

This instruction releases the temporary stop, and restarts the sequence from the step where the sequence was stopped in the specified block.

Ladder	ST
	ENO:=RSTART(EN, d);

FBD/LD



■Execution condition

Instruction	Execution condition
RSTART	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(d)	Block No. where the temporary stop is released	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□)
	X, Y, M, L, SM, F, B, S, 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	$	
(d)	—	—	—	—	—	—	—	—	—	—	—	○*1

*1 The BL can be index modified.

Processing details

- This instruction restarts the sequence from the step where the sequence was stopped in the specified block. Operation HOLD steps (with or without transition check), which have been stopped holding the operation status, restart operation with the hold status.
- If the block pause/restart bit of the SFC information device is set, the corresponding bit device turns off.
- After the step sequence is restarted, the operation of the PLS instruction and the instructions which is executed only on the rising edge depends on the status of SM325.

Status of SM325	Operation of the PLS instruction and the instructions executed on the rising edge
On (Coil output is held.)	The instruction is not executed.
Off (Coil output is off.)	The instruction is executed again.

- For the Process CPU (redundant mode), executing the instruction in the standby system in backup mode will result in non-processing.

Precautions

If the sequence is stopped while SM325 is off, coil HOLD steps become inactive. The sequence cannot be restarted with the hold status. If the sequence is stopped while SM325 is on, the sequence can be restarted with the hold status.

Operation error

Error code (SD0)	Description
31A1H	The specified block does not exist.
	The instruction is executed when no SFC program (scan execution type) exists or SM321 is off.
31A2H	The specified block No. is out of the range.

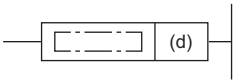
Activating a step

SET [S□/BL□\S□]

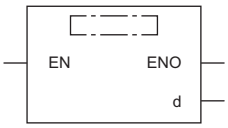
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "12" or later support this instruction. Use an engineering tool with version "1.015R" or later.
- The Process CPU (process mode) with firmware version "03" or later supports this instruction. Use an engineering tool with version "1.020W" or later.
- The Process CPU (redundant mode) with firmware version "18" or later supports this instruction. Use an engineering tool with version "1.050C" or later.

This instruction activates the specified step.

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)	Step No. to be activated (Set (on) target bit 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 (BL□\S□)
	X, Y, M, L, SM, F, B, S, 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	$	
(d)	○*1	—	—	—	—	—	—	—	—	—	—	○*1

*1 Only S, BL□\S□ can be used. The devices can be index modified.

Processing details

- This instruction activates the specified step in the specified block. The operation of the specified block will be as follows depending on the status (active or inactive) of the specified block.

Status of the specified block	Operation
Inactive	The specified block is activated at execution of the instruction, and starts the processing from the specified step. If the block start/end bit of the SFC information device is set, the corresponding bit device turns on.
Active	The specified step is newly activated while the step which has already been active continues processing.

- If the instruction is executed to an active step, the instruction is ignored and processing will continue. Note that if the specified step is holding the operating status, the hold status is reset. The step becomes a normal step, and executes the action and transition.
- If the instruction is executed to a step in an SFC block for which the online program change is being executed, the instruction is ignored and processing will continue.
- When there are several initial steps, any of them can be selected and activated.
- If no block is specified, the following block is targeted depending on the execution program type.
  - Sequence program: Block 0
  - SFC program (within the action): Block where the instruction is executed (current block)
- For the Process CPU (redundant mode), executing the instruction in the standby system in backup mode will result in non-processing.

Precautions

- In a simultaneous sequence, specify and activate all the steps by using the SET instruction (Activating a step). If there is any inactive step left, a convergence is not performed. Likewise, if the RST instruction (Deactivating a step) is executed to a single step in a simultaneous sequence, a convergence condition is not satisfied.
- The current step No. cannot be specified within the action of the SFC program. If specified, an error occurs.

Operation error

Error code (SD0)	Description
31A1H	The specified block does not exist. (SET/RST BL□\S□)
31A2H	When a block No. is specified, the specified block No. is out of the range.
31B1H	The instruction is executed when no SFC program (scan execution type) exists or SM321 is off.
	No specified step exists in the specified block. (SET/RST BL□\S□/S□)
31B2H	The specified step No. is out of the range.
31B5H	The current step is specified within the action.

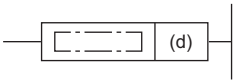
Deactivating a step

RST [S□/BL□\S□]

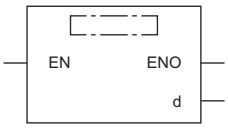
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "12" or later support this instruction. Use an engineering tool with version "1.015R" or later.
- The Process CPU (process mode) with firmware version "03" or later supports this instruction. Use an engineering tool with version "1.020W" or later.
- The Process CPU (redundant mode) with firmware version "18" or later supports this instruction. Use an engineering tool with version "1.050C" or later.

This instruction deactivates the specified step.

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)	Step No. to be deactivated (Reset (off) target bit device number)	—	Bit	ANY_ELEMENTARY
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others (BL□\S□)
	X, Y, M, L, SM, F, B, S, 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	$	
(d)	○*1	—	—	—	—	—	—	—	—	—	—	○*1

*1 Only S, BL□\S□ can be used. The devices can be index modified.

Processing details

- This instruction deactivates the specified step in the specified block.
- When the number of active steps in the specified becomes 0, the specified block will be inactive. If the block start/end bit of the SFC information device is set, the corresponding bit device turns off.
- If the instruction is executed to an inactive step, the instruction is ignored.
- If no block is specified, the following block is targeted depending on the execution program type.
  - Sequence program: Block 0
  - SFC program (within the action): Block where the instruction is executed (current block)
- For the Process CPU (redundant mode), executing the instruction in the standby system in backup mode will result in non-processing.

Precautions

- In a simultaneous sequence, specify and activate all the steps by using the SET instruction (Activating a step). If there is any inactive step left, a convergence is not performed. Likewise, if the RST instruction (Deactivating a step) is executed to a single step in a simultaneous sequence, a convergence condition is not satisfied.
- The current step No. cannot be specified within the action of the SFC program. If specified, an error occurs.

Operation error

Error code (SD0)	Description
31A1H	The specified block does not exist. (SET/RST BL□\S□)
31A2H	When a block No. is specified, the specified block No. is out of the range.
31B1H	The instruction is executed when no SFC program (scan execution type) exists or SM321 is off.
	No specified step exists in the specified block. (SET/RST BL□\S□/S□)
31B2H	The specified step No. is out of the range.
31B5H	The current step is specified within the action.

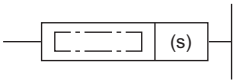
Switching a target block

BRSET

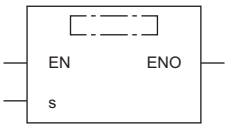
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "12" or later support this instruction. Use an engineering tool with version "1.015R" or later.
- The Process CPU (process mode) with firmware version "03" or later supports this instruction. Use an engineering tool with version "1.020W" or later.
- The Process CPU (redundant mode) with firmware version "18" or later supports this instruction. Use an engineering tool with version "1.050C" or later.

This instruction specifies an SFC control instruction target block No.

Ladder	ST
	ENO:=BRSET(EN,s);

FBD/LD



■Execution condition

Instruction	Execution condition
BRSET	

Setting data

■Description, range, data type

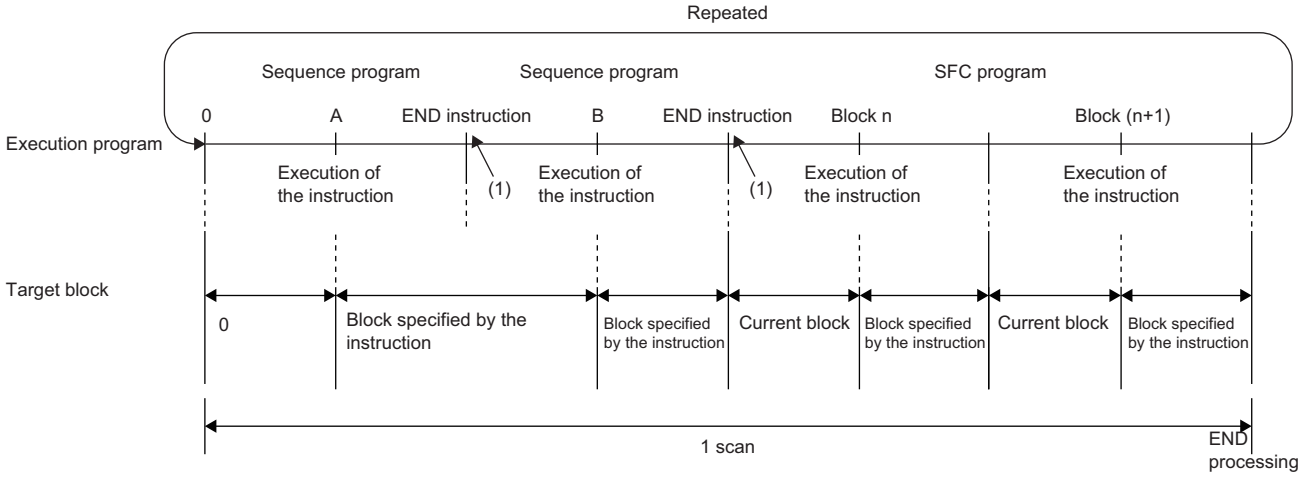
Operand	Description	Range	Data type	Data type (label)
(s)	Target block No.	—	16-bit signed binary	ANY16
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

■Applicable devices

Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, S, 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	$	
(s)	○	○	○	○	○	—	—	○	○	—	—	—

Processing details

- This instruction switches the SFC control instruction target block No. of the step (specified by S□) to the block No. specified by (s).
- The effective range of the instruction is as follows depending on the execution program type.
  - Sequence program: The block is switched from the point where the instruction is executed to the start of the SFC program. In the next scan, the target block will be the block 0 (default) until the instruction is executed again.
  - SFC program: The block is switched only within the step being executed. Even when the same step is targeted, the instruction must be executed for each block where the instruction specified by S□ is used. Moreover, even within a single step, the block is switched only from the point where the instruction is executed to the END processing of the step. In the next program, the target block will be the current block (default) until the instruction is executed again.



(1) END processing is not performed.

- If the block No. is specified by BL□\S□, the block No. is switched with or without execution of the instruction.
- The instruction is valid only for the target step. If multiple steps are active in such as a simultaneous sequence, the instruction needs to be executed for each step.

Operation error

Error code (SD0)	Description
31A1H	The specified block does not exist, or the SFC program is in a standby state.
	The instruction is executed when no SFC program (scan execution type) exists or SM321 is off.
31A2H	The specified block No. is out of the range.

28.2 SFC Dedicated Instruction

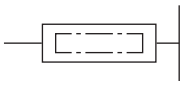
Creating a dummy transition condition

TRAN

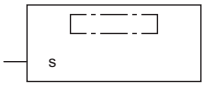
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | ~~RnPSFCPU (Standard)~~ | ~~RnPSFCPU (Safety)~~ | ~~RnSFCPU (Standard)~~ | ~~RnSFCPU (Safety)~~

- For the R00CPU, R01CPU, and R02CPU, there are no restrictions on the version.
- The RnCPU (excluding the R00CPU, R01CPU, and R02CPU) and RnENCPU with firmware version "12" or later support this instruction. Use an engineering tool with version "1.015R" or later.
- The Process CPU (process mode) with firmware version "03" or later supports this instruction. Use an engineering tool with version "1.020W" or later.
- The Process CPU (redundant mode) with firmware version "18" or later supports this instruction. Use an engineering tool with version "1.050C" or later.

This instruction is a dummy output which satisfies a transition condition.

Ladder	ST
	TRAN(s);

FBD/LD



Point

For details on transitions, refer to the following.

MELSEC iQ-R Programming Manual (Program Design)

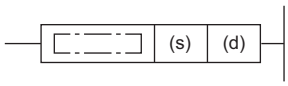
29 REDUNDANT SYSTEM INSTRUCTIONS

29.1 System Switching

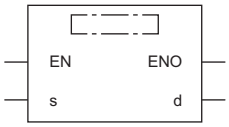
SP.CONTSW

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

This instruction switches the systems (control system and standby system) during END processing of the scan where the instruction is executed.

Ladder	ST
	ENO:=SP_CONTSW(EN,s,d);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.CONTSW	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Instruction ID number: A value used to identify a system switching request	-32768 to 32767	16-bit signed binary	ANY16
(d)	Error completion device: The device turns on when the system switching operation failed.	—	Bit	ANY_BOOL
EN	Execution condition	—	Bit	BOOL
ENO	Execution result	—	Bit	BOOL

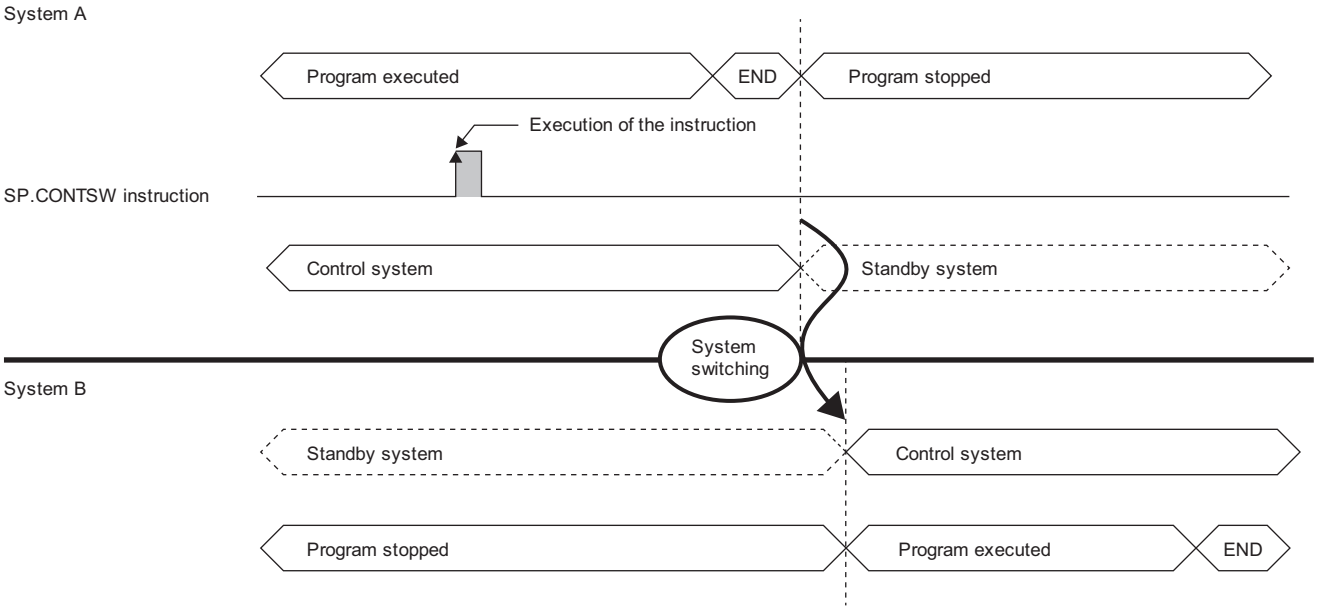
■Applicable devices

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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	○	—	○	—	—	—	—	○	○	—	—	—
(d)	○	—	○	—	—	—	—	○	—	—	—	—

Processing details

- This instruction switches the systems (control system and standby system) during END processing of the scan where the instruction is executed. The instruction must be executed in the control system. If it is executed in the standby system, no processing is performed.

The following figure shows the system switching operation by using the instruction.

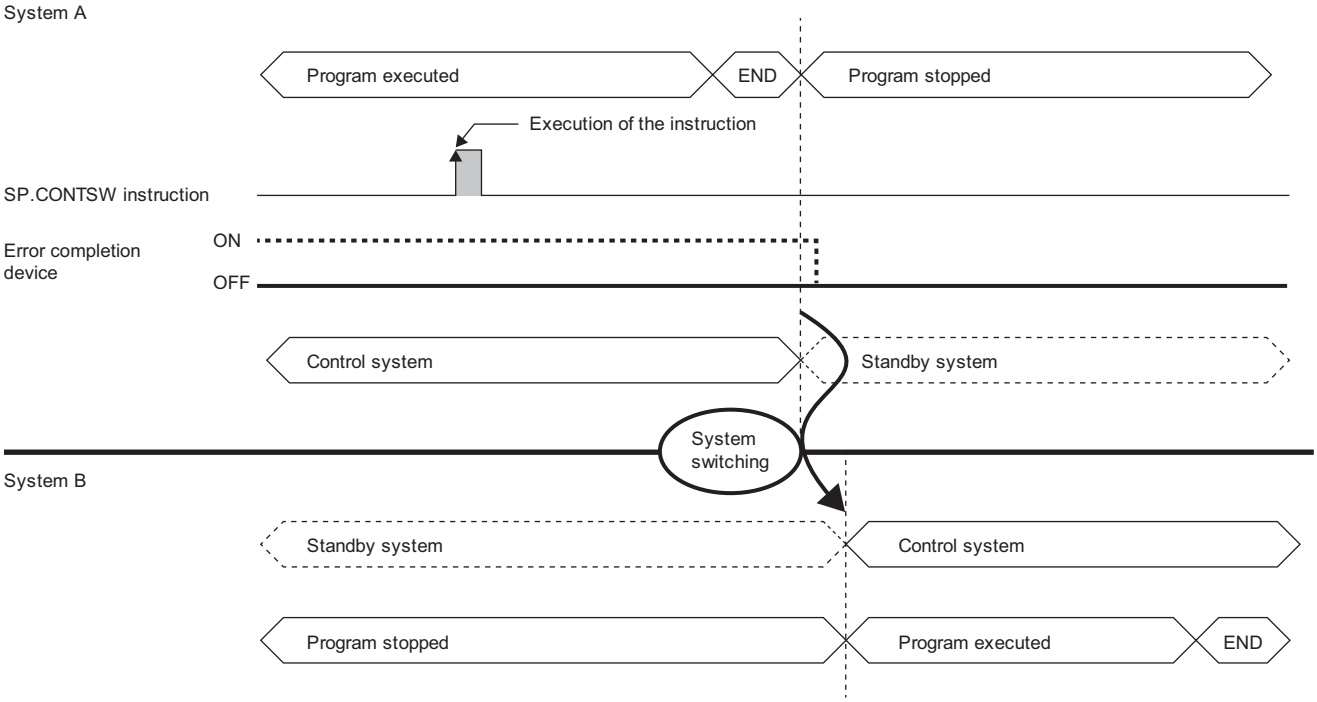


- To switch the systems by using the instruction, turn on SM1646 (System switching by a user) in advance. If the system switching operation is disabled by using the DCONTSW instruction, execute the ECONTSW instruction in the standby system before executing the SP.CONTSW instruction.
- The value specified by (s) is stored in SD1650 (System switching instruction ID number) of both CPU modules when the systems are switched successfully. When multiple SP.CONTSW instructions are executed in a program, the instruction used can be identified by reading data stored in SD1650. If two or more SP.CONTSW instructions are executed in one scan, only the data (argument) of the first instruction is stored.

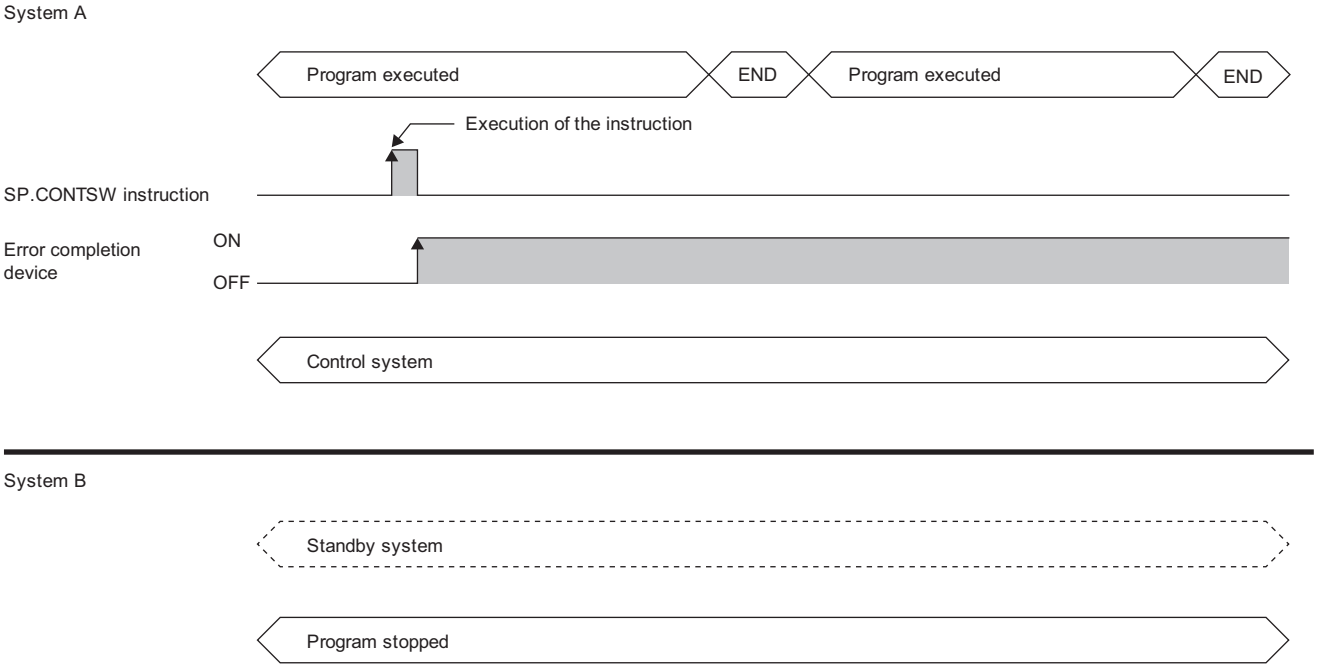
- The error completion device specified by (d) turns off when the systems were switched successfully, and turns on when the system switching operation failed. When there is a switching disable cause, 16 (System switching request by using the SP.CONTSW instruction) is stored in SD1643 of the CPU module in the control system. The corresponding disable cause number is stored in SD1644. For the disable cause number stored in SD1644, refer to the following.

MELSEC iQ-R CPU Module User's Manual (Application)

- When the systems are switched by using the instruction



- When SM1646 is off at execution of the instruction



Precautions

- To execute the instruction, turn on (enabled) SM1646. If SM1646 is off (disabled), the systems cannot be switched.
- Even if SM1646 is turned off after execution of the instruction and before the END processing that performs the system switching operation, the systems can be switched.

Operation error

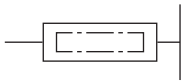
Error code (SD0)	Description
1BD0H	There is a switching disable cause and the systems are not switched during END processing.
1BD1H	When the instruction is executed, SM1646 is off (disabled).

29.2 Disabling/Enabling System Switching

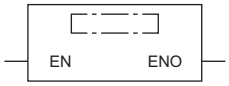
DCONTSW, ECONTSW

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- DCONTSW: This instruction disables manual system switching.
- ECONTSW: This instruction enables manual system switching.

Ladder	ST
	ENO:=DCONTSW(EN); ENO:=ECONTSW(EN);

FBD/LD



■Execution condition

Instruction	Execution condition
DCONTSW	
ECONTSW	

Processing details

- When the DCONTSW instruction is executed in the standby system, the system switching operation in the control system is disabled.
- When the ECONTSW instruction is executed in the standby system, the system switching operation in the control system is enabled.
- The system switching operation in the control system is enabled by default.
- These instructions can only be executed in the standby system. Even if these instructions are executed in the control system, no processing is performed.

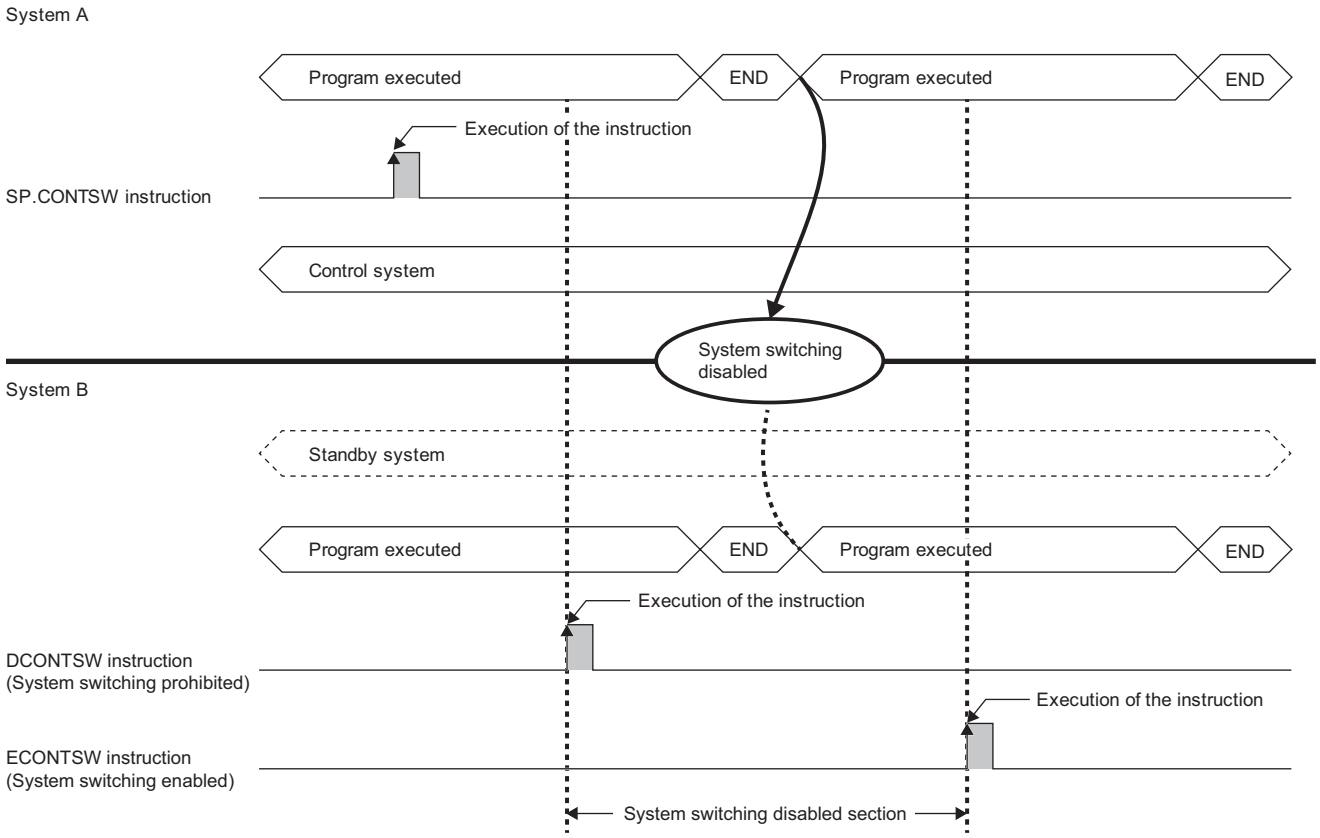
Point

To execute these instructions in the standby system in backup mode, write the instructions in a program executed in both systems or in a POU called by a program executed in both systems. For details on executing a program in both systems, refer to the following.

MELSEC iQ-R CPU Module User's Manual (Application)

Ex.

When the SP.CONTSW instruction is executed while the system switching operation is disabled by the DCONTSW instruction



- The following operations enables system switching. To disable system switching, execute the DCONTSW instruction again.
  - Powering off and on the standby system
  - Resetting the CPU module in the standby system
  - Changing the operating status of the CPU module in the standby system from RUN to STOP
  - A stop error in the CPU module of the standby system
  - Switching the operation mode from backup mode to separate mode
  - Automatic system switching

Operation error

There is no operation error.

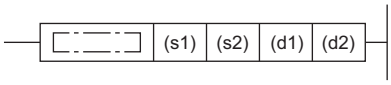
29.3 Writing Data from the Standby System to the Control System

CONTWR(P)

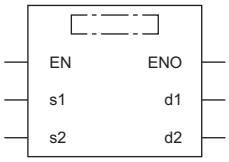
RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | RnPSFCPU (Safety) | RnSFCPU (Standard) | RnSFCPU (Safety)

- The Process CPU (redundant mode) with firmware version "18" or later supports these instructions. Use an engineering tool with version "1.050C" or later.

These instructions write data from the standby system to the control system in a program executed in both systems.

Ladder	ST
	ENO:=CONTWR(EN,s1,s2,d1,d2); ENO:=CONTWRP(EN,s1,s2,d1,d2);

FBD/LD



■Execution condition

Instruction	Execution condition
CONTWR	
CONTWRP	

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s1)	Start device where control data is stored	—	Word	ANY16_ARRAY (Number of elements: 2)
(s2)	Start device for storing the data to be written	—	Word	ANY16
(d1)	Start device of write destination (control system)	—	Word	ANY16
(d2)	Device that turns on for one scan upon completion of the instruction When the instruction is complete 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

■Applicable devices

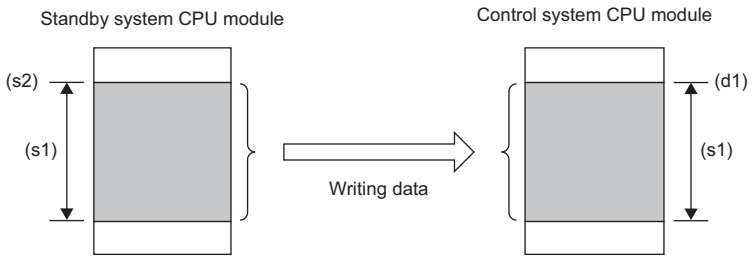
Operand	Bit		Word			Double word		Indirect specification	Constant			Others
	X, Y, M, L, SM, F, B, SB, FX, FY	J□\□	T, ST, C, D, W, SD, SW, FD, R, ZR, RD	U□\G□, J□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s1)	—	—	○	—	—	—	—	○	—	—	—	—
(s2)	—	—	○	—	—	—	—	○	—	—	—	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	○	—	○	—	—	—	—	○	—	—	—	—

■Control data

Operand: (s1)				
Device	**Item**	**Description**	**Setting range**	**Set by**
+0	Number of write data points	Specifies the number of write data points in units of words.	1 to 1024	User
+1	Completion status	Stores the instruction completion status. • 0: Completed successfully • Other than 0: Completed with an error (data write disabling cause)	—	System

Processing details

- This instruction writes the data for the number of write data points specified by (s1) from the device/label of the standby system specified by (s2) to the device/label of the control system specified by (d1). Upon completion of writing data to the control system, the completion device of the standby system specified by (d2) turns on.

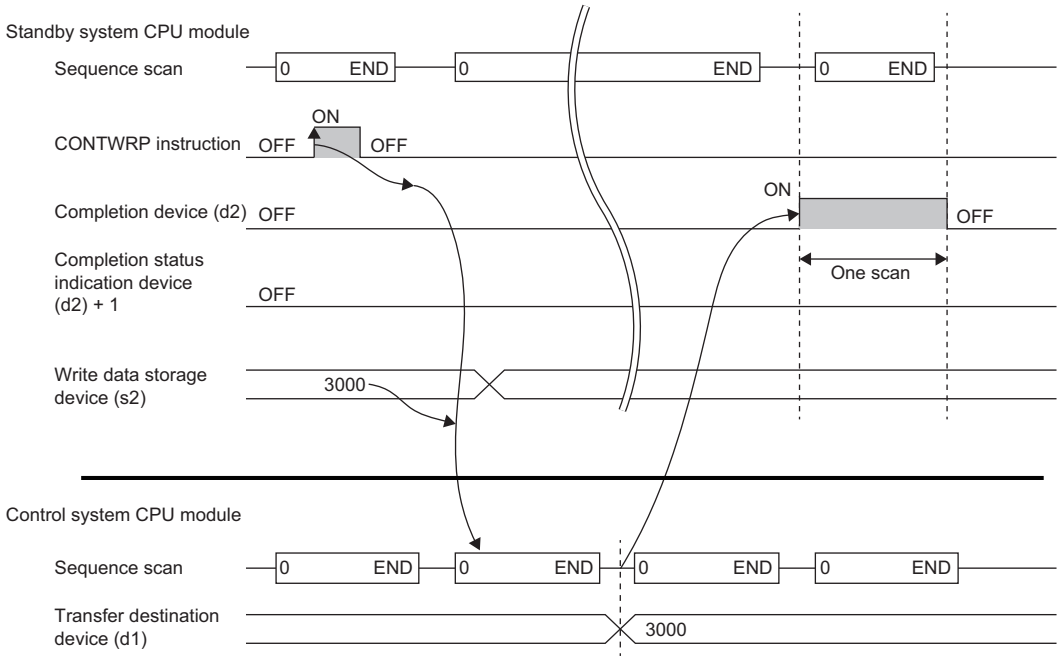


- The CONTWR(P) instruction can only be executed in the standby system. Even if this instruction is executed in the control system, no processing is performed.
- If the CONTWR(P) instruction is executed more than once, the instruction executed the second time or later is not processed until the first executed instruction is completed.
- If system switching occurs during execution of the CONTWR(P) instruction, the execution of the instruction continues without stopping. After the system switching, the completion device (d2) turns on in the new control system.
- The completion status (successful or with an error) of the CONTWR(P) instruction can be checked with the completion device (d2) specified in the setting data and the completion status indication device (d2)+1. When the instruction is completed with an error, the data write disabling cause is stored in the completion status (s1)+1.

Item	Description
Completion device (d2)	Turns on at END processing in the scan where the CONTWR(P) instruction has been completed, and turns off at the next END processing.
Completion status indication device (d2)+1	Turns on/off depending on the status upon completion of the CONTWR(P) instruction. Completed successfully: The device remains off. Completed with an error: The device turns on at END processing in the scan where the CONTWR(P) instruction has been completed, and turns off at the next END processing.
Completion status (s1)+1	Stores the following data write disabling causes. 0: Writing completed successfully 1: Tracking communications disabled 2: The other system in process mode 3: Control system not supporting the CONTWR(P) instruction 4: Device/label assignment mismatch with control system 5: Control system device/label out of range

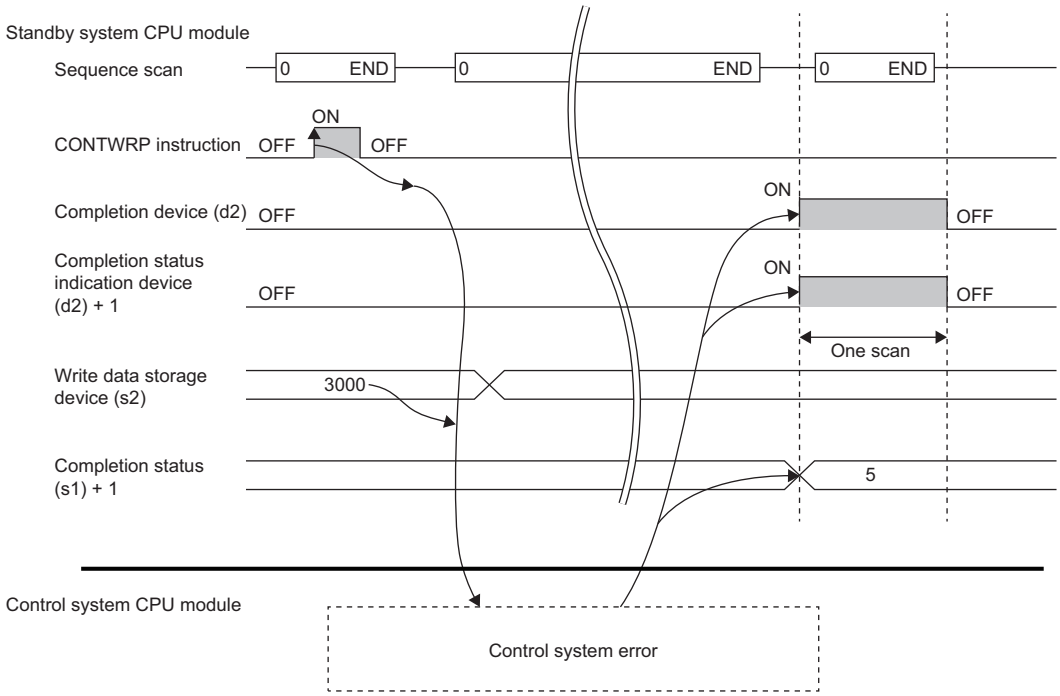
Ex.

When completed successfully



Ex.

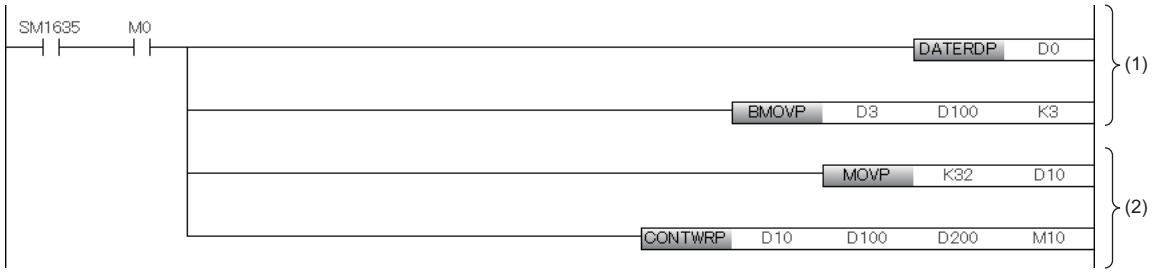
Completed with an error when the control system device/label is out of the range (completion status (s1)+1 = 5)



- By including the clock data at the time of instruction execution in the data to be written from the standby system to the control system, the last time at which the CONTWR(P) instruction has been completed successfully can be checked in the control system.

Ex.

Writing data including clock data at the beginning to the control system



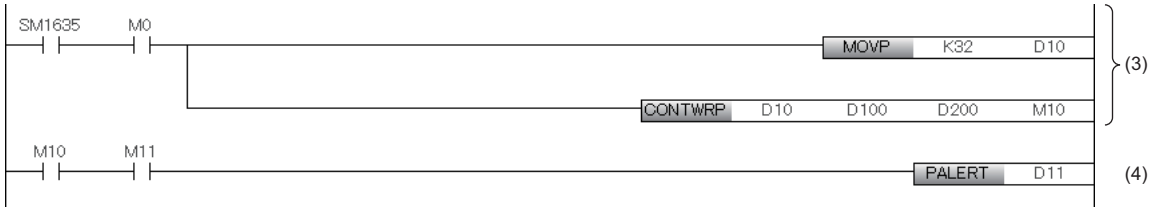
(1) Store the clock data (hours, minutes, seconds) acquired by the DATERDP instruction in D100 to D102 (the beginning of the data to be written to the control system).

(2) Issue the CONTWRP instruction to write the data of D100 to D131 of the standby system to D200 to D231 of the control system.

- When the CONTWR(P) instruction is completed with an error, the PALERT(P) instruction can cause a continuation error to notify completion with an error. The error code of the PALERT(P) instruction executed in the standby system and Detailed Information 2 can be checked with SD1681 (Latest self-diagnostics error code (the other system)) and SD1722 to SD1752 (Detailed Information 2 (the other system)) of the control system. (The latest error only)

Ex.

Notifying the data write disabling cause as well when the instruction is completed with an error



(3) Issue the CONTWRP instruction to write the data of D100 to D131 of the standby system to D200 to D231 of the control system.

(4) If the CONTWRP instruction is completed with an error, the PALERT instruction is executed to store D11 (data write disabling cause) in Detailed Information 2 of error code 1810H.

Precautions

- For devices (s1) and (s2), use a device/label not targeted for tracking transfer or a local device/local label. If a device/label targeted for tracking transfer is specified, the standby system may be overwritten by the control system data and an unintended operation may be performed.
- Execute the CONTWR(P) instruction with files of both systems being the same. If the instruction is executed while files of both systems are not the same, the instruction may be completed with an error.
- When the following device/label is specified in (d1), the data write destination is determined by the state of the standby system at the time of instruction execution.

Item	Description
File register (such as R10.)	Data is written to the file register of the control system having the same file name and block number of the file register used in the standby system at the time of instruction execution. At the time of data writing to the control system, if there is no file register file having the same name or if the file size does not match, the instruction is completed with an error.
Index modification (such as W100Z0)	At the time of execution of the instruction, the data write destination device in the control system is determined by the value stored in the index register of the standby system.
Indirect specification (such as @D2000)	At the time of execution of the instruction, the data write destination device in the control system is determined by the indirect address stored in the device of the standby system.
Label array (such as wLabel1[D0])	At the time of execution of the instruction, the data write destination label in the control system is determined by the value stored in the array index of the standby system.

- If the CONTWR(P) instruction is executed during writing to the programmable controller, online program change, or writing the file with the SLMP, FTP server function, or if data is written to the file during execution of the CONTWR(P) instruction, the instruction may be completed with an error due to mismatch of device/label assignment between both systems. If the instruction is completed with an error, re-execute the instruction after file writing.

Operation error

Error code (SD0)	Description
3405H	The number of write data points (s1)+0 is other than 1 to 1024.

30 SAFETY SYSTEM INSTRUCTIONS

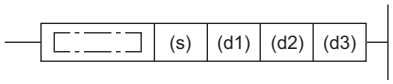
30.1 Reading Safety Data Identify Check Information

SP.SIDRD

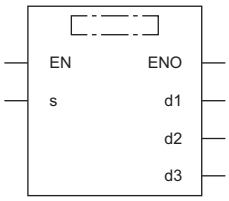
RnCPU ✗ | RnENCPU ✗ | RnPCPU (Process) ✗ | RnPCPU (Redundant) ✗ | RnPSFCPU (Standard) ✗ | RnPSFCPU (Safety) ✗ | RnSFCPU (Standard) | RnSFCPU (Safety) ✗

- The RnSFCPU (standard) with firmware version "15" or later supports this instruction. Use an engineering tool with version "1.050C" or later.

This instruction reads the identifier for the safety data identify check file from the specified file.

Ladder	ST
	ENO:=SP_SIDRD(EN,s,d1,d2,d3);

FBD/LD



■Execution condition

Instruction	Execution condition
SP.SIDRD	↑

Setting data

■Description, range, data type

Operand	Description	Range	Data type	Data type (label)
(s)	Start device for storing the file name	—	Unicode string	ANYSTRING_DOUBLE
(d1)	Device for storing the completion status 0000: Completed successfully Other than 0000: Completed with an error (error code)	—	Word	ANY16
(d2)	Start device for storing the file identifier that has been read	—	Double word	ANY32
(d3)	Bit device that turns on upon completion of the processing (When the instruction is complete with an error, (d3)+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□\□, U3E□\(H)G□	Z	LT, LST, LC	LZ		K, H	E	$	
(s)	—	—	○	—	—	—	—	○	—	—	○	—
(d1)	—	—	○	—	—	—	—	○	—	—	—	—
(d2)	—	—	○	—	—	—	—	○	—	—	—	—
(d3)	○	—	○	—	—	—	—	○	—	—	—	—

Processing details

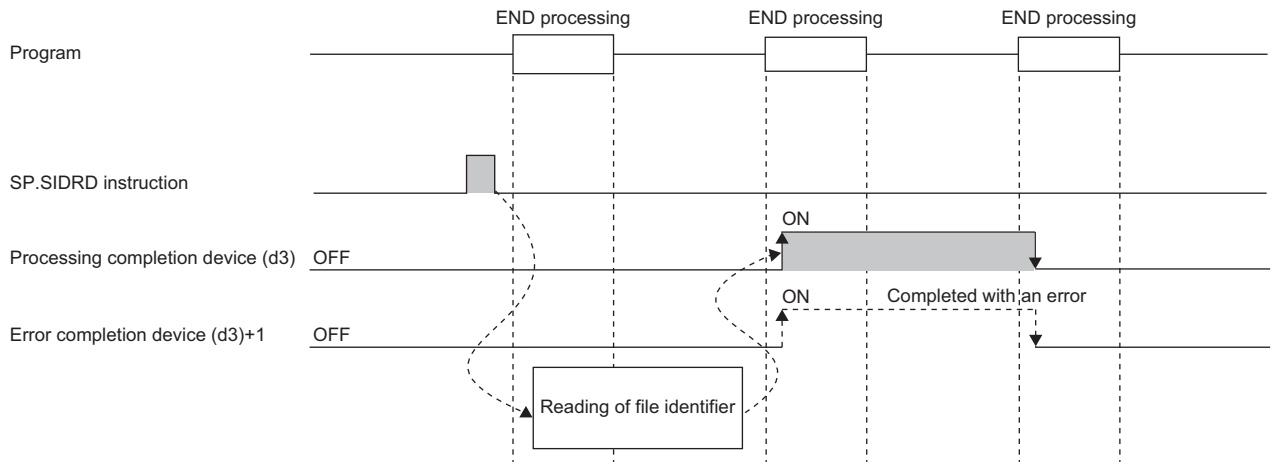
- This instruction reads the identifier for safety data identify check file from the file specified by (s) and stores it to the device specified by (d2). The number of characters of the name of file specified by (s) must be 64 characters or less including the extension. For details of the safety data identify check function, refer to the following.

📖 MELSEC iQ-R CPU Module User's Manual (Application)

- The table below lists the files that can be specified by (s).

File type	File name	Extension
Safety program	Any	SPG
Safety FB file	Any	SPB
Safety CPU parameter	CPU	SPR
Safety module parameter	UNIT	

- The bit device of processing completion (d3) automatically turns on at the END processing of the scan that has detected the completion of the SP.SIDRD instruction. The bit device turns off at the END processing in the next scan.
- If the SP.SIDRD instruction is complete with an error, the device of error completion ((d3)+1) turns on/off in synchronization with the device of processing completion (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.
- SM753 (File being accessed) turns on while the SP.SIDRD instruction is being executed. When SM753 is on, the SP.SIDRD instruction cannot be executed. (If the instruction is executed, no processing is performed.)
- The following figure shows the operation of the completion device at execution of the SP.SIDRD instruction.



Precautions

- Do not execute the SP.SIDRD instruction in interrupt programs. Doing so may cause malfunction of the module.
- Do not execute the SP.SIDRD instruction simultaneously with any other file access function.
- Even though the operating status of the CPU module is switched from RUN to STOP during instruction execution, the instruction continues until it completes the processing.

Operation error

Error code (SD0)	Description
3405H	The file name character string specified by (s) cannot be read. • The specified file name contains no character. • The specified file name contains 65 or more characters.

Error codes generated by safety system instructions

The following table lists the error codes that could be stored in the completion status (d1) of safety system instructions.

Error code (d1)	Error content	Action
8000H	The specified file does not exist.	Check whether the specified file exists.
8001H	The specified file has an invalid extension.	Check the extension of the specified file.
8002H	Another file access function is being executed.	Execute the instruction after the file access function is completed.
8004H	The specified file is damaged.	Rewrite the file specified in the CPU module.

MEMO

PART 6 MODULE DEDICATED INSTRUCTIONS

This part consists of the following chapter.

31 MODULE DEDICATED INSTRUCTIONS

31 MODULE DEDICATED INSTRUCTIONS

For the module dedicated instruction, refer to the following.

📖 MELSEC iQ-R Programming Manual (Module Dedicated Instructions)

MEMO

MEMO

PART 7 STANDARD FUNCTIONS

This part consists of the following chapters.

32 TYPE CONVERSION FUNCTIONS

33 SINGLE VARIABLE FUNCTIONS

34 ARITHMETIC OPERATION FUNCTIONS

35 BIT SHIFT FUNCTIONS

36 BOOLEAN FUNCTIONS

37 SELECTION FUNCTIONS

38 COMPARISON FUNCTIONS

39 STRING FUNCTIONS

40 TIME DATA TYPE FUNCTIONS

32 TYPE CONVERSION FUNCTIONS

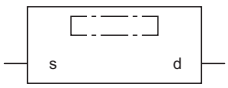
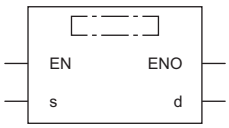
32.1 Converting BOOL to WORD

BOOL_TO_WORD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

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

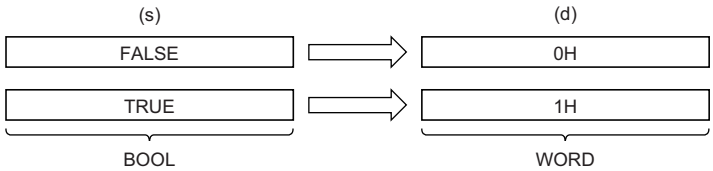
■Description, type, data type

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.

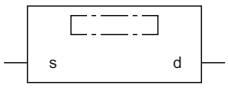
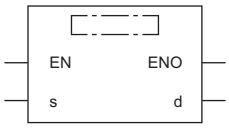
32.2 Converting BOOL to DWORD

BOOL_TO_DWORD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from BOOL data type to DWORD data type.

Ladder, FBD/LD		Structured text

[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

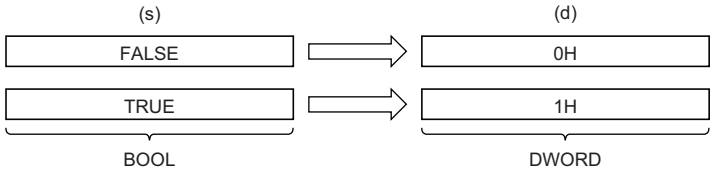
■Description, type, data type

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.

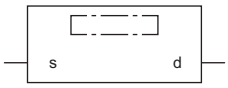
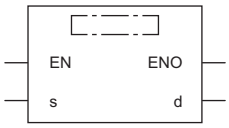
32.3 Converting BOOL to INT

BOOL_TO_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from BOOL data type to INT data type.

Ladder, FBD/LD		Structured text

[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

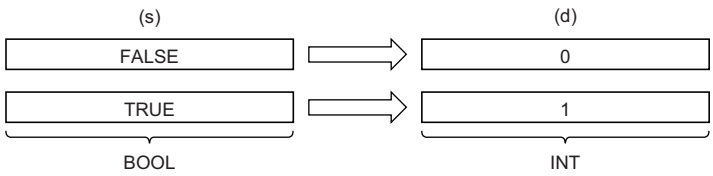
■Description, type, data type

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.

32.4 Converting BOOL to DINT

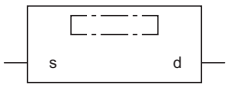
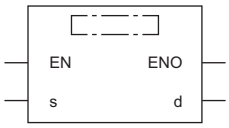
BOOL_TO_DINT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These instructions convert a value from BOOL data type to DINT data type.

Ladder, FBD/LD		Structured text

[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

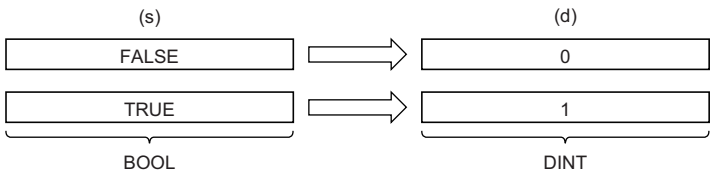
■Description, type, data type

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.

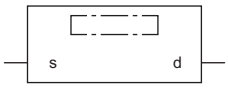
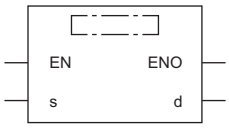
32.5 Converting BOOL to TIME

BOOL_TO_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from BOOL data type to TIME data type.

Ladder, FBD/LD		Structured text

[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

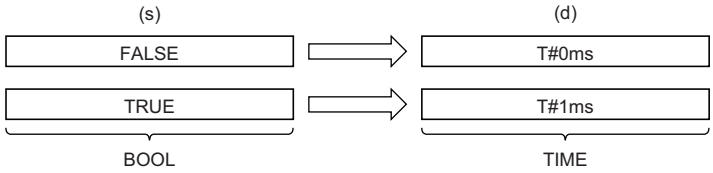
■Description, type, data type

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.

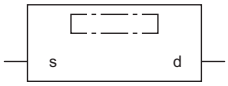
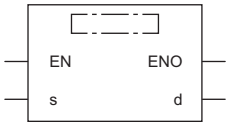
32.6 Converting BOOL to STRING

BOOL_TO_STRING(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from BOOL data type to STRING data type.

Ladder, FBD/LD		Structured text

[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

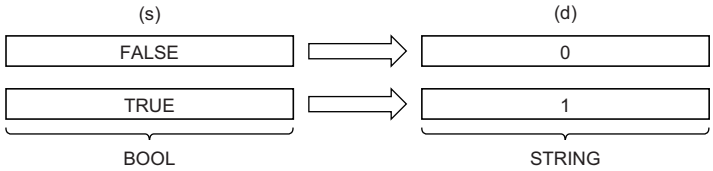
■Description, type, data type

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.

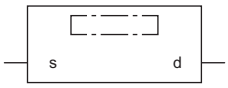
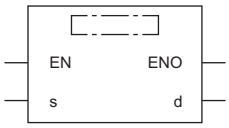
32.7 Converting WORD to BOOL

WORD_TO_BOOL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from WORD data type to BOOL data type.

Ladder, FBD/LD		Structured text

[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

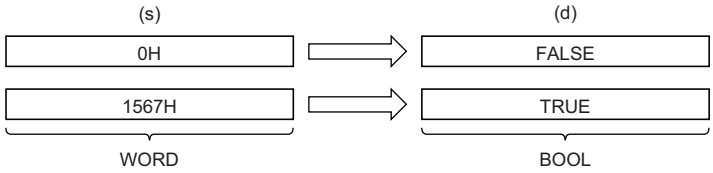
■Description, type, data type

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.

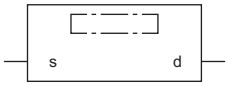
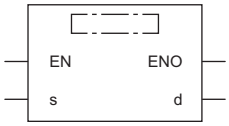
32.8 Converting WORD to DWORD

WORD_TO_DWORD(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from WORD data type to DWORD data type.

Ladder, FBD/LD		Structured text

[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

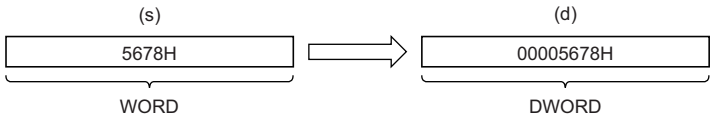
■Description, type, data type

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.

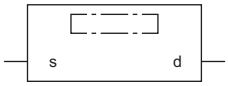
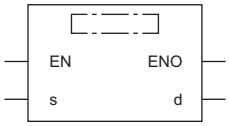
32.9 Converting WORD to INT

WORD_TO_INT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from WORD data type to INT data type.

Ladder, FBD/LD		Structured text

[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

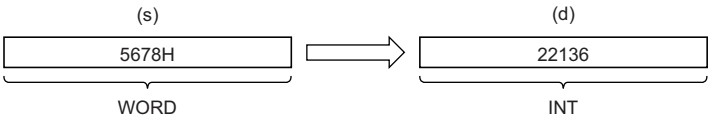
■Description, type, data type

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.

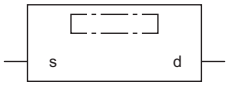
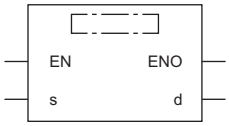
32.10 Converting WORD to DINT

WORD_TO_DINT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from WORD data type to DINT data type.

Ladder, FBD/LD		Structured text

[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

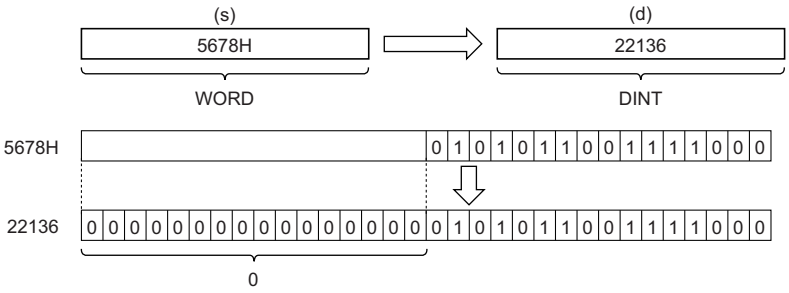
■Description, type, data type

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.

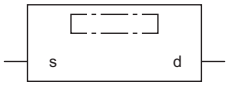
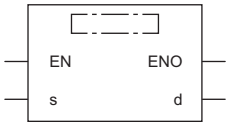
32.11 Converting WORD to TIME

WORD_TO_TIME(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from WORD data type to TIME data type.

Ladder, FBD/LD		Structured text

[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

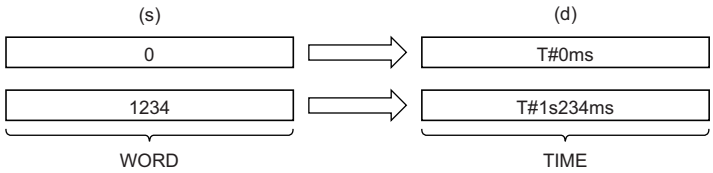
■Description, type, data type

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.

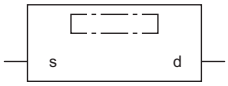
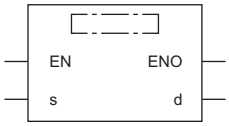
32.12 Converting WORD to STRING

WORD_TO_STRING(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from WORD data type to STRING data type.

Ladder, FBD/LD		Structured text

[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

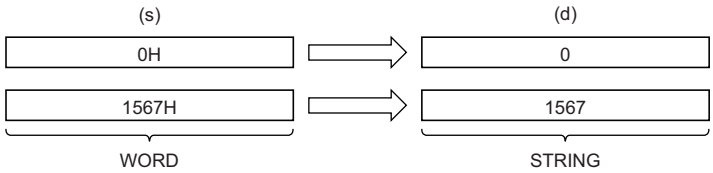
■Description, type, data type

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.

32.13 Converting DWORD to BOOL

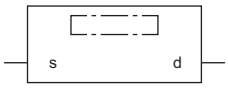
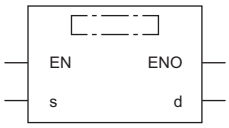
DWORD_TO_BOOL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from DWORD data type to BOOL data type.

Ladder, FBD/LD		Structured text

[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

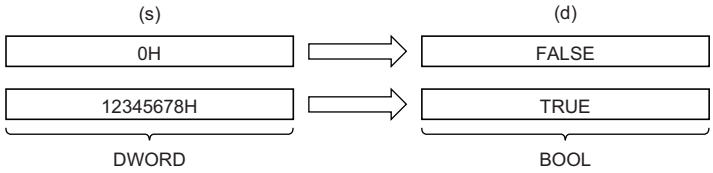
■Description, type, data type

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.

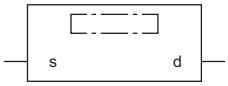
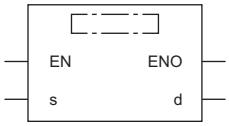
32.14 Converting DWORD to WORD

DWORD_TO_WORD(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from DWORD data type to WORD data type.

Ladder, FBD/LD		Structured text

[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

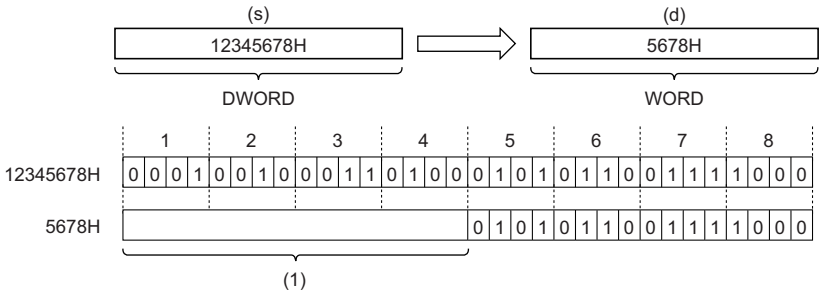
■Description, type, data type

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.

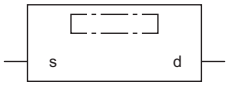
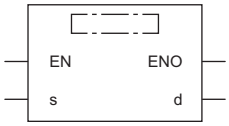
32.15 Converting DWORD to INT

DWORD_TO_INT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from DWORD data type to INT data type.

Ladder, FBD/LD		Structured text

[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

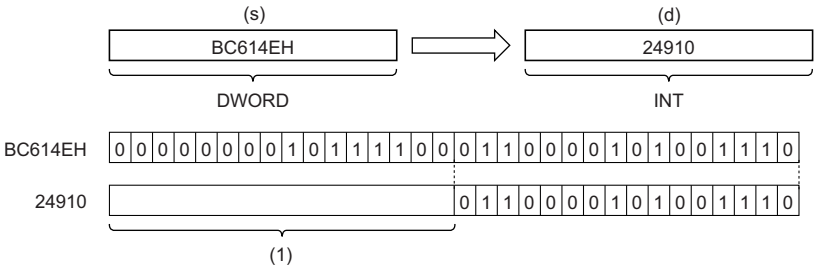
■Description, type, data type

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.

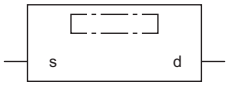
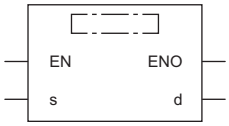
32.16 Converting DWORD to DINT

DWORD_TO_DINT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from DWORD data type to DINT data type.

Ladder, FBD/LD		Structured text

[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

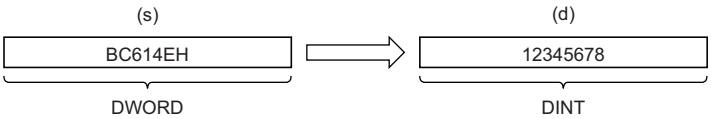
■Description, type, data type

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.

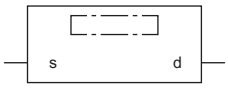
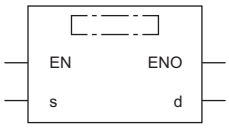
32.17 Converting DWORD to TIME

DWORD_TO_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from DWORD data type to TIME data type.

Ladder, FBD/LD		Structured text

[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

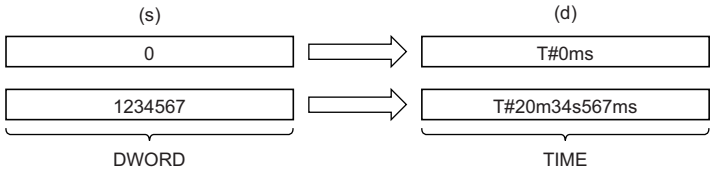
■Description, type, data type

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.

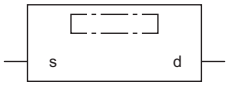
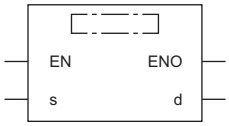
32.18 Converting DWORD to STRING

DWORD_TO_STRING(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from DWORD data type to STRING data type.

Ladder, FBD/LD		Structured text

[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

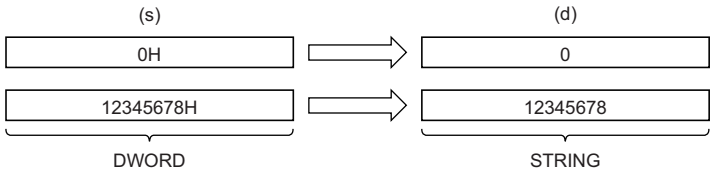
■Description, type, data type

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.

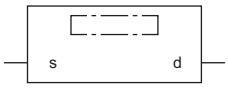
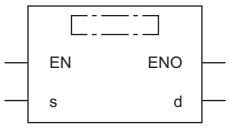
32.19 Converting INT to BOOL

INT_TO_BOOL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from INT data type to BOOL data type.

Ladder, FBD/LD		Structured text

[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

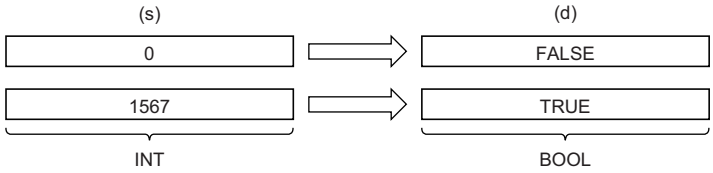
■Description, type, data type

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.

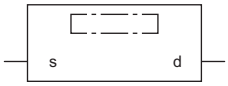
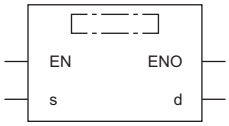
32.20 Converting INT to WORD

INT_TO_WORD(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from INT data type to WORD data type.

Ladder, FBD/LD		Structured text

[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

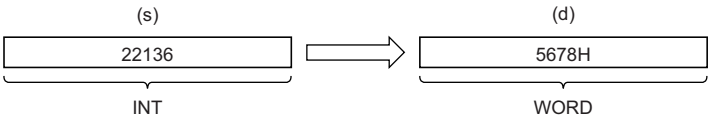
■Description, type, data type

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.

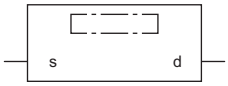
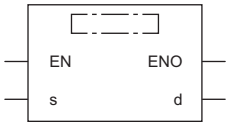
32.21 Converting INT to DWORD

INT_TO_DWORD(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from INT data type to DWORD data type.

Ladder, FBD/LD		Structured text

[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

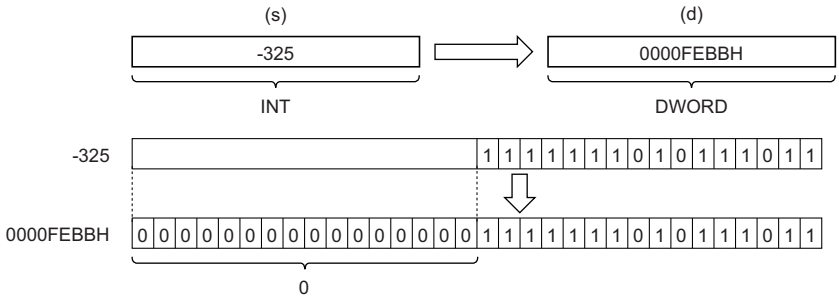
■Description, type, data type

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.

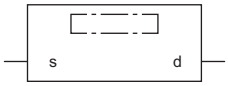
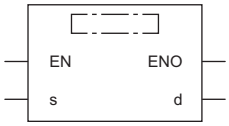
32.22 Converting INT to DINT

INT_TO_DINT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from INT data type to DINT data type.

Ladder, FBD/LD		Structured text

[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

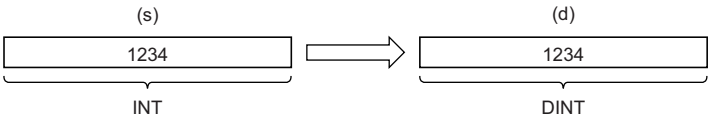
■Description, type, data type

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.

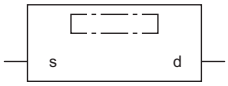
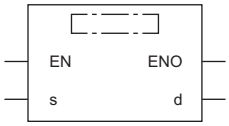
32.23 Converting INT to BCD

INT_TO_BCD(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from INT data type to BCD data type.

Ladder, FBD/LD		Structured text

[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

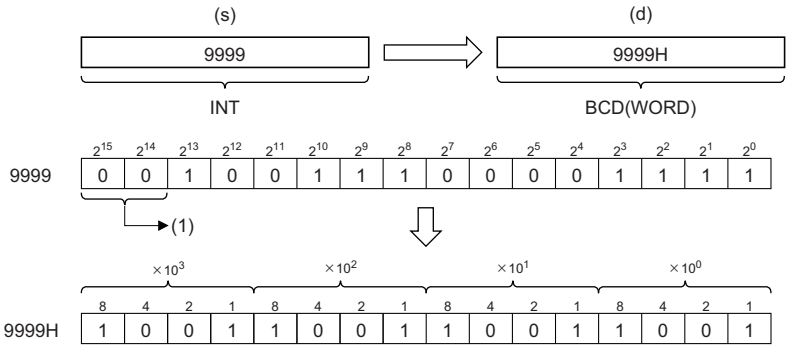
■Description, type, data type

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
3401H	Data input to (s) is out of the range, 0 to 9999.

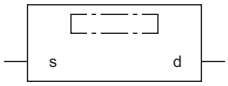
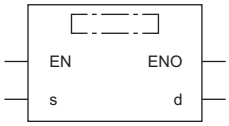
32.24 Converting INT to REAL

INT_TO_REAL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from INT data type to REAL data type.

Ladder, FBD/LD		Structured text

[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

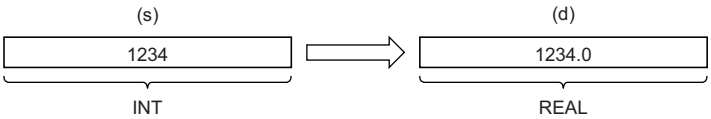
■Description, type, data type

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.

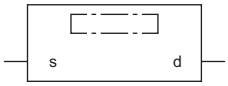
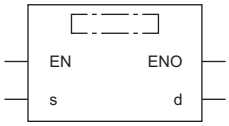
32.25 Converting INT to LREAL

INT_TO_LREAL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from INT data type to LREAL data type.

Ladder, FBD/LD		Structured text

[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

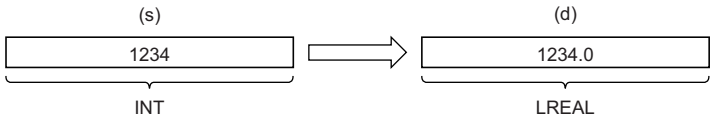
■Description, type, data type

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.

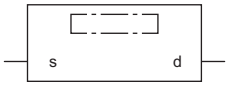
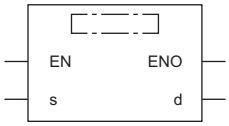
32.26 Converting INT to TIME

INT_TO_TIME(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from INT data type to TIME data type.

Ladder, FBD/LD		Structured text

[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

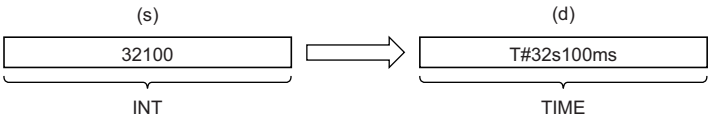
■Description, type, data type

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.

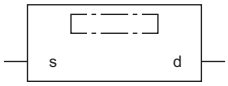
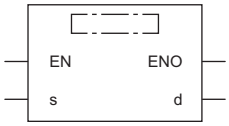
32.27 Converting INT to STRING

INT_TO_STRING(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from INT data type to STRING data type.

Ladder, FBD/LD		Structured text

[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

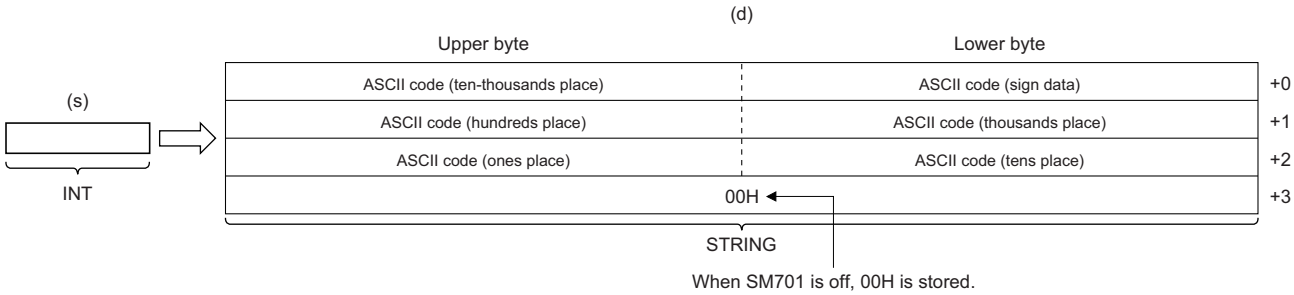
■Description, type, data type

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

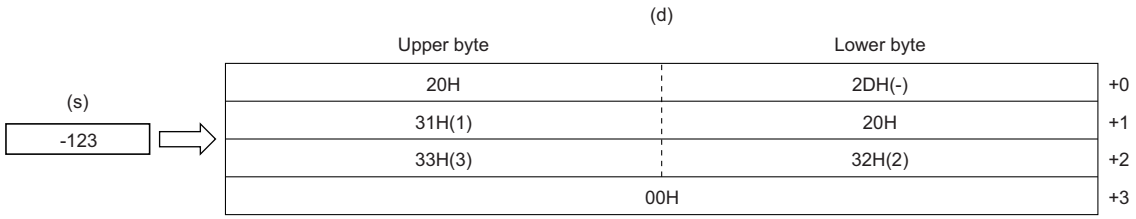
- These functions convert the value input to (s) from INT data type to STRING data type, and output the converted value from (d).



- 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.

32.28 Converting DINT to BOOL

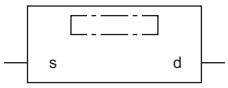
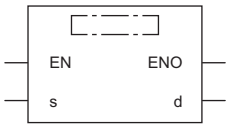
DINT_TO_BOOL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from DINT data type to BOOL data type.

Ladder, FBD/LD		Structured text

[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

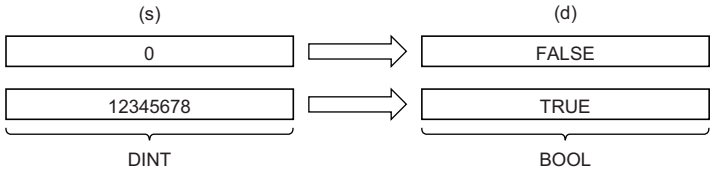
■Description, type, data type

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.

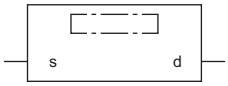
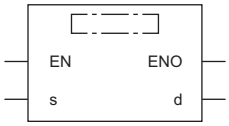
32.29 Converting DINT to WORD

DINT_TO_WORD(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from DINT data type to WORD data type.

Ladder, FBD/LD		Structured text

[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

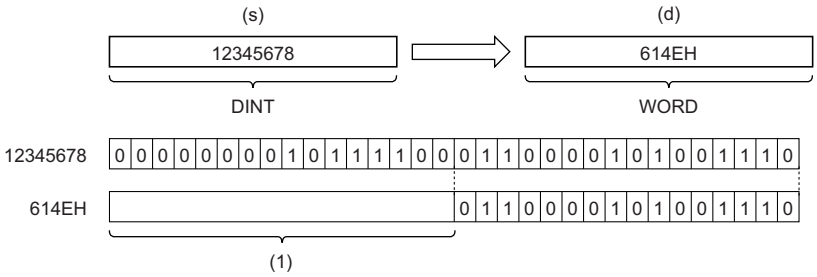
■Description, type, data type

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.

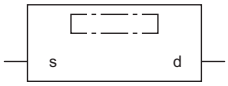
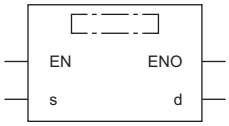
32.30 Converting DINT to DWORD

DINT_TO_DWORD(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from DINT data type to DWORD data type.

Ladder, FBD/LD		Structured text

[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

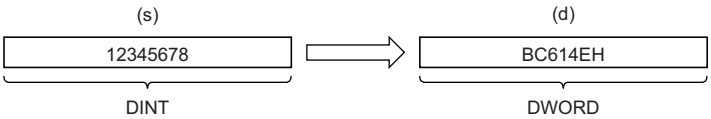
■Description, type, data type

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.

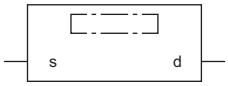
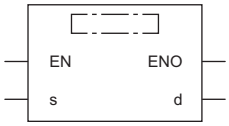
32.31 Converting DINT to INT

DINT_TO_INT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from DINT data type to INT data type.

Ladder, FBD/LD		Structured text

[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

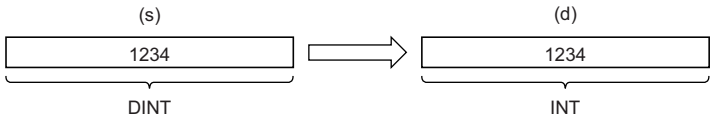
■Description, type, data type

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
3401H	The 32-bit signed binary data input to (s) is out of the range, -32768 to 32767.

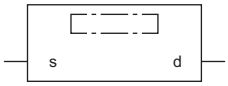
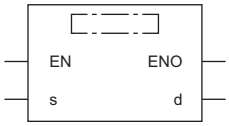
32.32 Converting DINT to BCD

DINT_TO_BCD(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from DINT data type to BCD data type.

Ladder, FBD/LD		Structured text

[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

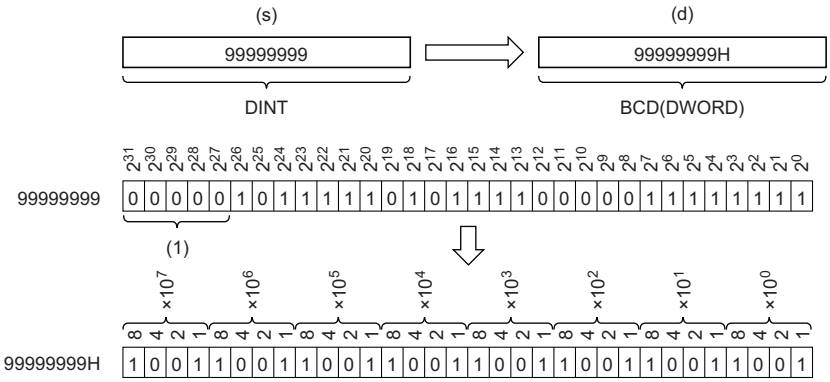
■Description, type, data type

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
3401H	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
3401H	Data input to (s) is out of the range, 0 to 99999999.

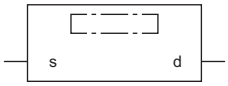
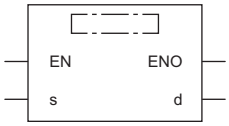
32.33 Converting DINT to REAL

DINT_TO_REAL(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from DINT data type to REAL data type.

Ladder, FBD/LD		Structured text

[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

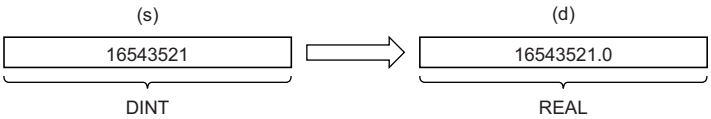
■Description, type, data type

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.

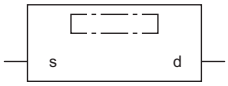
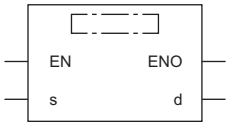
32.34 Converting DINT to LREAL

DINT_TO_LREAL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from DINT data type to LREAL data type.

Ladder, FBD/LD		Structured text

[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

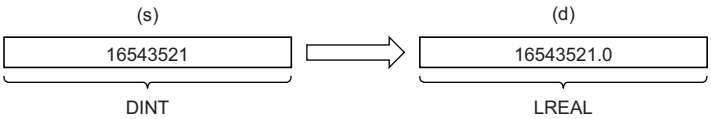
■Description, type, data type

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.

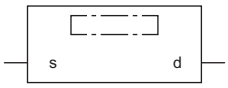
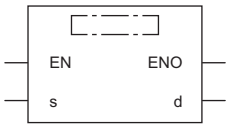
32.35 Converting DINT to TIME

DINT_TO_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from DINT data type to TIME data type.

Ladder, FBD/LD		Structured text

[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

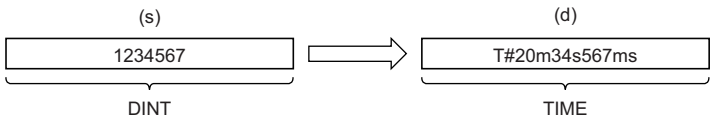
■Description, type, data type

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.

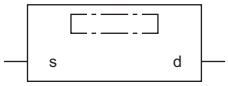
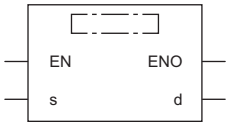
32.36 Converting DINT to STRING

DINT_TO_STRING(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from DINT data type to STRING data type.

Ladder, FBD/LD		Structured text

[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

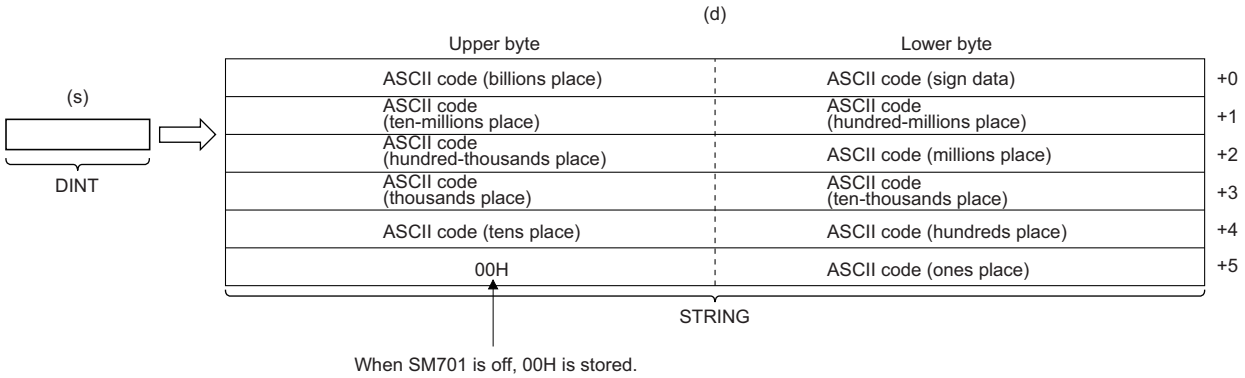
■Description, type, data type

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

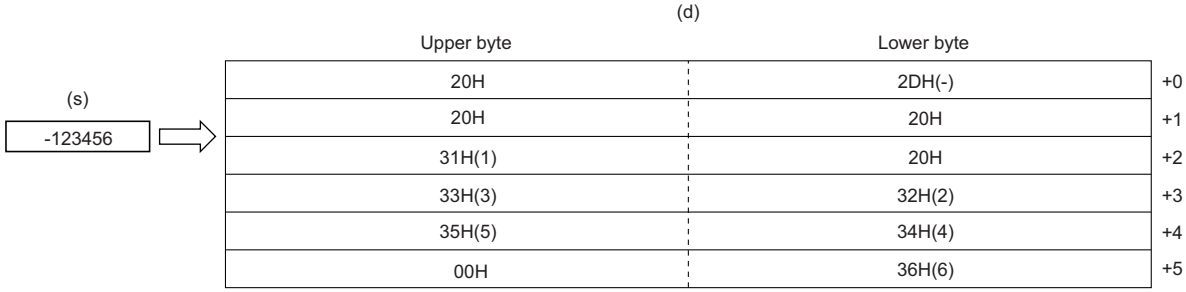
- These functions convert the value input to (s) from DINT data type to STRING data type, and output the converted value from (d).



- 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.

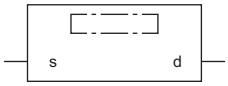
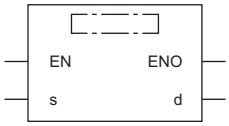
32.37 Converting BCD to INT

BCD_TO_INT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from BCD data type to INT data type.

Ladder, FBD/LD		Structured text

[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

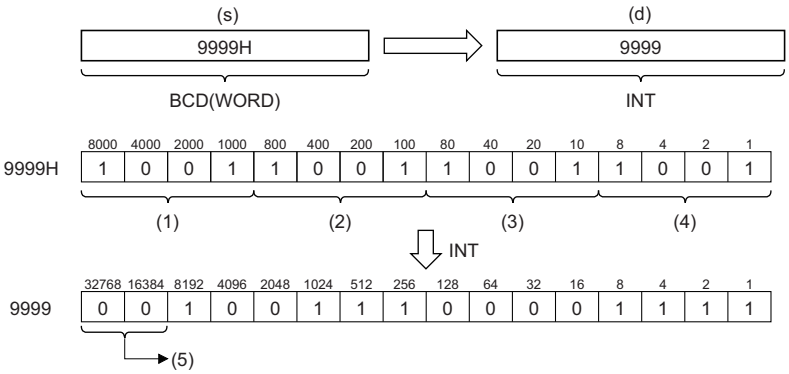
■Description, type, data type

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
3401H	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.

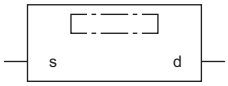
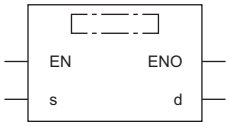
32.38 Converting BCD to DINT

BCD_TO_DINT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from BCD data type to DINT data type.

Ladder, FBD/LD		Structured text

[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

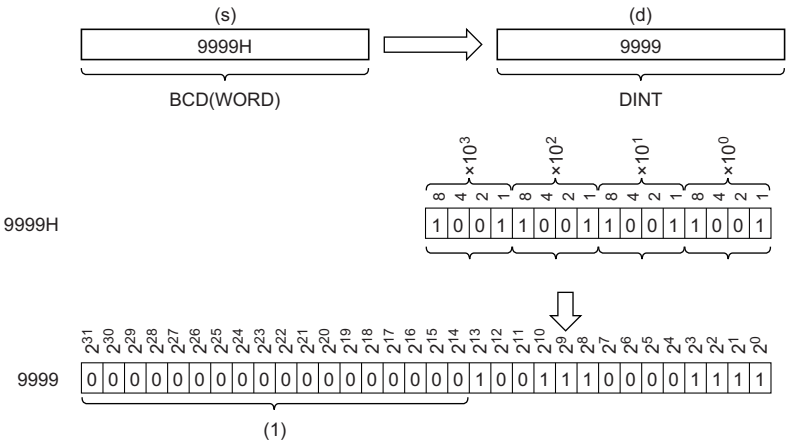
Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	DINT

Processing details

■Operation processing

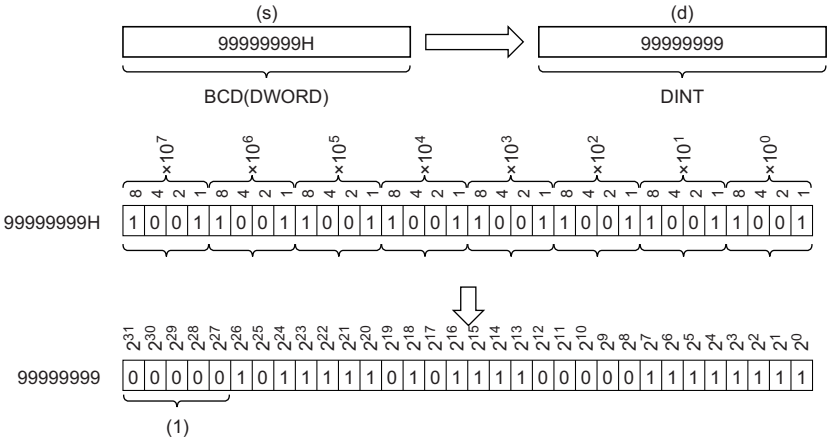
- These functions convert the value input to (s) from BCD (WORD or DWORD) data type to DINT data type, and output the converted value from (d).

• 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
3401H	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.

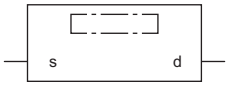
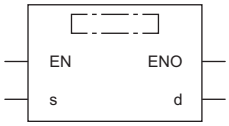
32.39 Converting BCD to STRING

BCD_TO_STRING(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from BCD data type to STRING data type.

Ladder, FBD/LD		Structured text

[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

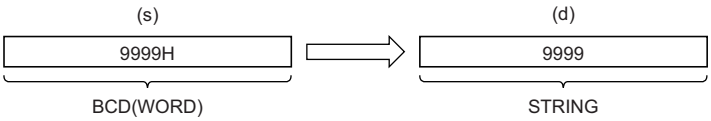
Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY_BIT
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	STRING(8)

Processing details

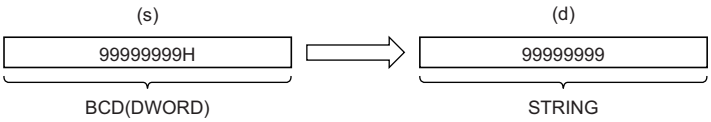
■Operation processing

- 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
3401H	Data input to (s) is out of the range, 0 to 9999.

- When (s) is of DWORD data type

Error code (SD0)	Description
3401H	Data input to (s) is out of the range, 0 to 99999999.

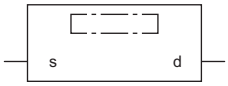
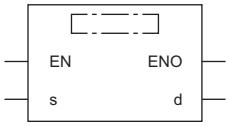
32.40 Converting REAL to INT

REAL_TO_INT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from REAL data type to INT data type.

Ladder, FBD/LD		Structured text

[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

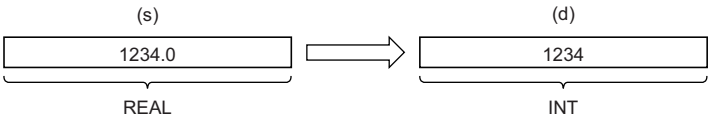
■Description, type, data type

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
3401H	The single-precision real number input to (s) is out of the range, -32768 to 32767.
3402H	• An unusual number is input 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 ±∞.

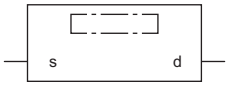
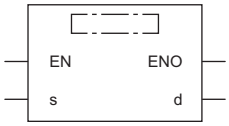
32.41 Converting REAL to DINT

REAL_TO_DINT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from REAL data type to DINT data type.

Ladder, FBD/LD		Structured text

[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

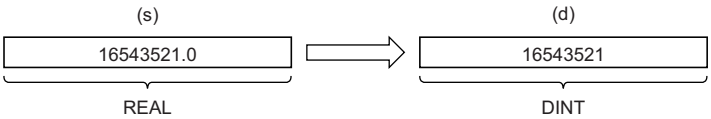
■Description, type, data type

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
3401H	The single-precision real number input to (s) is out of the range, -2147483648 to 2147483647.
3402H	• An unusual number is input 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 ±∞.

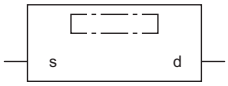
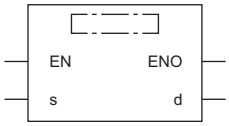
32.42 Converting REAL to LREAL

REAL_TO_LREAL(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from REAL data type to LREAL data type.

Ladder, FBD/LD		Structured text

[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

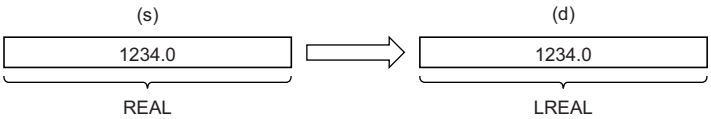
■Description, type, data type

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
3402H	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)
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

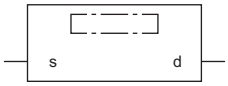
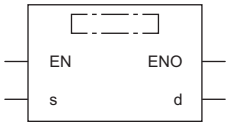
32.43 Converting REAL to STRING

REAL_TO_STRING(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions converts a REAL data type value to STRING data type (exponential form).

Ladder, FBD/LD		Structured text

[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

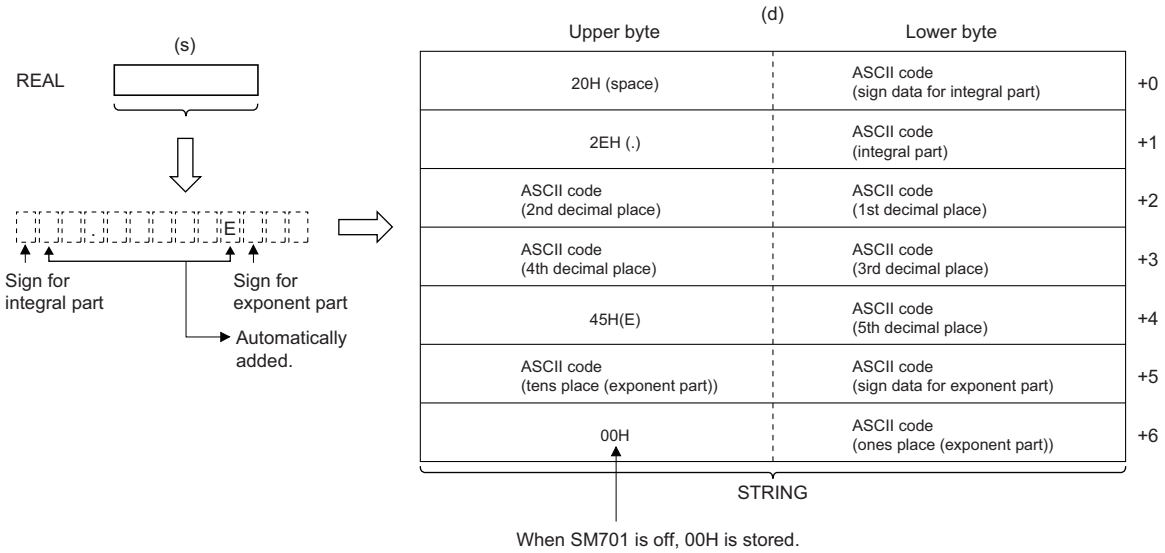
■Description, type, data type

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)

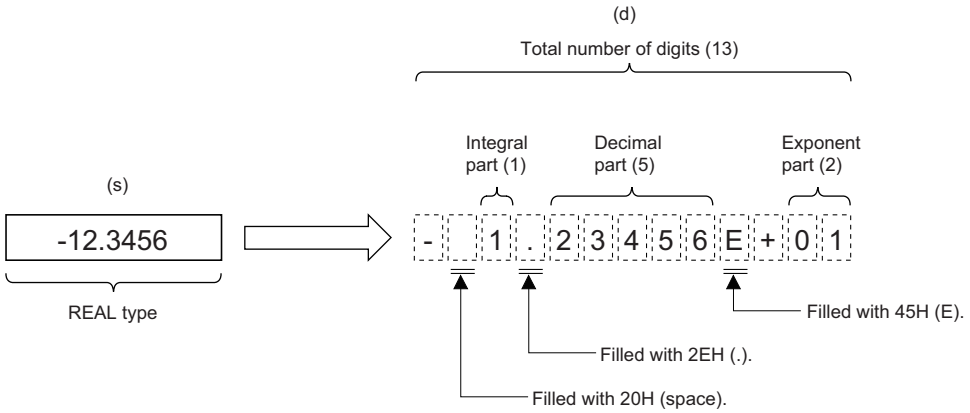
Processing details

■Operation processing

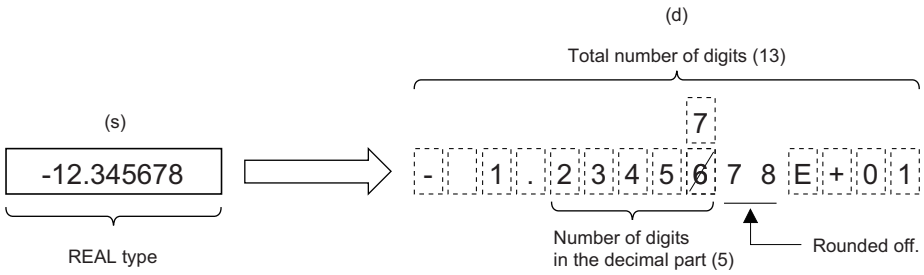
- These functions convert the value input to (s) from REAL data type to STRING data type (exponential form), and output the converted value from (d).



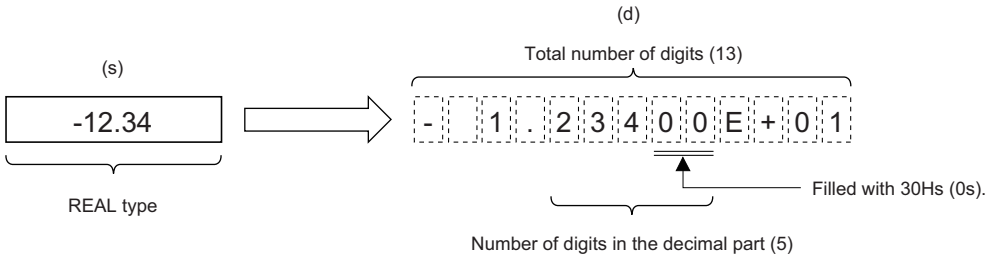
- Input a REAL data type value to (s).
- The converted string data is output from (d) as follows.
  - The number of digits for an integral part, decimal part, and exponent is fixed, integral part: one digit; decimal part: five digits; exponent: two digits.
  - As the second byte, 20H (space) is stored; as the fourth byte, 2EH (.) is stored; and as the 10th byte, 45H (E) is stored automatically.



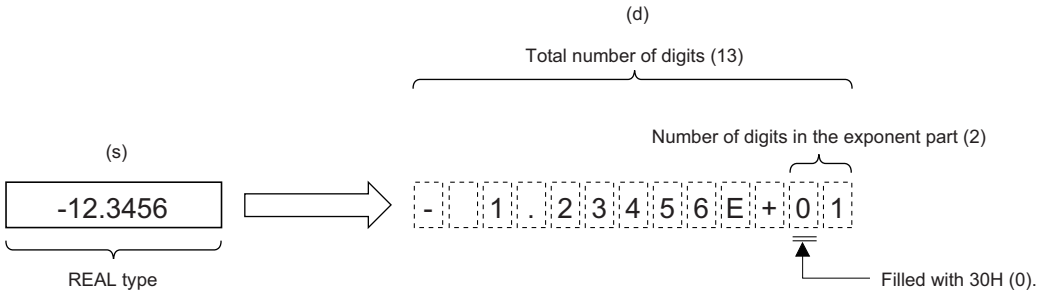
- As sign data (for integral part), 20H (space) is stored if the input value is positive, and 2DH (-) is stored if the input value is negative.
- The sixth and later digits of the decimal part are rounded off.



- 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
3402H	• 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 ±∞.
3406H	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.)

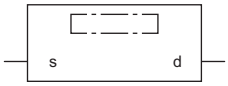
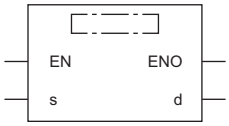
32.44 Converting LREAL to INT

LREAL_TO_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions convert a value from LREAL data type to INT data type.

Ladder, FBD/LD		Structured text

[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

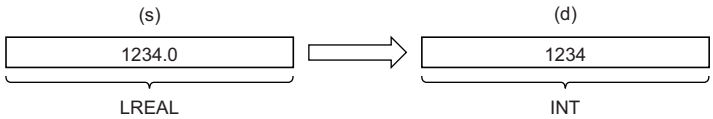
■Description, type, data type

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
3402H	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 data input to (s) is other than -32768 to 32767.

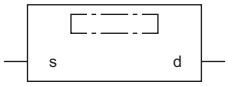
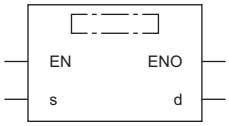
32.45 Converting LREAL to DINT

LREAL_TO_DINT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from LREAL data type to DINT data type.

Ladder, FBD/LD		Structured text

[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

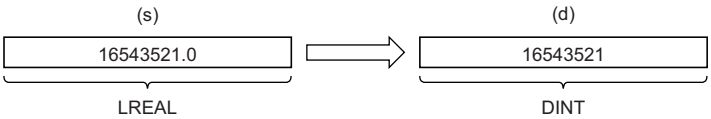
■Description, type, data type

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
3402H	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 data input to (s) is other than -2147483648 to 2147483647.

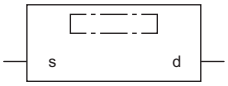
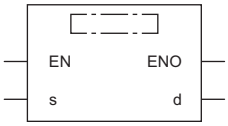
32.46 Converting LREAL to REAL

LREAL_TO_REAL(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from LREAL data type to REAL data type.

Ladder, FBD/LD		Structured text

[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

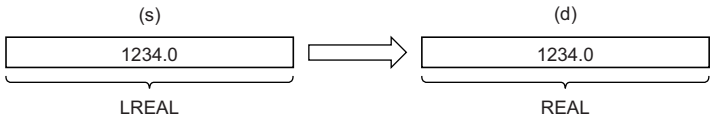
■Description, type, data type

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
3402H	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)
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\|(d)\|<2^{128}$

32.47 Converting TIME to BOOL

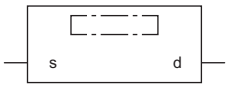
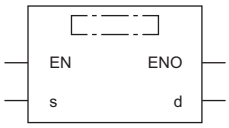
TIME_TO_BOOL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from TIME data type to BOOL data type.

Ladder, FBD/LD		Structured text

[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

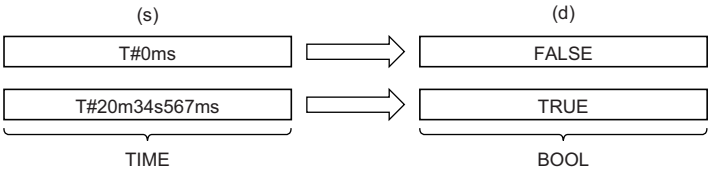
■Description, type, data type

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.

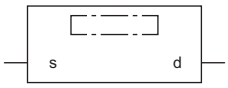
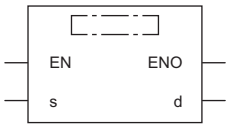
32.48 Converting TIME to WORD

TIME_TO_WORD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from TIME data type to WORD data type.

Ladder, FBD/LD		Structured text

[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

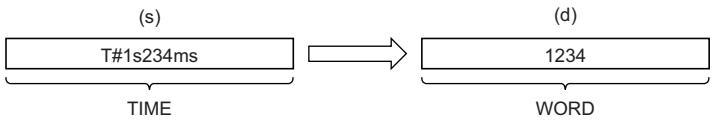
■Description, type, data type

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.

32.49 Converting TIME to DWORD

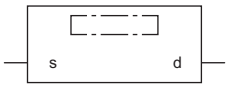
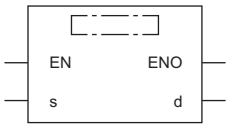
TIME_TO_DWORD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from TIME data type to DWORD data type.

Ladder, FBD/LD		Structured text

[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

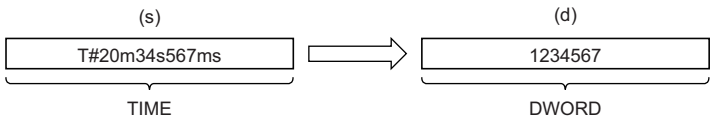
■Description, type, data type

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.

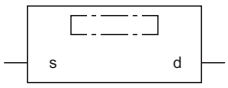
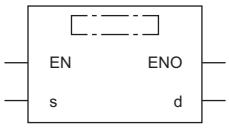
32.50 Converting TIME to INT

TIME_TO_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from TIME data type to INT data type.

Ladder, FBD/LD		Structured text

[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

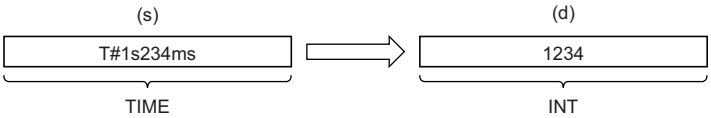
■Description, type, data type

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.

32.51 Converting TIME to DINT

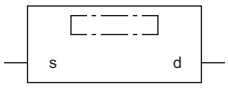
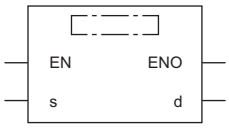
TIME_TO_DINT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert a value from TIME data type to DINT data type.

Ladder, FBD/LD		Structured text

[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

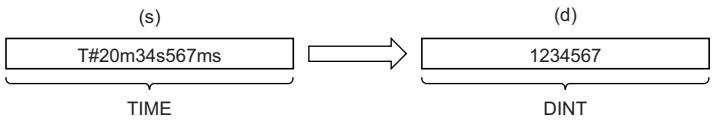
■Description, type, data type

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.

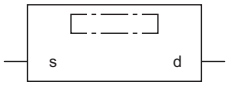
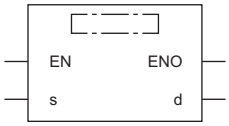
32.52 Converting TIME to STRING

TIME_TO_STRING(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from TIME data type to STRING data type.

Ladder, FBD/LD		Structured text

[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

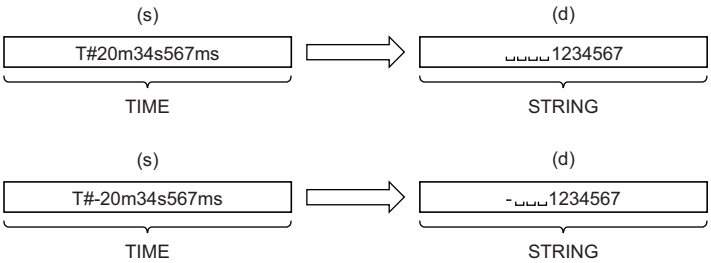
■Description, type, data type

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.

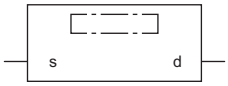
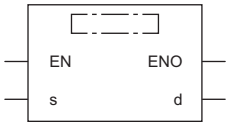
32.53 Converting STRING to BOOL

STRING_TO_BOOL(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from STRING data type to BOOL data type.

Ladder, FBD/LD		Structured text

[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

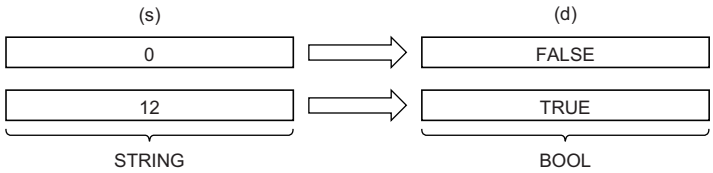
■Description, type, data type

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.

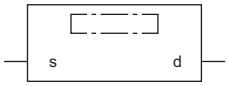
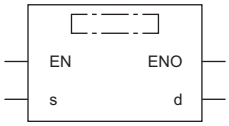
32.54 Converting STRING to WORD

STRING_TO_WORD(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from STRING data type to WORD data type.

Ladder, FBD/LD		Structured text

[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

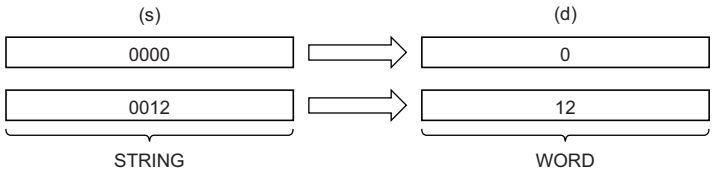
■Description, type, data type

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
3401H	An ASCII code other than 30H to 39H and 41H to 46H is input.

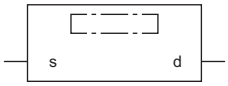
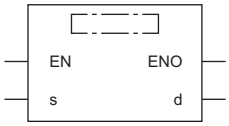
32.55 Converting STRING to DWORD

STRING_TO_DWORD(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from STRING data type to DWORD data type.

Ladder, FBD/LD		Structured text

[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

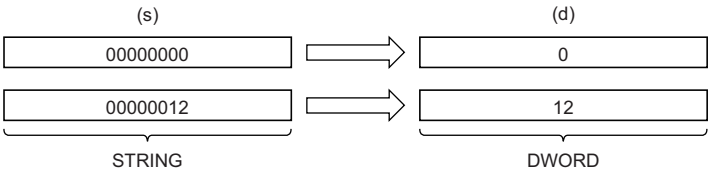
■Description, type, data type

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
3401H	An ASCII code other than 30H to 39H and 41H to 46H is input.

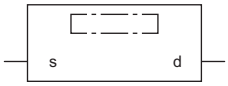
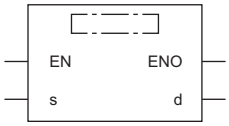
32.56 Converting STRING to INT

STRING_TO_INT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from STRING data type to INT data type.

Ladder, FBD/LD		Structured text

[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

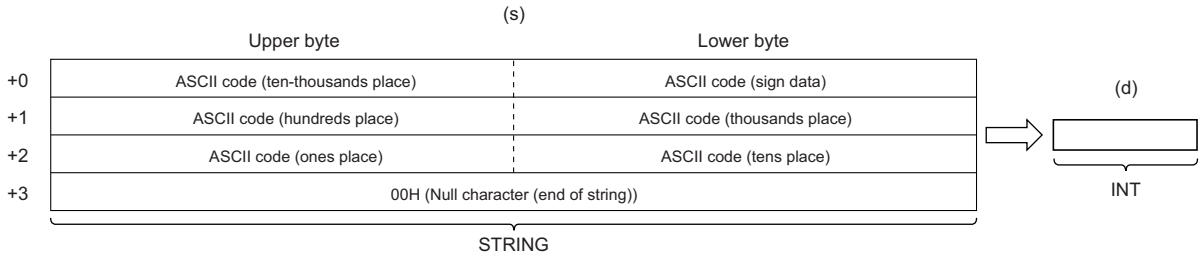
■Description, type, data type

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
3401H	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

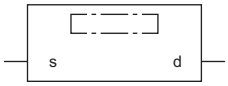
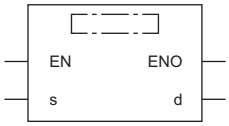
32.57 Converting STRING to DINT

STRING_TO_DINT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from STRING data type to DINT data type.

Ladder, FBD/LD		Structured text

[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

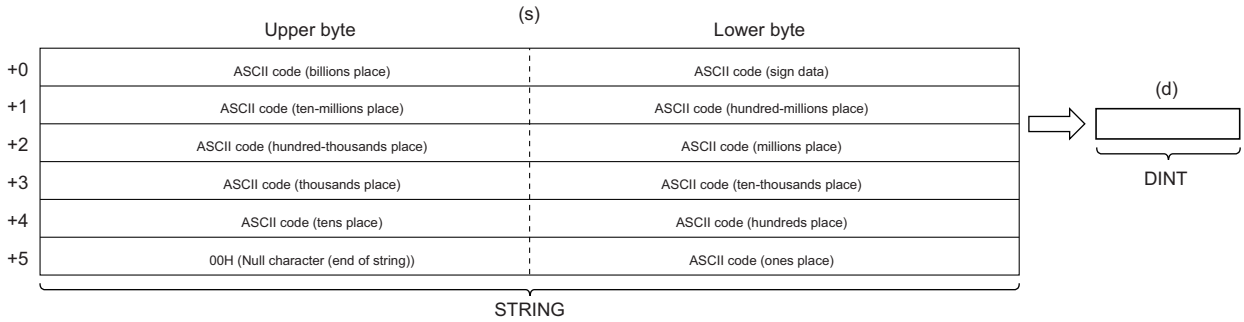
■Description, type, data type

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
3401H	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

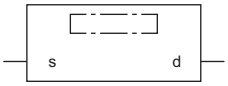
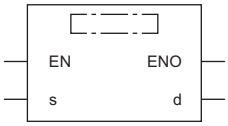
32.58 Converting STRING to BCD

STRING_TO_BCD(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from STRING data type to BCD data type.

Ladder, FBD/LD		Structured text

[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

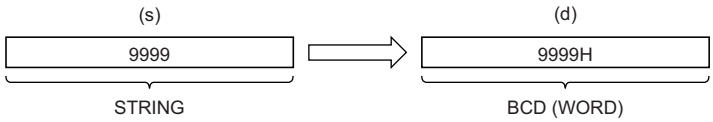
■Description, type, data type

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 character 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
3401H	An ASCII code other than 30H to 39H, 20H, or 00H is input.

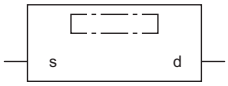
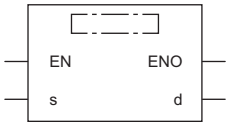
32.59 Converting STRING to REAL

STRING_TO_REAL(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from STRING data type to REAL data type.

Ladder, FBD/LD		Structured text

[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

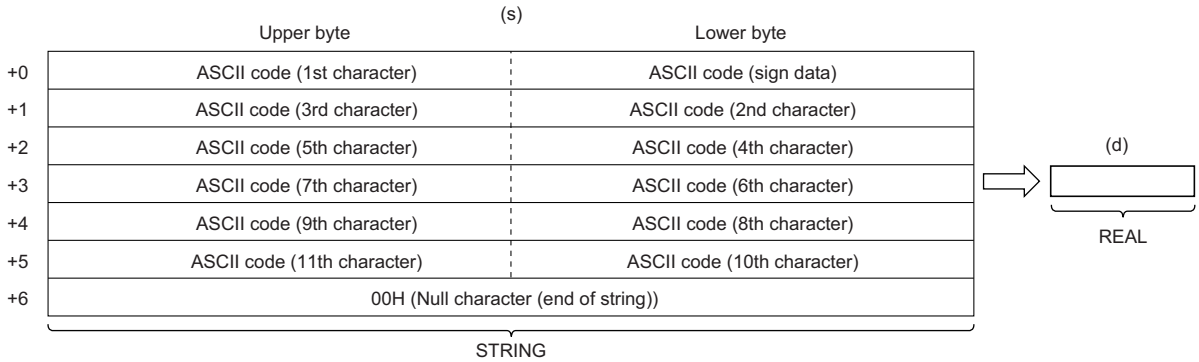
■Description, type, data type

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

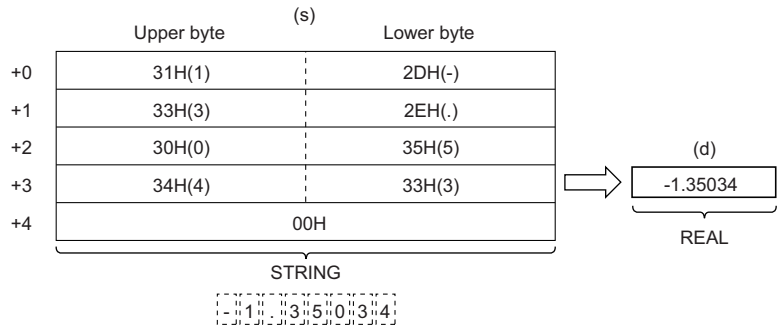
Processing details

■Operation processing

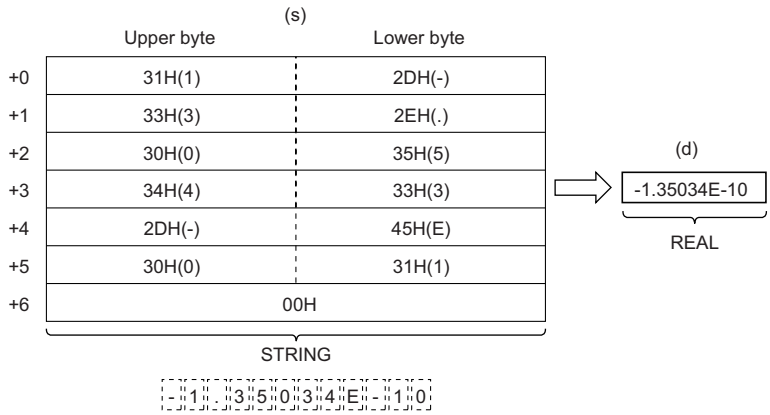
- These functions convert the value input to (s) from STRING data type (decimal form/exponential form) to REAL data type, and output the converted value from (d).



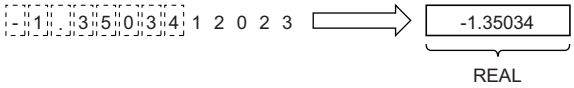
- The STRING data type value both in decimal form and exponential form can be converted.
  - When (s) is in decimal form



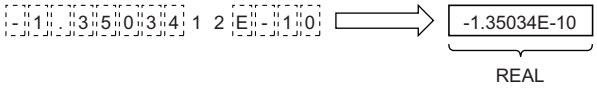
  - When (s) is in exponential form



- 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

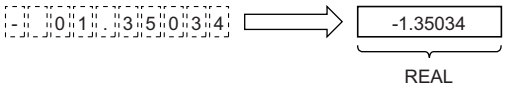


  - When (s) is in exponential form

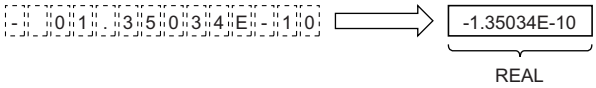


- In decimal form, when 2BH (+) is specified as sign data or the sign data is omitted, the data is converted as a positive value. When 2DH (-) is specified, the data is converted as a negative value.
- In exponential form, when 2BH (+) is specified as sign data for the exponent or the sign data is omitted, the data is converted as a positive value. When 2DH (-) is specified, the data is converted as a negative value.

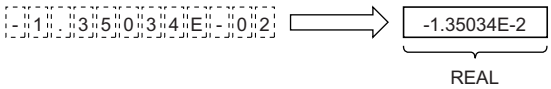
- The ASCII code 20H (space) or 30H (0) that exists before the first numerical value 0 in the STRING data type value is ignored.
  - When (s) is in decimal form



  - 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).
3401H	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 character string contains a character other than 45H(E), 65H(e), 2BH(+), and 2DH(-). • The specified character 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 character 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.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

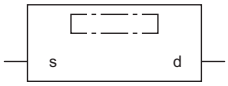
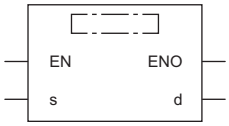
32.60 Converting STRING to TIME

STRING_TO_TIME(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions convert a value from STRING data type to TIME data type.

Ladder, FBD/LD		Structured text

[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

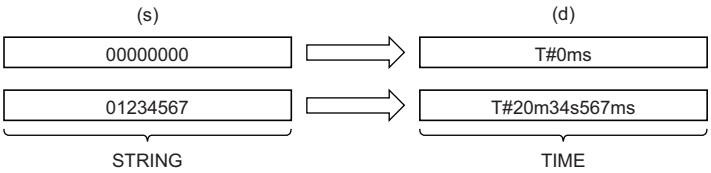
■Description, type, data type

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
3401H	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 4147483647

32.61 Converting Bit Array to INT

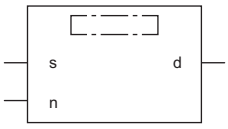
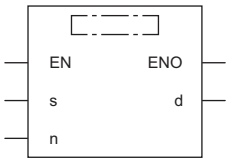
BITARR_TO_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert the specified number of bits in a bit array to an INT data type value.

Ladder, FBD/LD		Structured text

[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

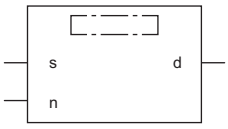
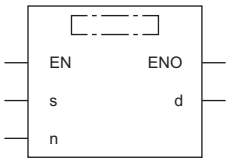
There is no operation error.

32.62 Converting Bit Array to DINT

BITARR_TO_DINT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions convert the specified number of bits in a bit array to a DINT data type value.

Ladder, FBD/LD		Structured text
[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.

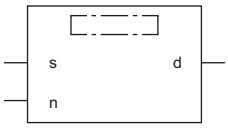
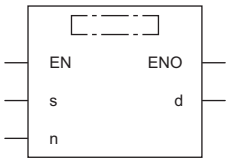
32.63 Converting INT to Bit Array

INT_TO_BITARR(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the lower n bits of the INT data type value to the bit array.

Ladder, FBD/LD		Structured text

[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.

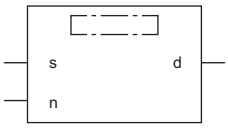
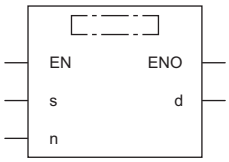
32.64 Converting DINT to Bit Array

DINT_TO_BITARR(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the lower n bits of the DINT data type value to the bit array.

Ladder, FBD/LD		Structured text

[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.

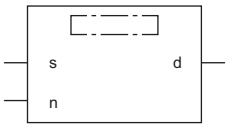
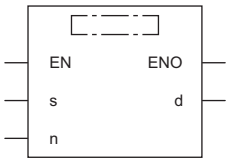
32.65 Copying the Bit Array

CPY_BITARR(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions copy the bit array by the specified number of bits.

Ladder, FBD/LD		Structured text

[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.

32.66 Reading the Specified Bit of the Word Label

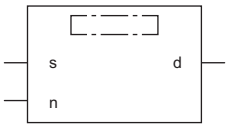
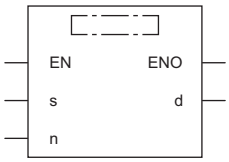
GET_BIT_OF_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions read a value from the specified bit of a word label.

Ladder*1		Structured text

[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
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

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

32.67 Writing the Specified Bit of the Word Label

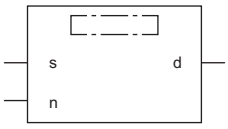
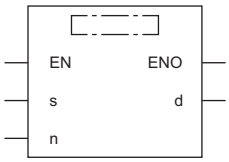
SET_BIT_OF_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions write a value to the specified bit of a word label.

Ladder*1		Structured text

[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

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

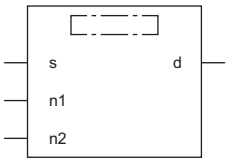
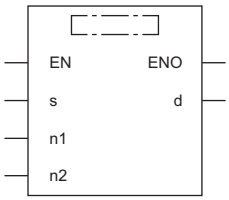
32.68 Copying the Specified Bit of the Word Label

CPY_BIT_OF_INT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions copy the specified bit of the word label to the specified bit of another word label.

Ladder*1		Structured text

[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
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

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

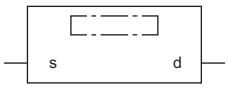
32.69 Getting the Start Data

GET_BOOL_ADDR, GET_INT_ADDR, GET_WORD_ADDR

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the start data of the specified array as BOOL, INT, or WORD type data.

Ladder, FBD/LD	Structured text

	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.

33 SINGLE VARIABLE FUNCTIONS

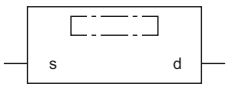
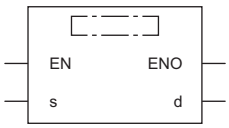
33.1 Calculating the Absolute Value

ABS(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the absolute value of an input value.

Ladder, FBD/LD		Structured text

[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*1
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_NUM*1

*1 Do not specify a REAL or LREAL data type in safety programs. An error occurs when the REAL or LREAL data type is specified.

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
3402H	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
3402H	The value output from (d) is -0, a subnormal number, NaN (not a number), or ±∞.

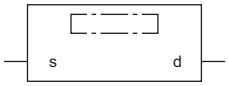
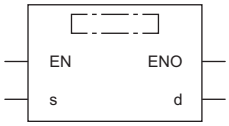
33.2 Calculating the Square Root

SQRT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions calculate the square root of an input value.

Ladder, FBD/LD		Structured text

[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
3405H	The input value is negative.

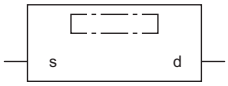
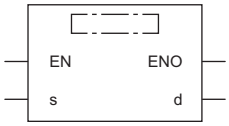
33.3 Calculating the Natural Logarithm

LN(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions output the natural logarithm (logarithm with base e) of an input value.

Ladder, FBD/LD		Structured text

[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
3405H	The input value is 0 or negative.

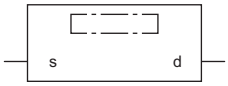
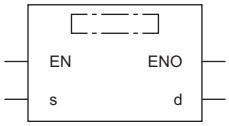
33.4 Calculating the Common Logarithm

LOG(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions output the common logarithm (logarithm with base 10) of an input value.

Ladder, FBD/LD		Structured text

[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
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	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
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3405H	Out-of-range data is set to (s). • The specified value is a negative number. • The specified value is 0.

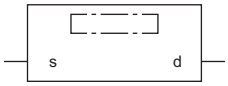
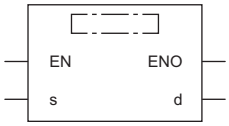
33.5 Calculating the Exponent

EXP(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions output the exponent of an input value.

Ladder, FBD/LD		Structured text

[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
3402H	The value input to (s) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

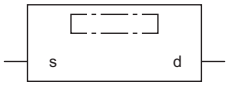
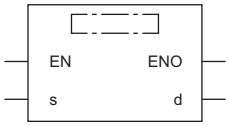
33.6 Calculating the Sine

SIN(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions output the sine of an input value.

Ladder, FBD/LD		Structured text

[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
3402H	The data specified by (s) is -0.

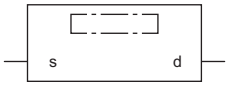
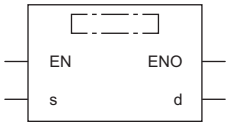
33.7 Calculating the Cosine

COS(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions output the cosine of an input value.

Ladder, FBD/LD		Structured text

[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
3402H	The data specified by (s) is -0.

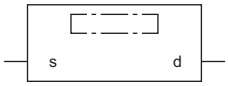
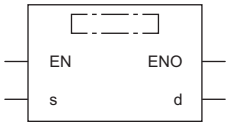
33.8 Calculating the Tangent

TAN(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions output the tangent of an input value.

Ladder, FBD/LD		Structured text

[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 $\pi/2$ radian or $(3/2)\pi$ 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$\times\pi/180$).

■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
3402H	The data specified by (s) is -0.

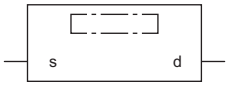
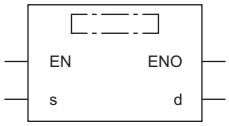
33.9 Calculating the Arc Sine

ASIN(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions output the arc sine (SIN^{-1}) of an input value.

Ladder, FBD/LD		Structured text

[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
3402H	The data specified by (s) is -0.
3405H	The value input with ASIN(_E) is other than -1.0 to 1.0.

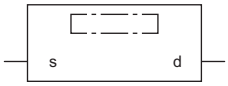
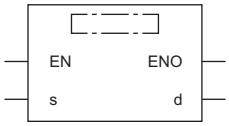
33.10 Calculating the Arc Cosine

ACOS(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions output the arc cosine (COS^{-1}) of an input value.

Ladder, FBD/LD		Structured text

[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
3402H	The data specified by (s) is -0.
3405H	The value input with ACOS(_E) is other than -1.0 to 1.0.

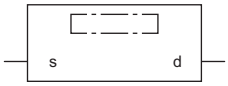
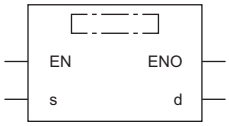
33.11 Calculating the Arc Tangent

ATAN(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These functions output the arc tangent (TAN^{-1}) of an input value.

Ladder, FBD/LD		Structured text

[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
3402H	The data specified by (s) is -0.

34 ARITHMETIC OPERATION FUNCTIONS

34.1 Addition

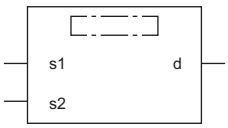
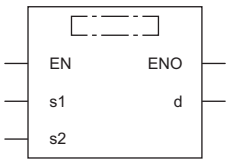
ADD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the sum of input values ((s1)+(s2)+···+(s28)).

Ladder, FBD/LD*1		Structured text*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*1
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_NUM*1

*1 Do not specify a REAL or LREAL data type in safety programs. An error occurs when the REAL or LREAL data type is specified.

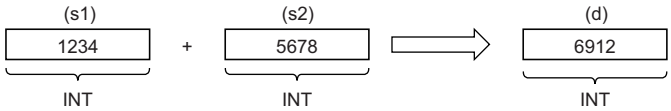
Processing details

■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.

Data type	Description
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
3402H	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 ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

- When (s1) to (s28) are of LREAL data type

Error code (SD0)	Description
3402H	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 ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

34.2 Multiplication

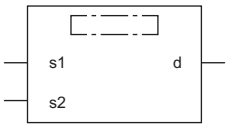
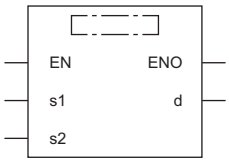
MUL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the product of input values ((s1)×(s2)×···×(s28)).

Ladder, FBD/LD*1		Structured text*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*1
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_NUM*1

*1 Do not specify a REAL or LREAL data type in safety programs. An error occurs when the REAL or LREAL data type is specified.

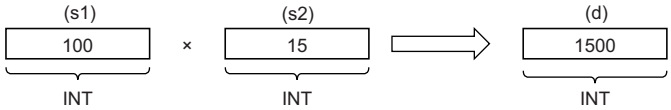
Processing details

■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
3402H	The value input to (s1) to (s28) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

- When (s1) to (s28) are of LREAL data type

Error code (SD0)	Description
3402H	The value input to (s1) to (s28) is -0, a subnormal number, NaN (not a number), or ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

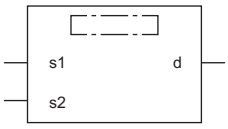
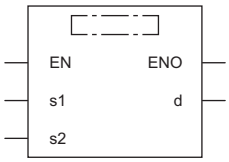
34.3 Subtraction

SUB(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the difference between input values ((s1)-(s2)).

Ladder, FBD/LD		Structured text

[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*1
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_NUM*1

*1 Do not specify a REAL or LREAL data type in safety programs. An error occurs when the REAL or LREAL data type is specified.

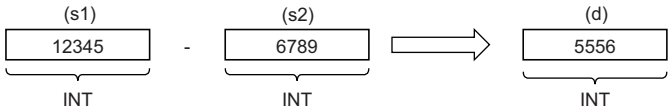
Processing details

■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.

Data type	Description
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
3402H	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 ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

- When (s1) and (s2) are of LREAL data type

Error code (SD0)	Description
3402H	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 ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

34.4 Division

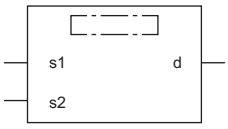
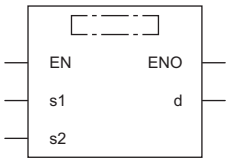
DIV(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the quotient of input values ((s1)÷(s2)).

Ladder, FBD/LD		Structured text

[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*1
s2 (IN2)	Divisor	Input variable	ANY_NUM*1
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_NUM*1

*1 Do not specify a REAL or LREAL data type in safety programs. An error occurs when the REAL or LREAL data type is specified.

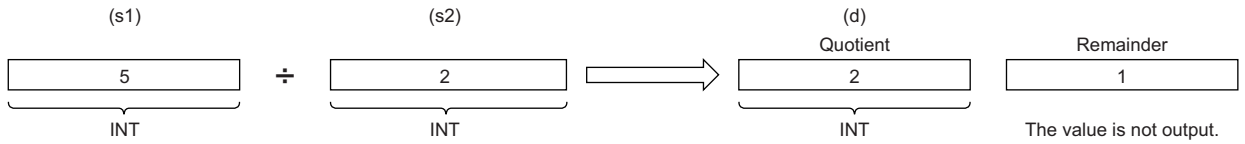
Processing details

■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
3400H	The value (divisor) input to (s2) is 0.

- When (s1) and (s2) are of DINT or DWORD data type

Error code (SD0)	Description
3400H	The value (divisor) input to (s2) is 0.

- When (s1) and (s2) are of REAL data type

Error code (SD0)	Description
3400H	The value (divisor) input to (s2) is 0.
3402H	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 ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{128}$

- When (s1) and (s2) are of LREAL data type

Error code (SD0)	Description
3400H	The value (divisor) input to (s2) is 0.
3402H	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 ±∞.
3403H	The data output from (d) exceeds the following range. (An overflow has occurred.) $\lvert(d)\rvert<2^{1024}$

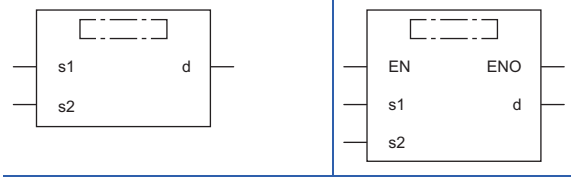
34.5 Remainder

MOD(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the remainder of input values ((s1)÷(s2)).

Ladder, FBD/LD		Structured text
[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO] The function is described as an operator. (📖 MELSEC iQ-R Programming Manual (Program Design)) [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

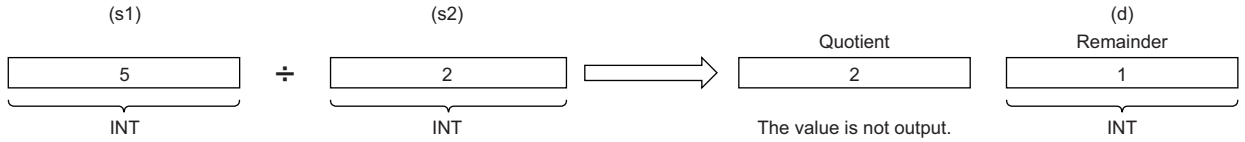
Processing details

■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
3400H	The value (divisor) input to (s2) is 0.

- When (s1) and (s2) are of DINT or DWORD data type

Error code (SD0)	Description
3400H	The value (divisor) input to (s2) is 0.

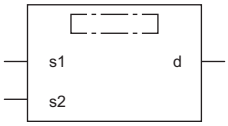
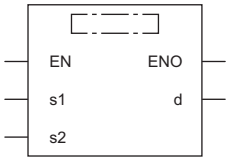
34.6 Exponentiation

EXPT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions output the exponentiation of an input value.

Ladder, FBD/LD		Structured text

[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

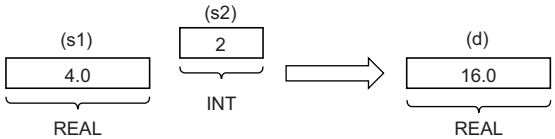
■Description, type, data type

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
3402H	The value input to (s1) or (s2) is -0, a subnormal number, NaN (not a number), or ±∞.

34.7 Assignment (Move Operation)

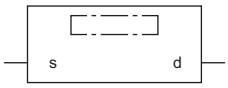
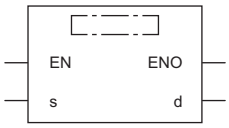
MOVE(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the assignment value of an input value.

Ladder, FBD/LD		Structured text

[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

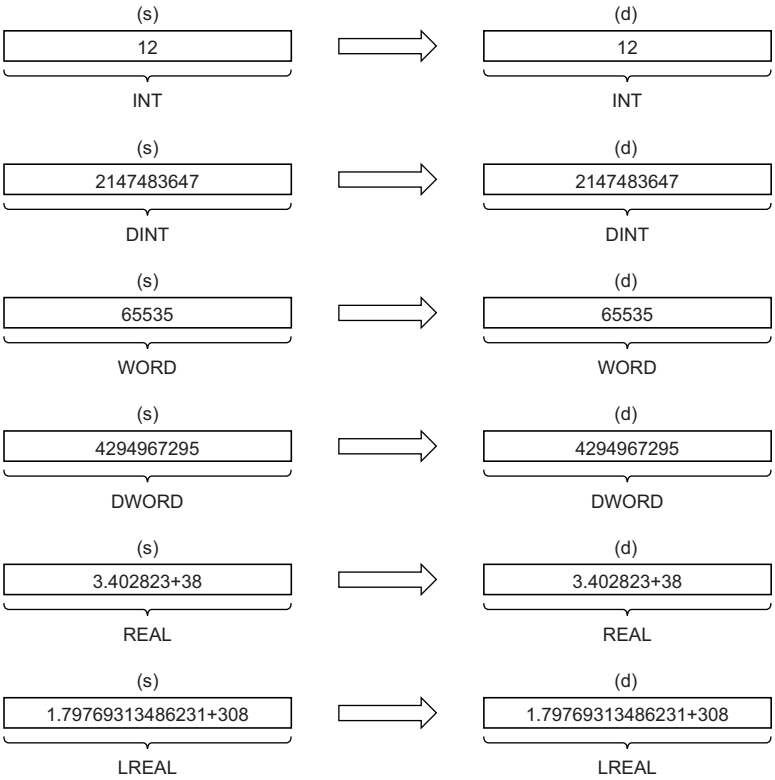
Argument	Description	Type	Data type
EN	Execution condition (TRUE: Executed, FALSE: Not executed)	Input variable	BOOL
s (IN)	Input	Input variable	ANY*1
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY*1

*1 Do not specify a REAL, LREAL, or STRING data type in safety programs. An error occurs when the REAL, LREAL, or STRING data type is specified.

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 the setting area specified by (s) in the device/label memory.
3405H	The number of characters in the string input to (s) exceeds 16383.
3406H	The entire string cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

35 BIT SHIFT FUNCTIONS

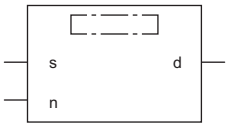
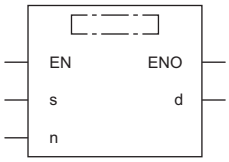
35.1 Shifting Data to the Left by n Bit(s)

SHL(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions shift the input value to the left by (n) bit(s), and output the operation result.

Ladder, FBD/LD		Structured text

[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

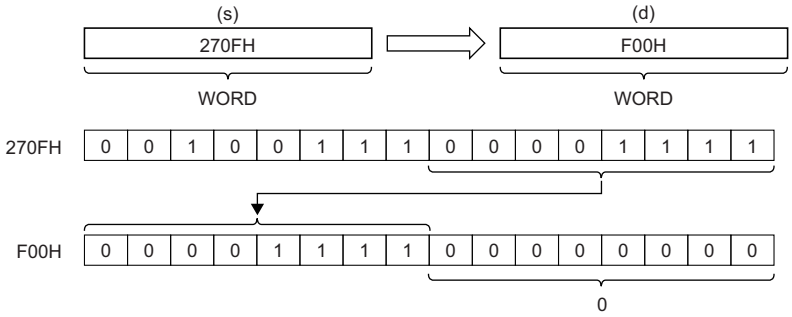
Processing details

■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.

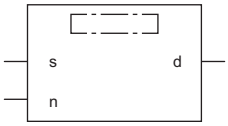
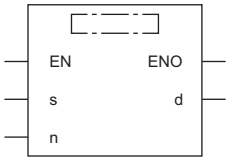
35.2 Shifting Data to the Right by n Bit(s)

SHR(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions shift the input value to the right by (n) bit(s), and output the operation result.

Ladder, FBD/LD		Structured text

[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

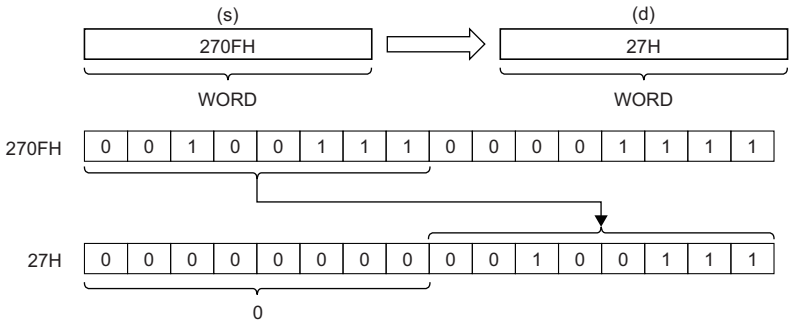
Processing details

■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.

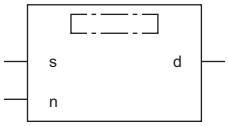
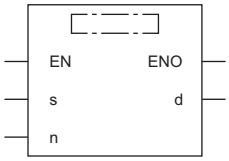
35.3 Rotating Data to the Left by n Bit(s)

ROL(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions rotate the input value to the left by (n) bit(s), and output the operation result.

Ladder, FBD/LD		Structured text

[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

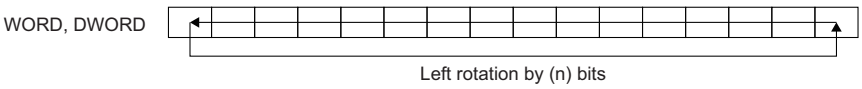
■Description, type, data type

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

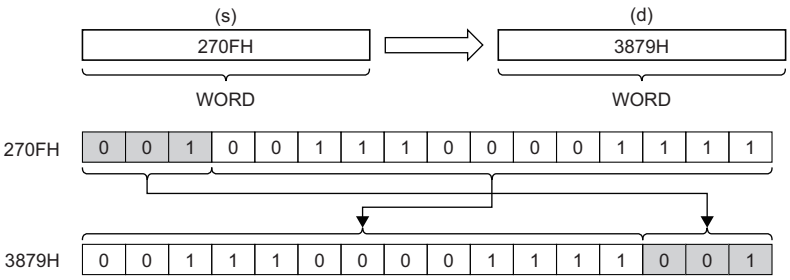
■Operation processing

- 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.

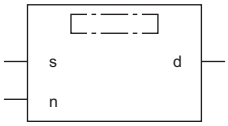
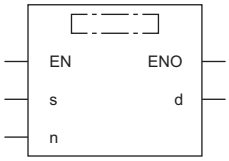
35.4 Rotating Data to the Right by n Bit(s)

ROR(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions rotate the input value to the right by (n) bit(s), and output the operation result.

Ladder, FBD/LD		Structured text

[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

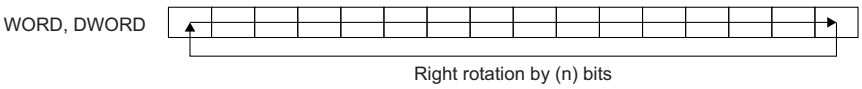
■Description, type, data type

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

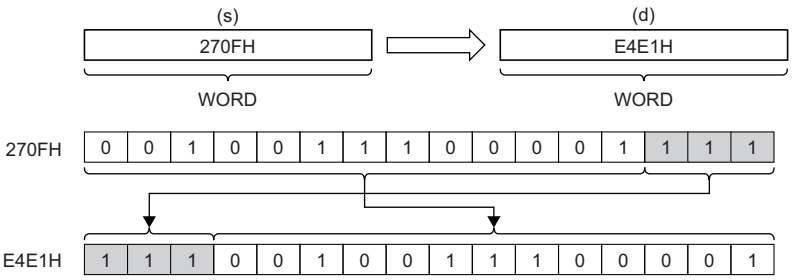
■Operation processing

- 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.

36 BOOLEAN FUNCTIONS

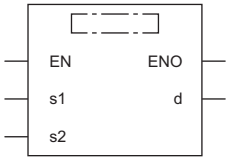
36.1 AND Operation, OR Operation, and XOR Operation

AND(_E), OR(_E), XOR(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

- 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		Structured text*1

[Without EN/ENO]
 	
	[With EN/ENO]
	 
	
		[Without EN/ENO]
		The function is described as an operator. (📖 MELSEC iQ-R Programming Manual (Program Design))
		[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

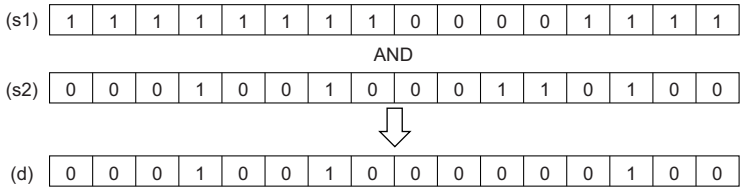
■Operation processing

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

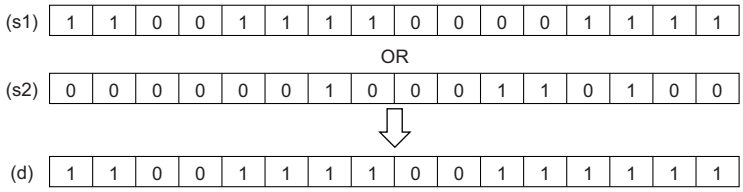


2. OR(_E)

- 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

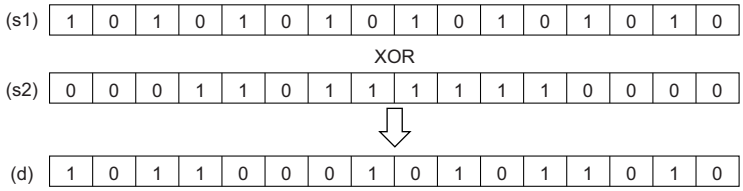


3. XOR(_E)

- 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.

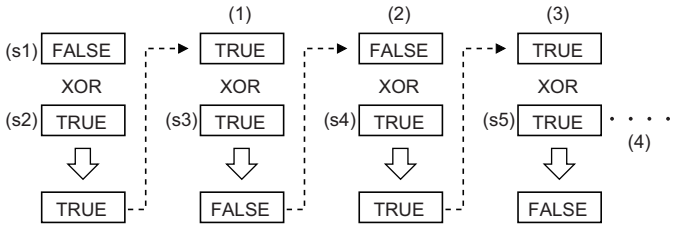
Data type: WORD



- 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

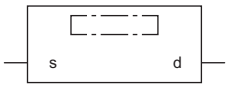
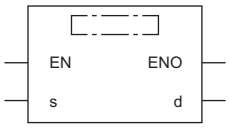
There is no operation error.

36.2 NOT Operation

NOT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the logical NOT of input values.

Ladder, FBD/LD		Structured text
[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO] The function is described as an operator. (📖 MELSEC iQ-R Programming Manual (Program Design)) [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*2
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_BIT*2

*1 DX cannot be used.
*2 Do not specify a BOOL data type in safety programs. An error occurs when the BOOL data type is specified.

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.

37 SELECTION FUNCTIONS

37.1 Selecting a Value

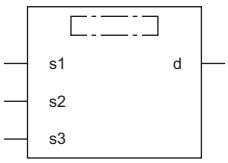
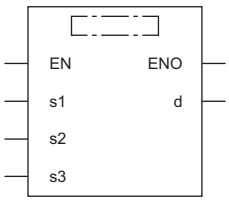
SEL(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the selected input value.

Ladder, FBD/LD		Structured text

[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*1
s3 (IN1)			
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY*1

*1 Do not specify a REAL, LREAL, or STRING data type in safety programs. An error occurs when the REAL, LREAL, or STRING data type is specified.

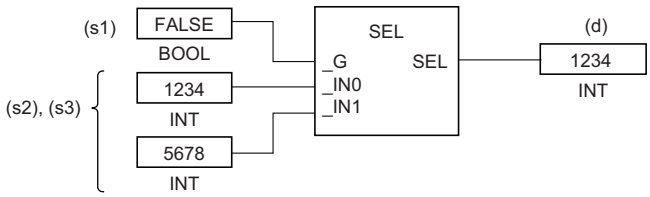
Processing details

■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).
3406H	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.)

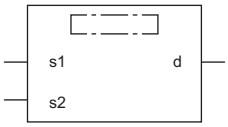
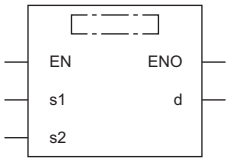
37.2 Selecting the Maximum/Minimum Value

MAX(_E), MIN(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

- MAX(_E): These functions output the maximum input value.
- MIN(_E): These functions output the minimum input value.

Ladder, FBD/LD*1		Structured text

[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*1
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY_ELEMENTARY*1

*1 Do not specify a REAL, LREAL, or STRING data type in safety programs. An error occurs when the REAL, LREAL, or STRING data type is specified.

Processing details

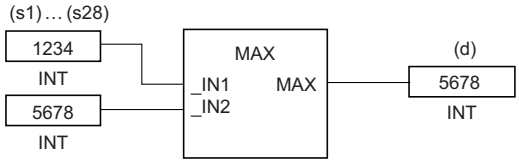
■Operation processing

• 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

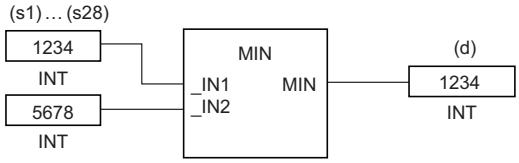


• MIN(_E)

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 specified by (s1) to (s28) in the device/label memory.
3405H	The number of characters in the strings input to (s1) to (s28) exceeds 16383.
3406H	The entire string cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

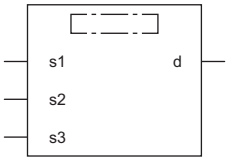
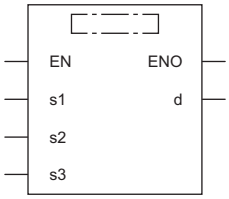
37.3 Controlling the Upper/Lower Limit

LIMIT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions output an input value that has been controlled in terms of the upper and lower limits.

Ladder, FBD/LD		Structured text

[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.

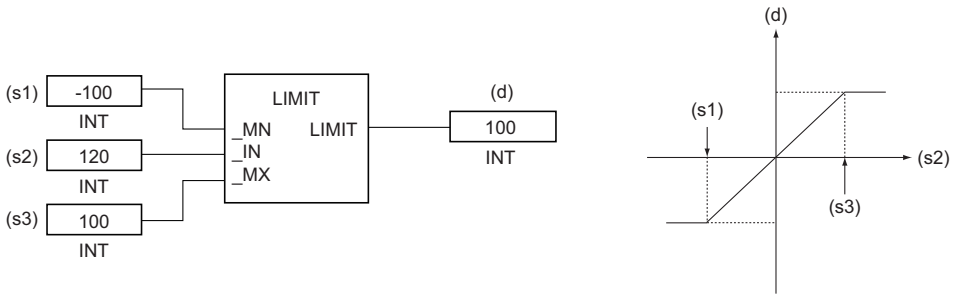
Processing details

■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
3405H	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
3405H	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
3405H	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
3402H	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 ±∞.
3405H	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
3402H	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 ±∞.
3405H	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).
3405H	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.
3406H	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.)

37.4 Multiplexer

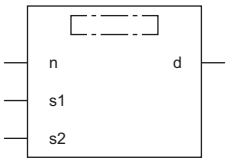
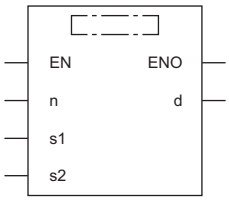
MUX(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output one of the input values.

Ladder, FBD/LD[*1]		Structured text[*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[*1]
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	ANY[*1]

*1 Do not specify a REAL, LREAL, or STRING data type in safety programs. An error occurs when the REAL, LREAL, or STRING data type is specified.

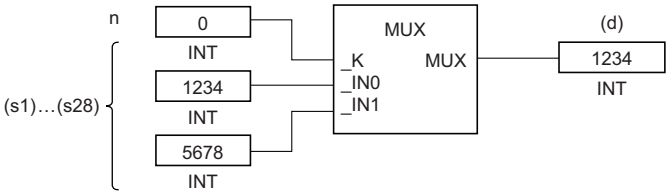
Processing details

■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 specified by (s1) to (s28) in the device/label memory.
3405H	The number of characters in the strings input to (s1) to (s28) exceeds 16383.
3406H	The entire string cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

38 COMPARISON FUNCTIONS

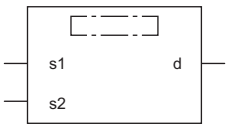
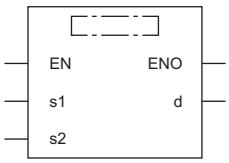
38.1 Comparing Data

GT(_E), GE(_E), EQ(_E), LE(_E), LT(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the comparison result of input values.

Ladder, FBD/LD*1		Structured text*1

[Without EN/ENO]
 
 	
	[With EN/ENO]
	 
	
		[Without EN/ENO]*2
		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.
*2 The engineering tool with version "1.035M" or later supports the ST. The function is described as an operator when an engineering tool with an earlier version is used. (MELSEC iQ-R Programming Manual (Program Design))

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*2
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output (TRUE, FALSE)	Output variable	BOOL

*1 DX cannot be used.
*2 Do not specify a REAL, LREAL, or STRING data type in safety programs. An error occurs when the REAL, LREAL, or STRING data type is specified.

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 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 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 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 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 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 specified by (s1) to (s28) in the device/label memory.
3405H	The number of characters in the strings input to (s1) to (s28) exceeds 16383.
3406H	The entire string cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

38.2 Comparing Data

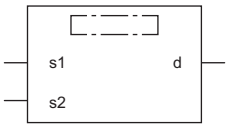
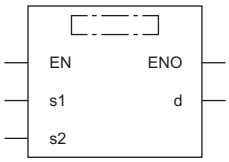
NE(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the comparison result of input values.

Ladder, FBD/LD		Structured text

[Without EN/ENO]	[With EN/ENO]	[Without EN/ENO][*1]
		d:=NE(s1,s2);
		[With EN/ENO]
		d:=NE_E(EN,ENO,s1,s2);

*1 The engineering tool with version "1.035M" or later supports the ST. The function is described as an operator when an engineering tool with an earlier version is used. (📖 MELSEC iQ-R Programming Manual (Program Design))

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[*1]
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output (TRUE, FALSE)	Output variable	BOOL

*1 Do not specify a REAL, LREAL, or STRING data type in safety programs. An error occurs when the REAL, LREAL, or STRING data type is specified.

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 the setting area specified by (s) in the device/label memory.
3405H	The number of characters in the string input to (s) exceeds 16383.
3406H	The entire string cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

39 STRING FUNCTIONS

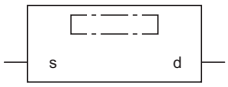
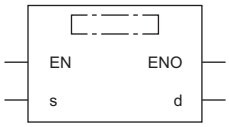
39.1 Detecting a String Length

LEN(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions detect and output the length of the string input.

Ladder, FBD/LD		Structured text

[Without EN/ENO]
 
 	
	[With EN/ENO]
	 
	
		[Without EN/ENO]
		d:=LEN(s);
		[With EN/ENO]
		d:=LEN_E(EN,ENO,s);

Setting data

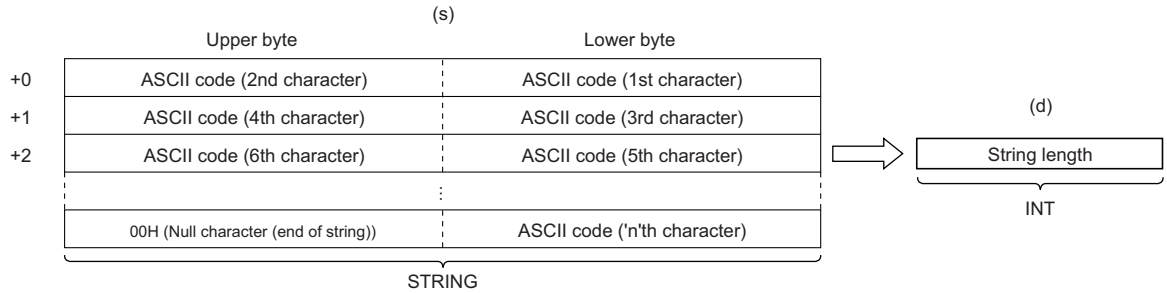
■Description, type, data type

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

There is no operation error.

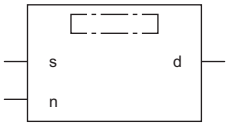
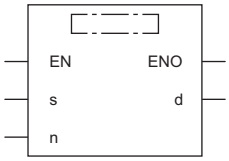
39.2 Extracting String Data From the Left/Right

LEFT(_E), RIGHT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- 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		Structured text

[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

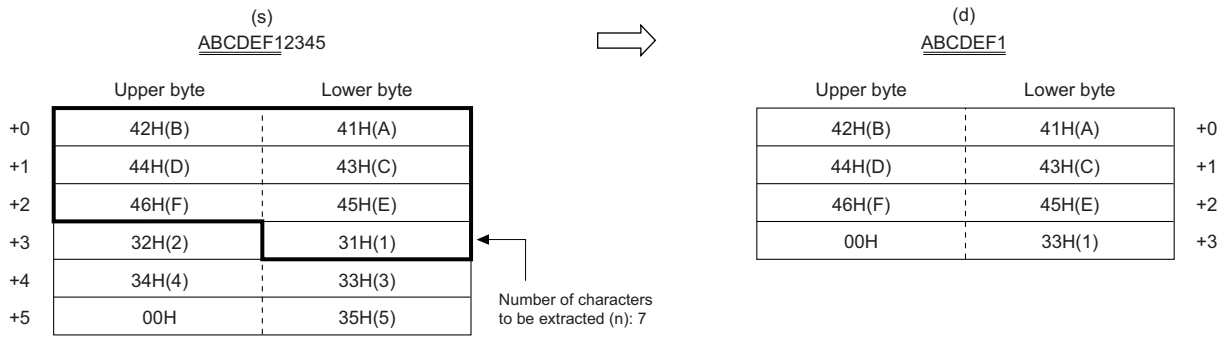
- LEFT(_E)

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



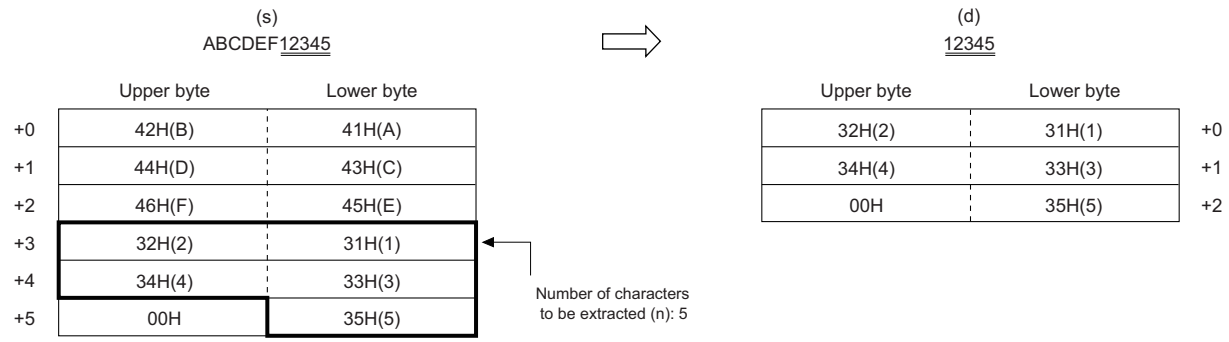
- RIGHT(_E)

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

There is no operation error.

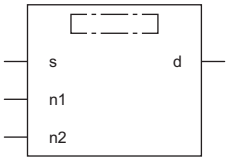
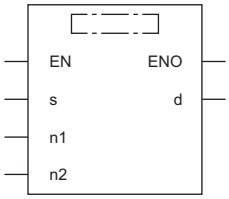
39.3 Extracting String Data

MID(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions extract and output the specified number of characters, starting from the specified position of the string input.

Ladder, FBD/LD		Structured text

[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

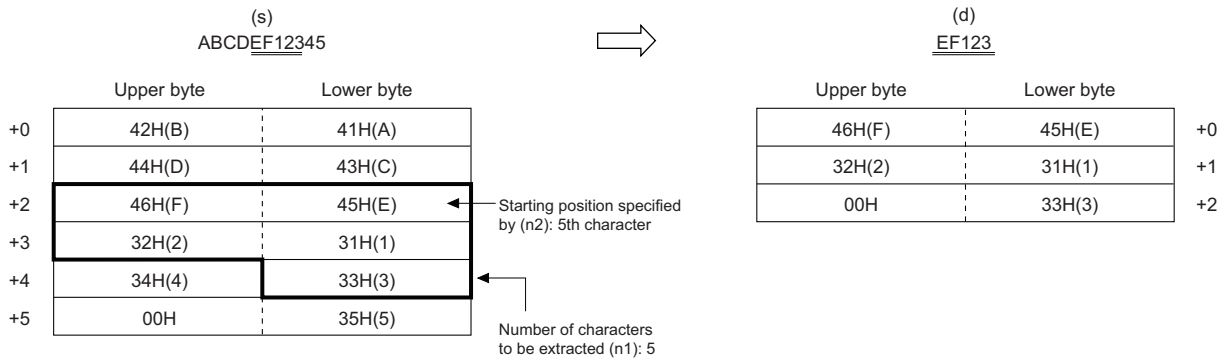
Processing details

■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 the setting area specified by (s) in the device/label memory.
3405H	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).

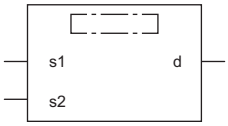
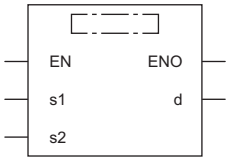
39.4 Concatenating String Data

CONCAT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions concatenate character strings, and output the operation result.

Ladder, FBD/LD*1		Structured text*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. Note that in structured text languages, for constructs other than functions, if a word device is specified as the return value and the following option setting is enabled, the substring up to the 2047th byte will be output.

[Convert] ⇨ [Basic Setting] ⇨ [Operational Setting] ⇨ [ST Program] ⇨ [Use Large Size CONCAT]

(s1) ABCDE + (s2) 123456 ⇨ (d) ABCDE123456

	Upper byte	Lower byte
+0	42H(B)	41H(A)
+1	44H(D)	43H(C)
+2	00H	45H(E)

	Upper byte	Lower byte
+0	32H(2)	31H(1)
+1	34H(4)	33H(3)
+2	36H(6)	35H(5)
+3	00H	

	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. Note that in structured text languages, for constructs other than functions, if a word device is specified as the argument and the following option setting is enabled, the range is 0 to 2047 bytes.

[Convert] ⇨ [Basic Setting] ⇨ [Operational Setting] ⇨ [ST Program] ⇨ [Use Large Size CONCAT]

■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 specified by (s1) to (s28) in the device/label memory.
	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).
3405H	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 character string in the device specified by (d) exceeds 16383.
3406H	The entire string after concatenate processing cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

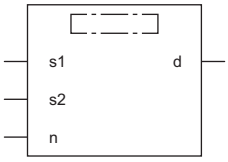
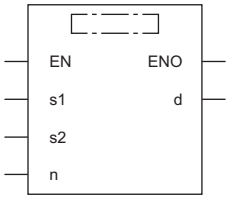
39.5 Inserting String Data

INSERT(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions insert a character string into another string, and output the operation result.

Ladder, FBD/LD		Structured text

[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

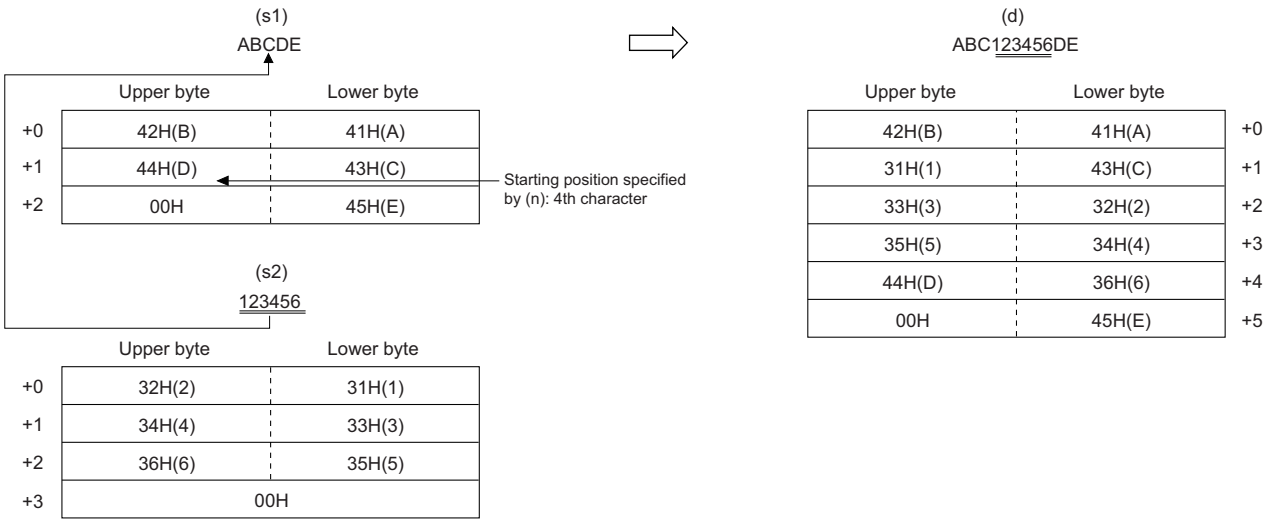
Processing details

■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 specified by (s1) to (s28) in the device/label memory.
	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).
3405H	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 character string in the device specified by (d) exceeds 16383.
3406H	The entire string after concatenate processing cannot be stored in the setting area specified by (d) in the device/label memory. (The number of required points is insufficient.)

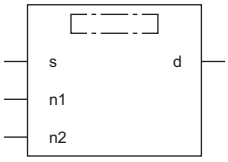
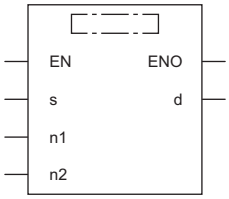
39.6 Deleting String Data

DELETE(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions delete the specified range in a character string, and output the operation result.

Ladder, FBD/LD		Structured text

[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

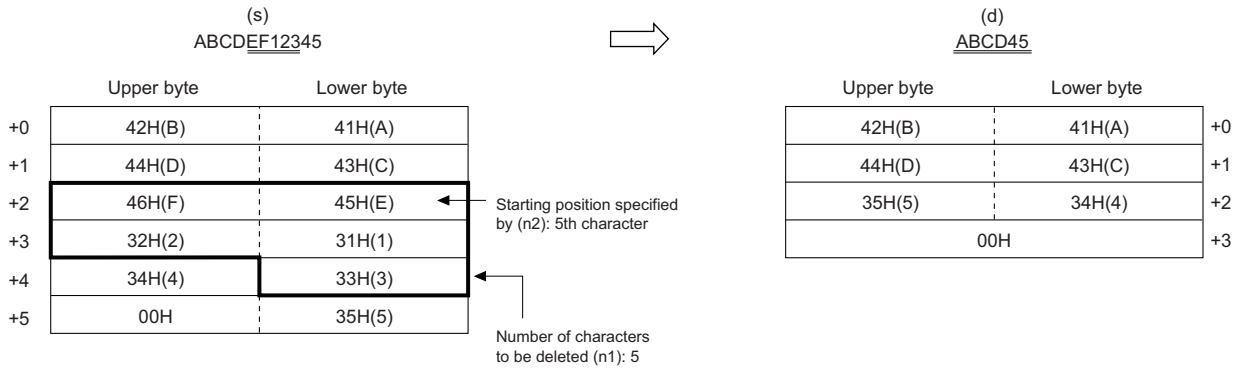
Processing details

■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).
3405H	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).
3406H	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).

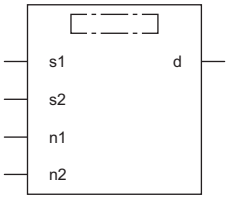
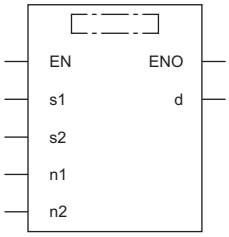
39.7 Replacing String Data

REPLACE(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions replace the specified range in a character string, and output the operation result.

Ladder, FBD/LD		Structured text

[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

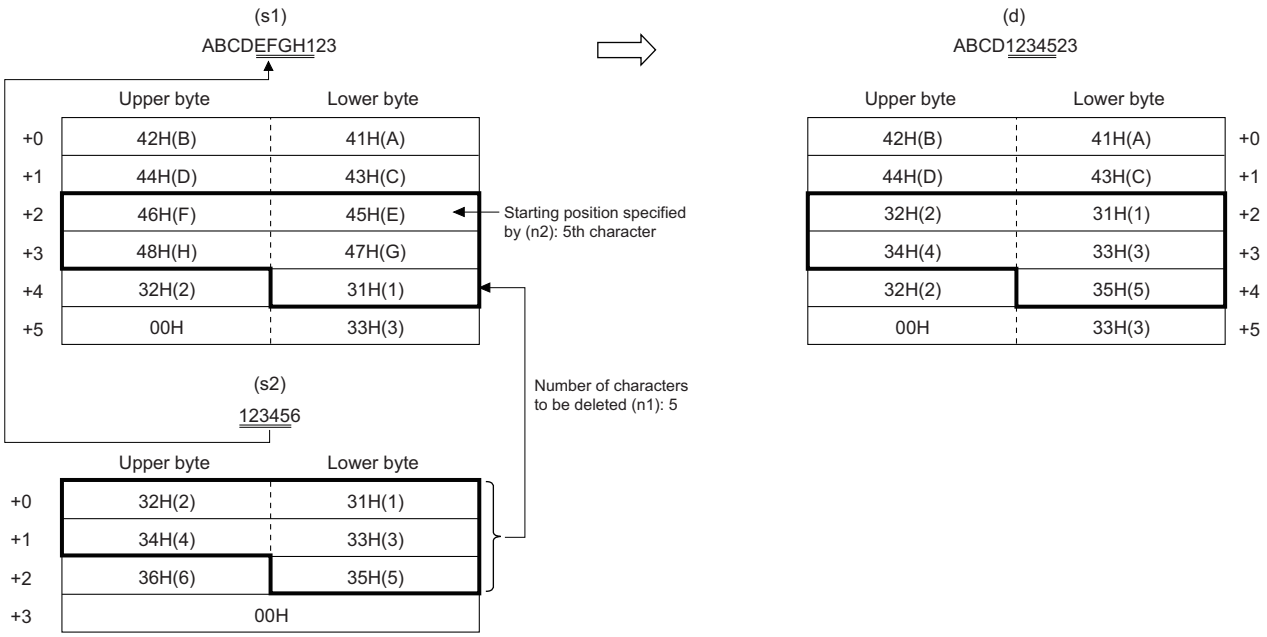
Processing details

■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).
3405H	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).
3406H	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).

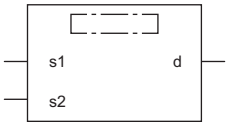
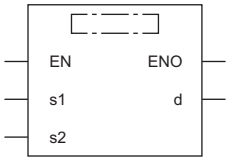
39.8 Searching String Data

FIND(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These functions search a character string, and output the operation result.

Ladder, FBD/LD		Structured text

[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

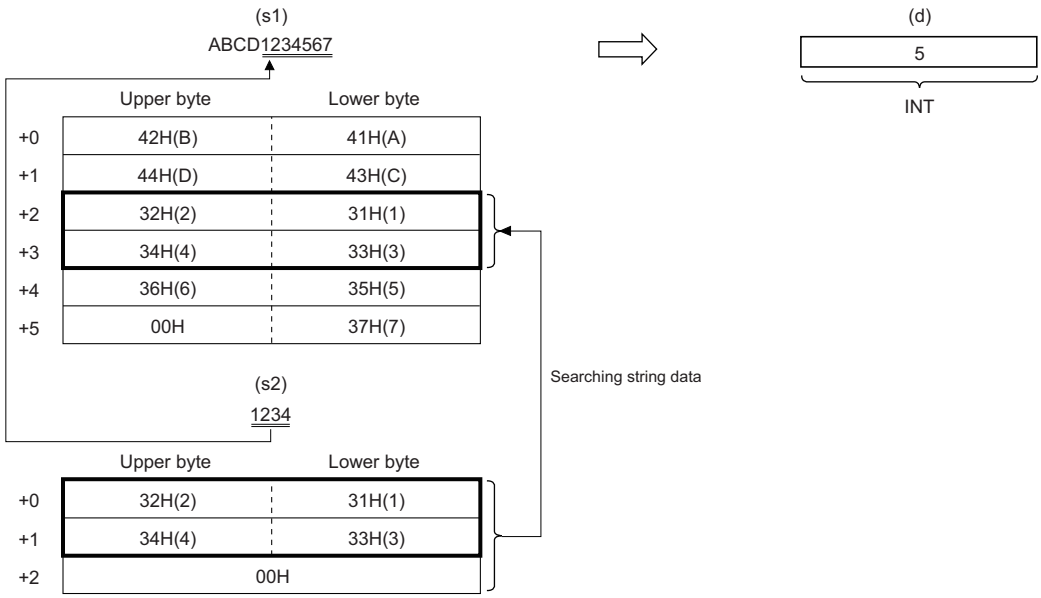
■Description, type, data type

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

There is no operation error.

40 TIME DATA TYPE FUNCTIONS

40.1 Addition

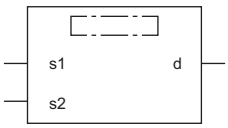
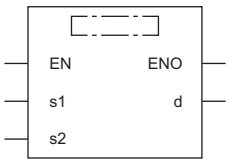
ADD_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the sum ((s1)+(s2)) of the TIME data type input values.

Ladder, FBD/LD		Structured text

[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

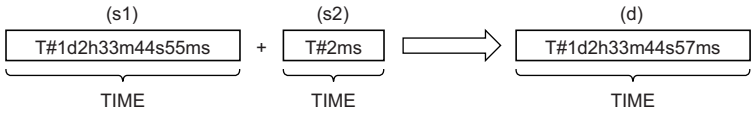
Processing details

■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.

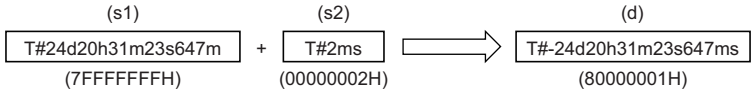
When (s1)=T#1d2h33m44s55ms (1 day, 2 hours, 33 minutes, 44 seconds, 55 milliseconds) and (s2)=T#2ms (2 milliseconds)



- 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.

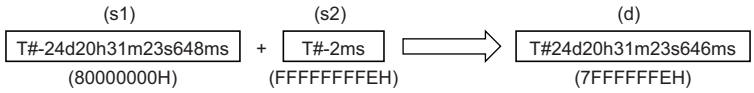
Overflow



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.

40.2 Subtraction

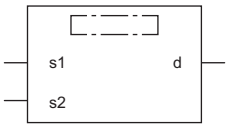
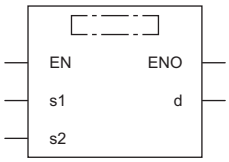
SUB_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the difference ((s1)-(s2)) between the TIME data type input values.

Ladder, FBD/LD		Structured text

[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).

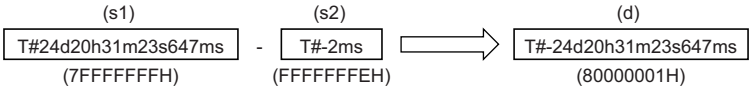
Ex.

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.

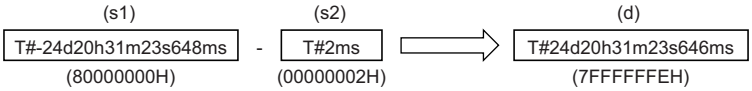
Overflow



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.

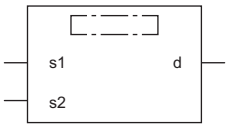
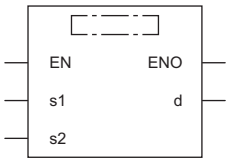
40.3 Multiplication

MUL_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the product ((s1)×(s2)) of the TIME data type input values.

Ladder, FBD/LD		Structured text

[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*1
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

*1 Do not specify an INT, REAL, or LREAL data type in safety programs. An error occurs when the INT, REAL, or LREAL data type is specified.

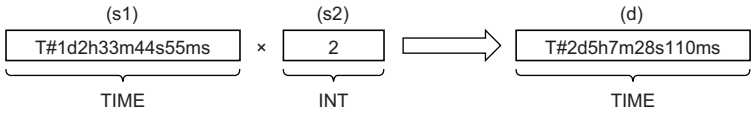
Processing details

■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.

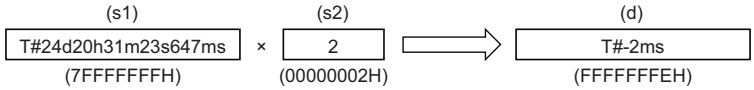
When (s1)=T#1d2h33m44s55ms (1 day, 2 hours, 33 minutes, 44 seconds, 55 milliseconds) and (s2)=2



- 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.

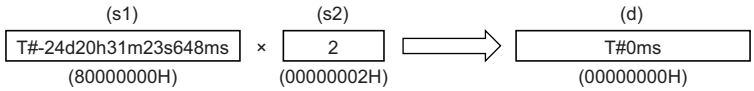
Overflow



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
3402H	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 ±∞.
3405H	The single-precision real number input to (s2) is out of the range, -2147483648 to 2147483647.

40.4 Division

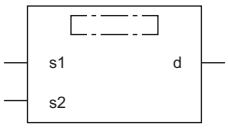
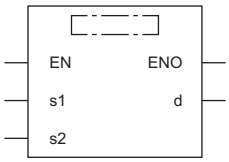
DIV_TIME(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) RnPSFCPU (Safety) RnSFCPU (Standard) RnSFCPU (Safety)

These functions output the quotient ((s1)÷(s2)) of the TIME data type input values.

Ladder, FBD/LD		Structured text

[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*1
ENO	Output status (TRUE: Normal, FALSE: Abnormal)	Output variable	BOOL
d	Output	Output variable	TIME

*1 Do not specify an INT, REAL, or LREAL data type in safety programs. An error occurs when the INT, REAL, or LREAL data type is specified.

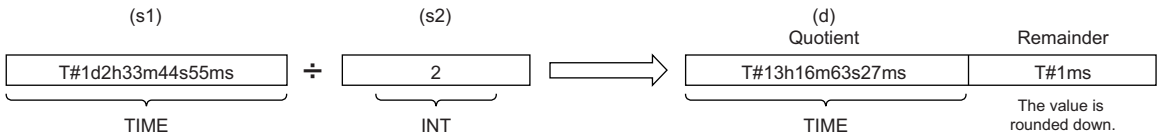
Processing details

■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
3400H	The value input to (s2) is 0. (The value is divided by zero.)

- When (s2) is of LREAL data type

Error code (SD0)	Description
3402H	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)
3405H	The data input to (s2) is out of the range, -2147483648 to 2147483647.

PART 8 STANDARD FUNCTION BLOCKS

This part consists of the following chapters.

41 BISTABLE FUNCTION BLOCKS

42 EDGE DETECTION FUNCTION BLOCKS

43 COUNTER/TIMER FUNCTION BLOCKS

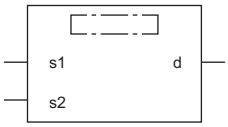
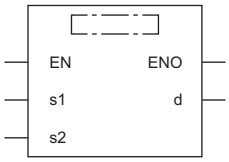
41 BISTABLE FUNCTION BLOCKS

41.1 Bistable Function Block (Set-Dominant)

SR(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These function blocks discriminate between two input values, and output 1 (TRUE) or 0 (FALSE).

Ladder, FBD/LD		Structured text
[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

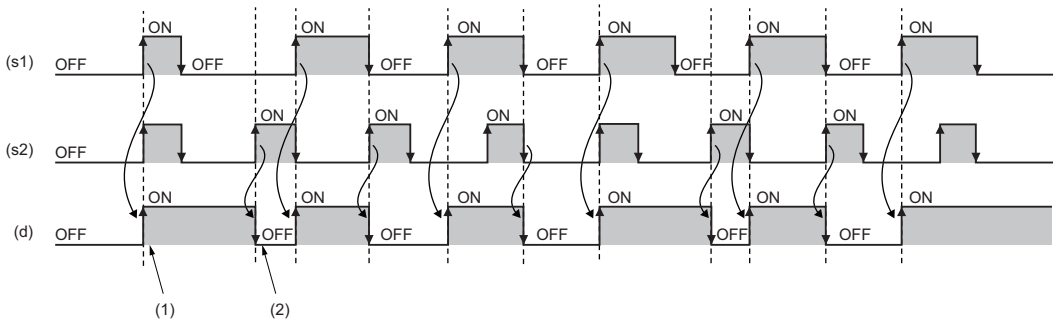
- When (s1) turns on, (d) is set. Turning on (s2) while (s1) is off resets (d).
- Even when (s2) turns on while (s1) is on, (d) is not reset.

■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.

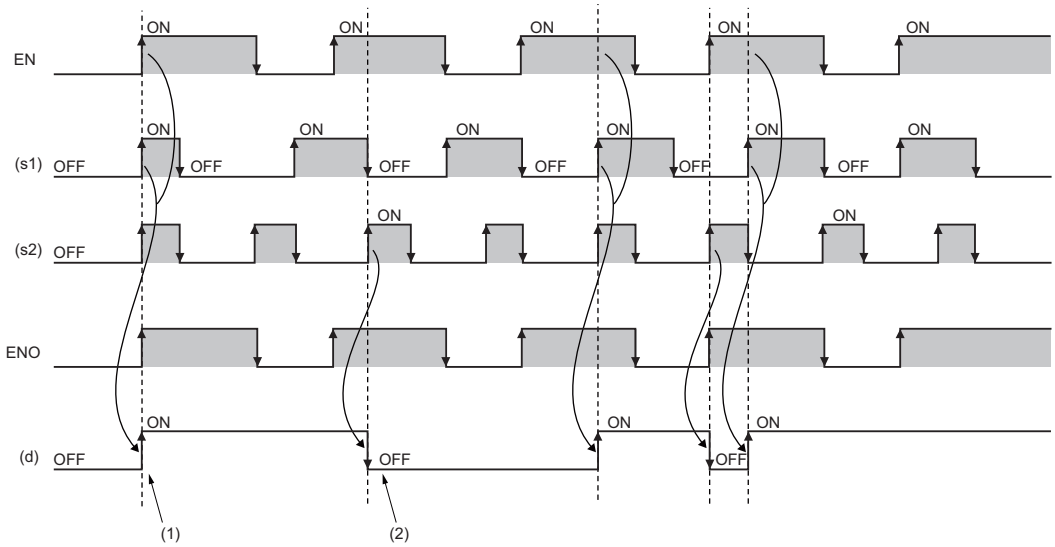
(2) When (s2) turns on while (s1) is off, (d) turns off.

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

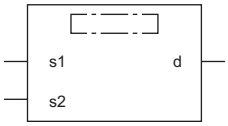
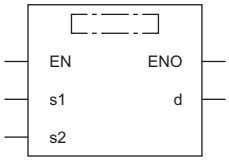
There is no operation error.

41.2 Bistable Function Block (Reset-Dominant)

RS(_E)

RnCPU RnENCPU RnPCPU (Process) RnPCPU (Redundant) RnPSFCPU (Standard) ~~RnPSFCPU (Safety)~~ RnSFCPU (Standard) ~~RnSFCPU (Safety)~~

These function blocks discriminate between two input values, and output 1 (TRUE) or 0 (FALSE).

Ladder, FBD/LD		Structured text
[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

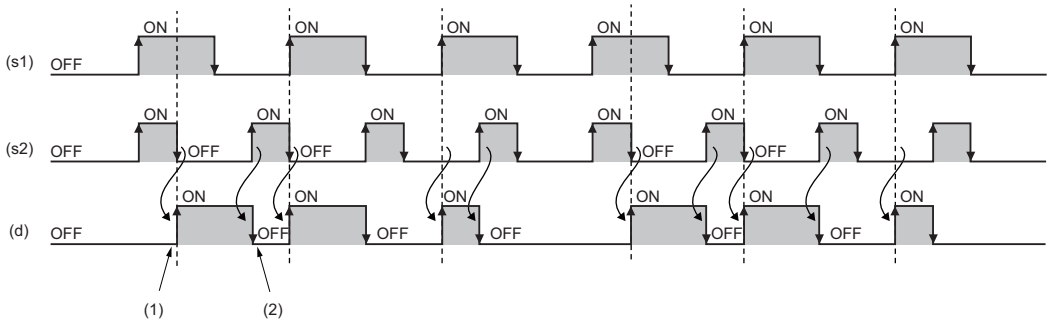
- When (s1) turns on, (d) is set. When (s2) turns on, (d) is reset.
- Even when (s1) turns on while (s2) is on, (d) is not set.

■Operation result

1. Function block without EN/ENO

The operation processing is performed. The operation result is output from (d).

- Timing chart



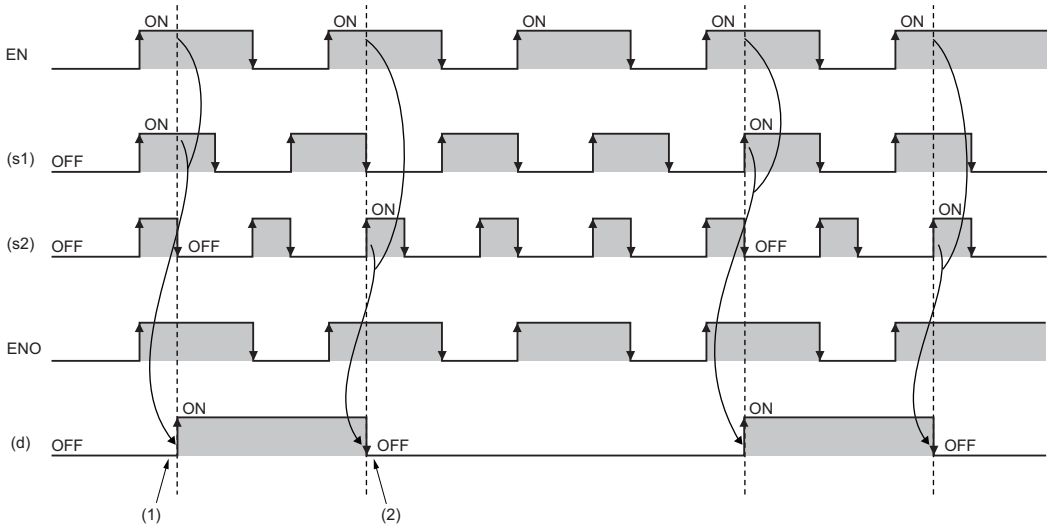
(1) When (s2) turns off while (s1) is on, (d) turns on.
(2) When (s2) turns on, (d) turns off.

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.

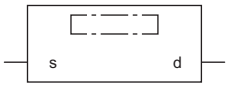
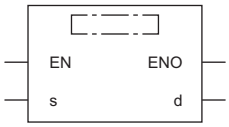
42 EDGE DETECTION FUNCTION BLOCKS

42.1 Detecting a Rising Edge

R_TRIG(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These function blocks detect a signal rising edge, and outputs the pulse signal.

Ladder, FBD/LD		Structured text
[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

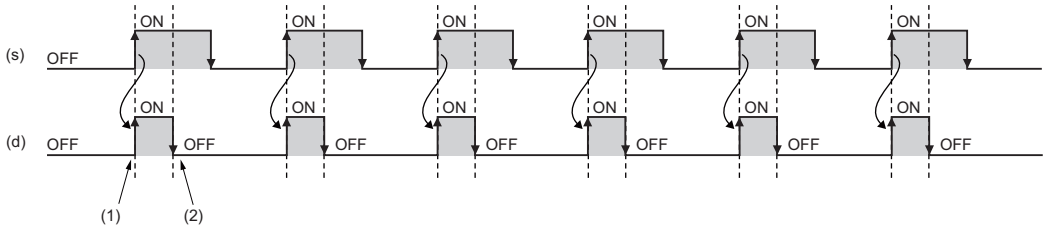
When (s) turns on, (d) turns on only for one scan.

■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).

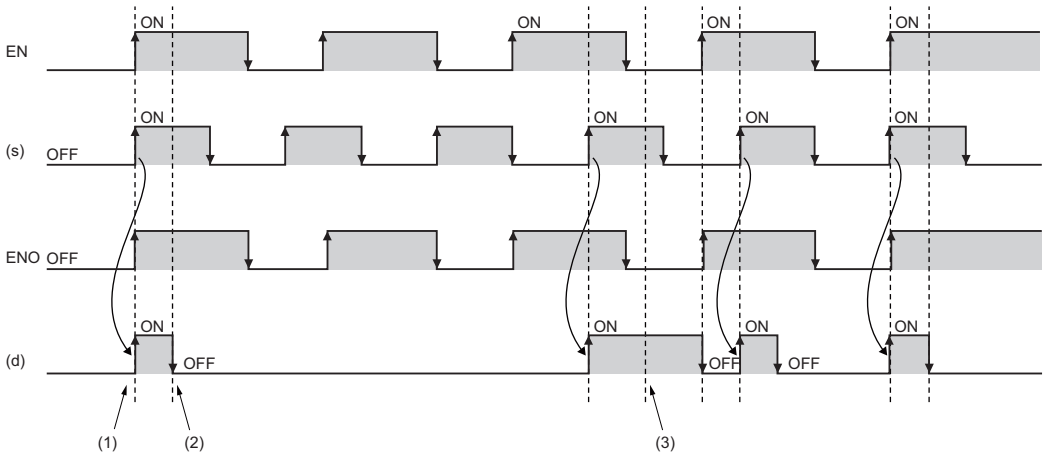
(2) (d) turns off in the next scan.

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

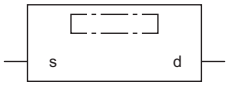
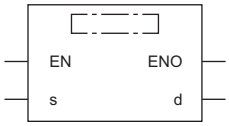
There is no operation error.

42.2 Detecting a Falling Edge

F_TRIG(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These function blocks detect a signal falling edge, and outputs the pulse signal.

Ladder, FBD/LD		Structured text
[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

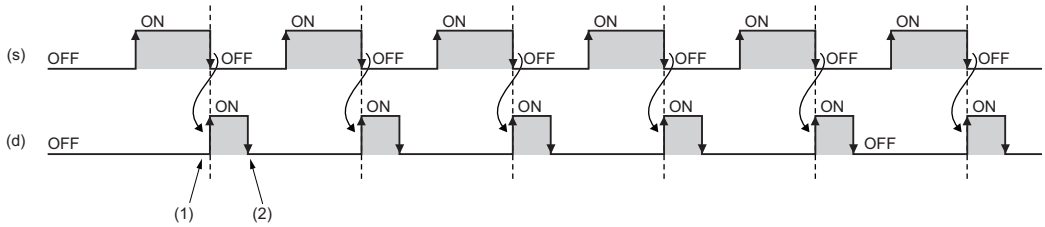
When (s) turns off, (d) turns on only for one scan.

■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).

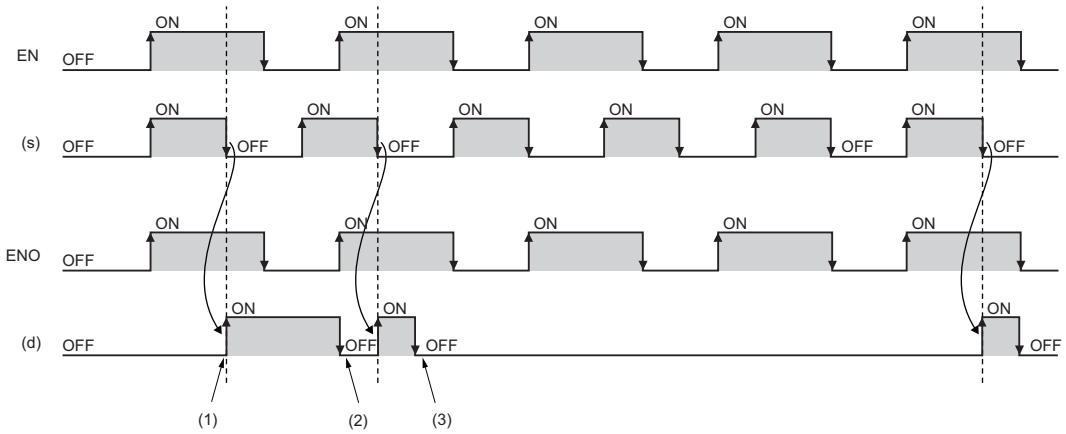
(2) (d) turns off in the next scan.

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.

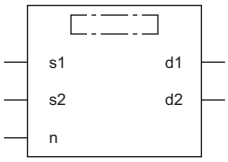
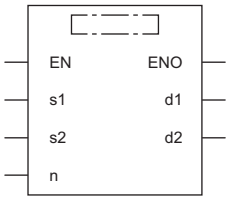
43 COUNTER/TIMER FUNCTION BLOCKS

43.1 Up Counter

CTU(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These function blocks count up the number of rising edges of a signal.

Ladder, FBD/LD		Structured text
[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.

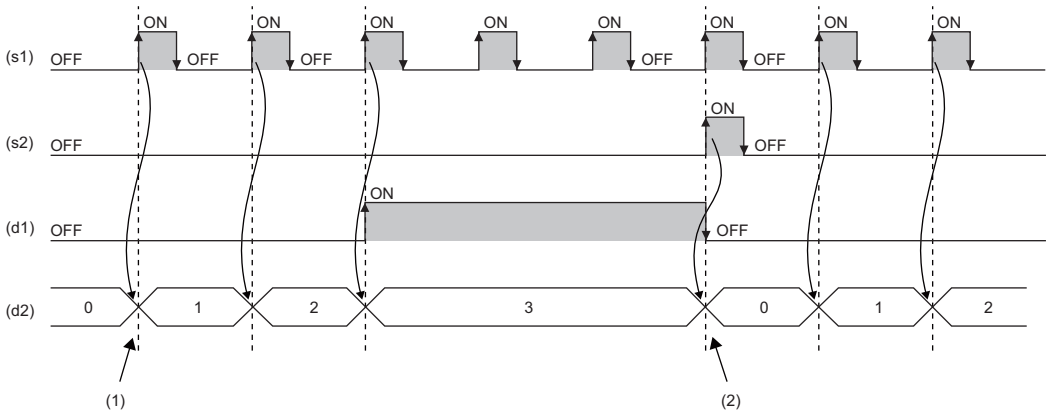
■Operation result

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.

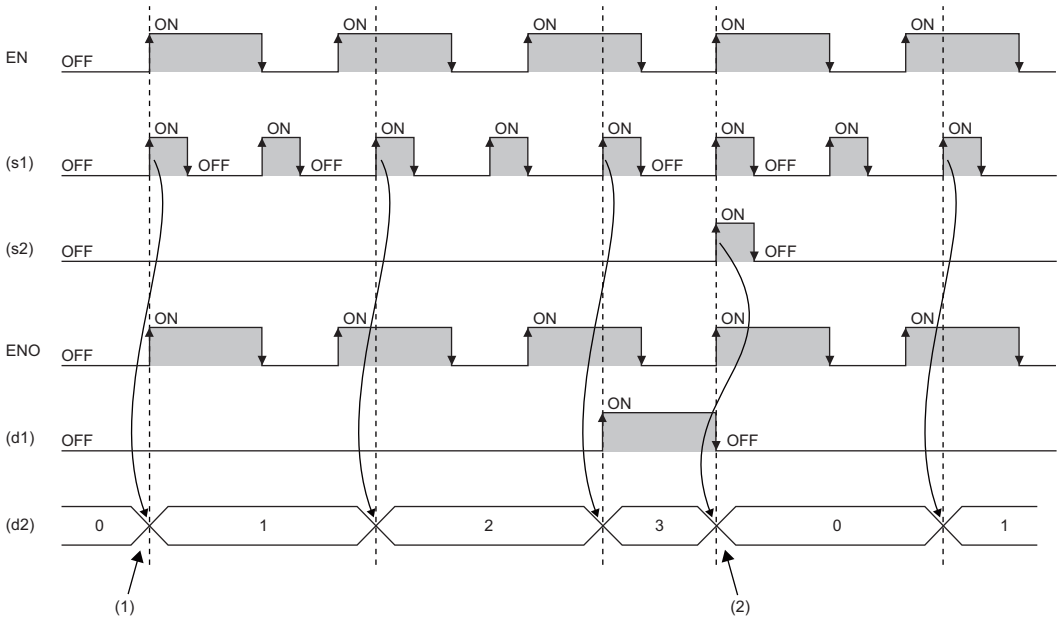
2. Function block with EN/ENO

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

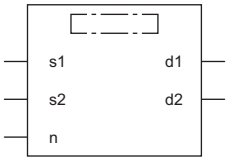
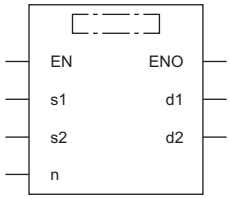
There is no operation error.

43.2 Down Counter

CTD(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These function blocks count down the number of rising edges of a signal.

Ladder, FBD/LD		Structured text
[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.

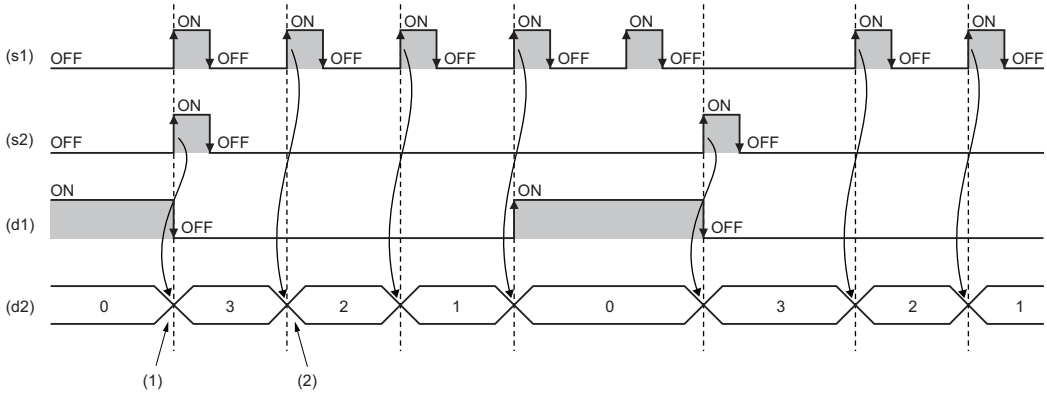
■Operation result

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.

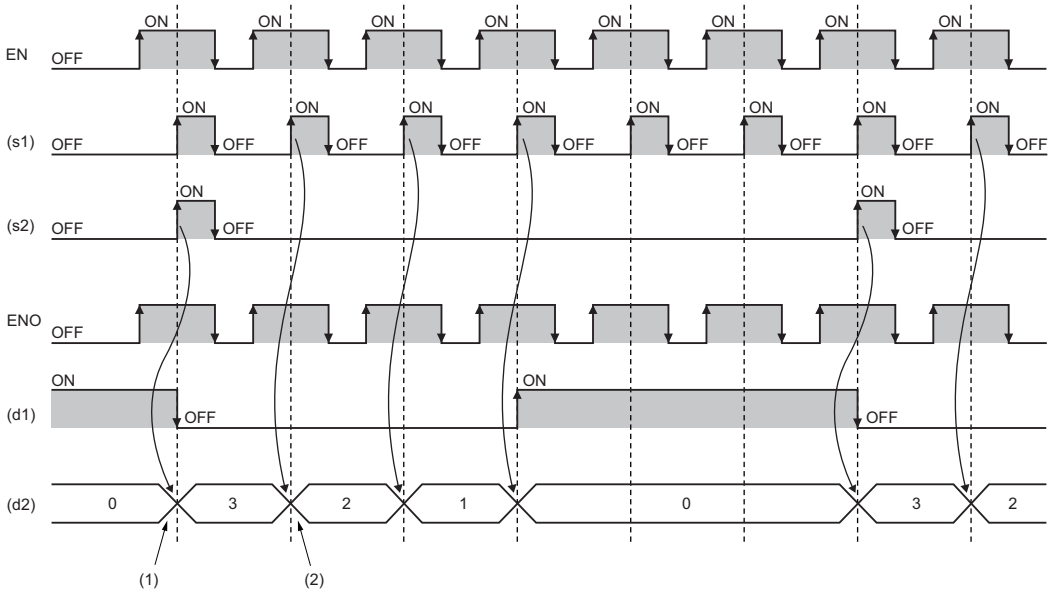
2. Function block with EN/ENO

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

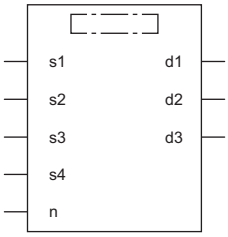
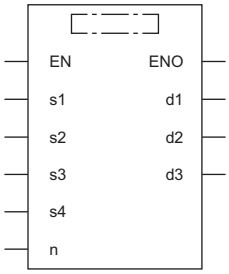
There is no operation error.

43.3 Up/Down Counter

CTUD(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

These function blocks count up or down the number of rising edges of a signal.

Ladder, FBD/LD		Structured text
[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).

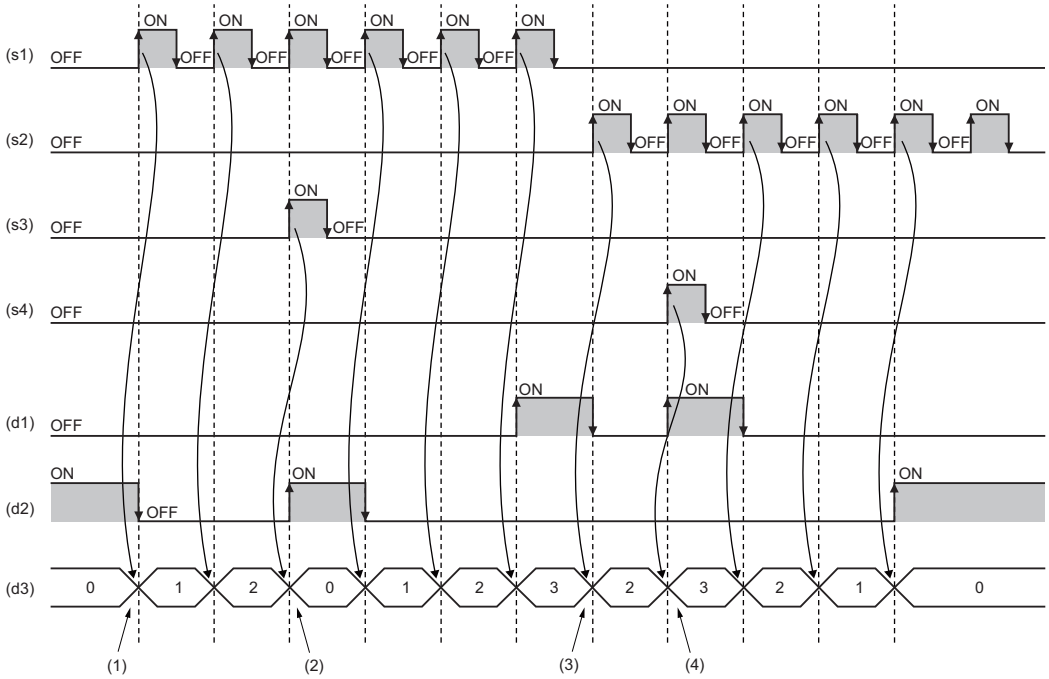
■Operation result

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.

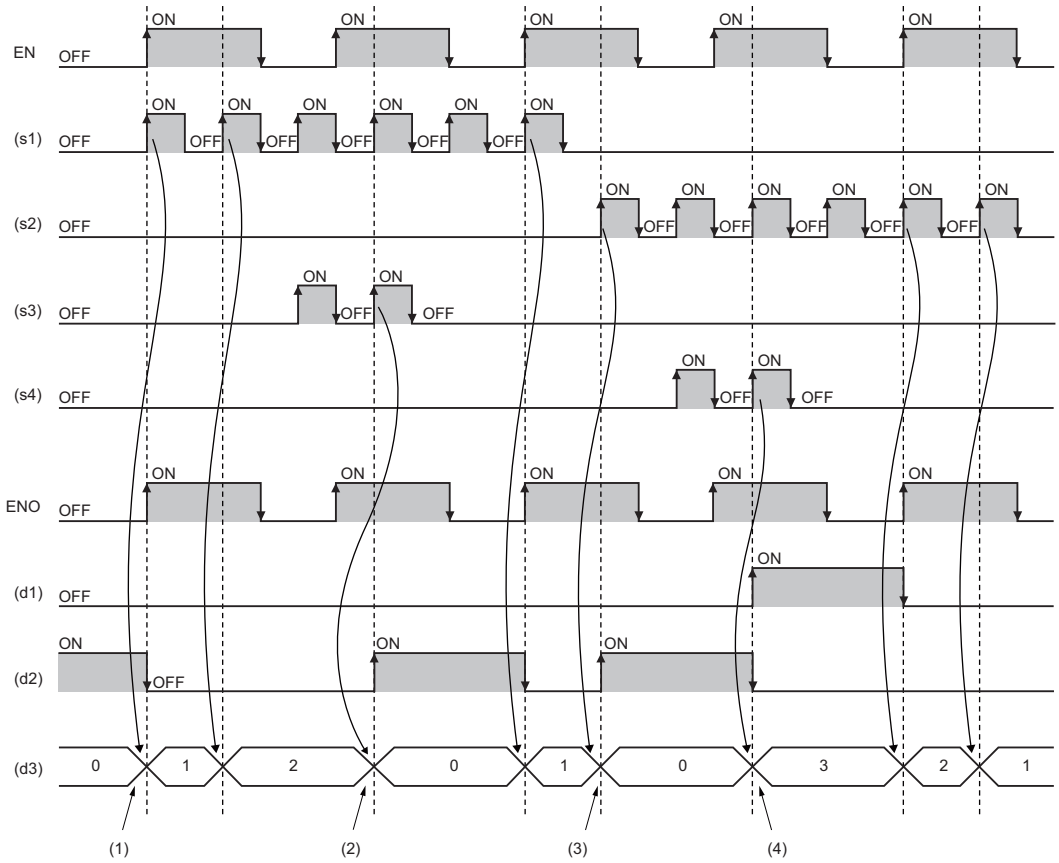
2. Function block with EN/ENO

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

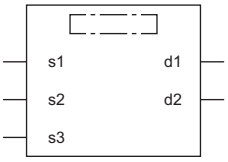
There is no operation error.

43.4 Counter Function Block

COUNTER_FB_M

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

This function block starts counting up when the execution condition is satisfied.

Ladder, FBD/LD	Structured text
	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

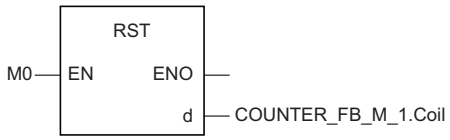
■Operation processing

- 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

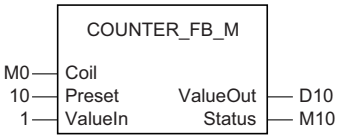
[Ladder program]



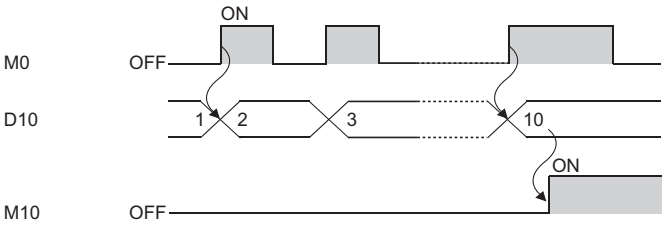
[ST program]

RST(M0, COUNTER_FB_M_1.Coil)

[Ladder example]



[Timing chart]



Operation error

There is no operation error.

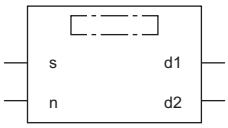
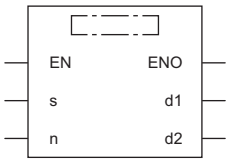
43.5 Pulse Timer

TP(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant) and SIL2 Process CPU] If these function blocks are used in a program executed in both systems, they do not operate in the standby system when the redundant system is in backup mode. (MELSEC iQ-R CPU Module User's Manual (Application))

These function blocks keep the signal on for the specified period of time.

Ladder, FBD/LD		Structured text
[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

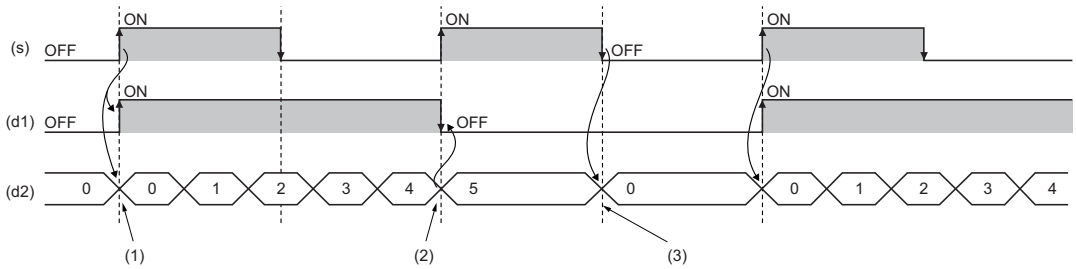
1. Function block without EN/ENO

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.

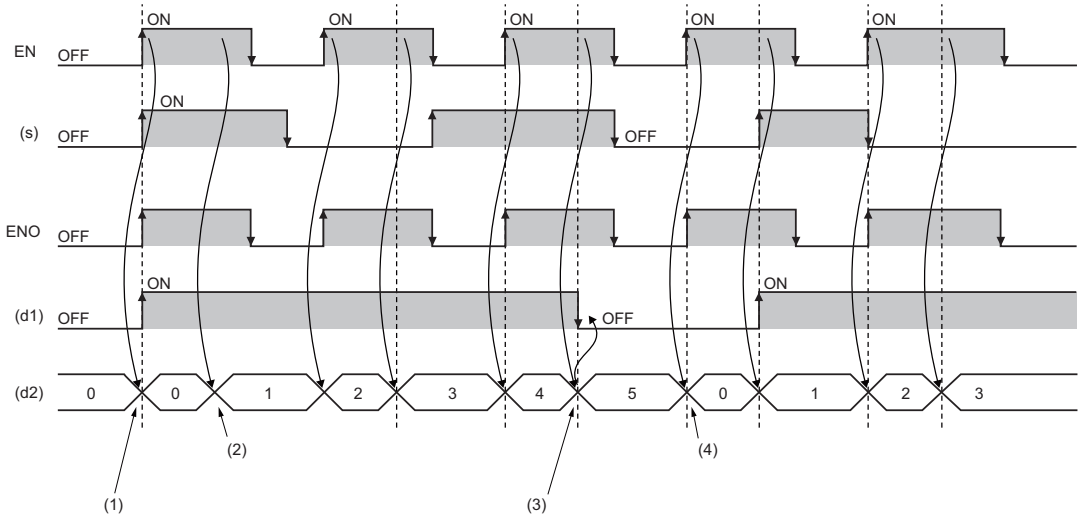
2. Function block with EN/ENO

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
3401H	The output time setting value exceeds the valid range.

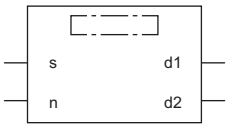
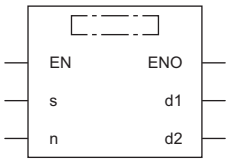
43.6 On Delay Timer

TON(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant) and SIL2 Process CPU] If these function blocks are used in a program executed in both systems, they do not operate in the standby system when the redundant system is in backup mode. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

These function blocks turn on a signal after the specified period of time.

Ladder, FBD/LD		Structured text
[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

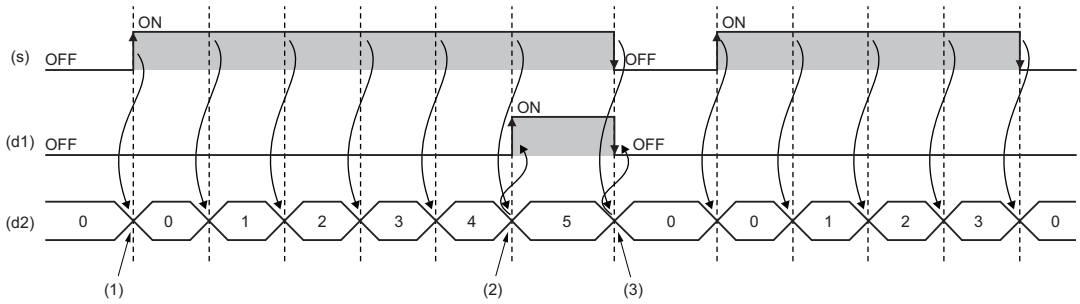
1. Function block without EN/ENO

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.

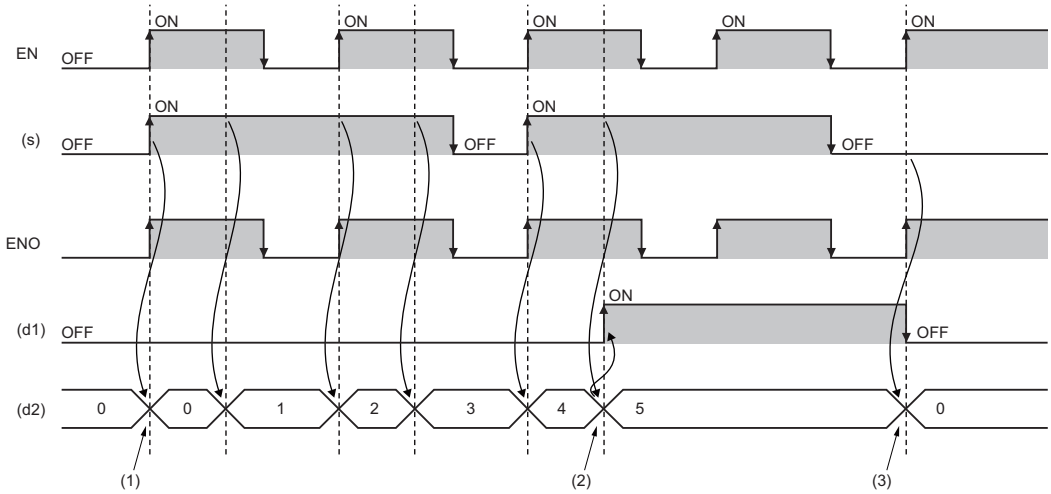
2. Function block with EN/ENO

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
3401H	The output time setting value exceeds the valid range.

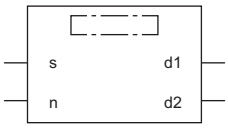
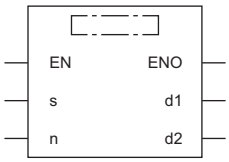
43.7 Off Delay Timer

TOF(_E)

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant) and SIL2 Process CPU] If these function blocks are used in a program executed in both systems, they do not operate in the standby system when the redundant system is in backup mode. (MELSEC iQ-R CPU Module User's Manual (Application))

These function blocks turn off a signal after the specified period of time.

Ladder, FBD/LD		Structured text
[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

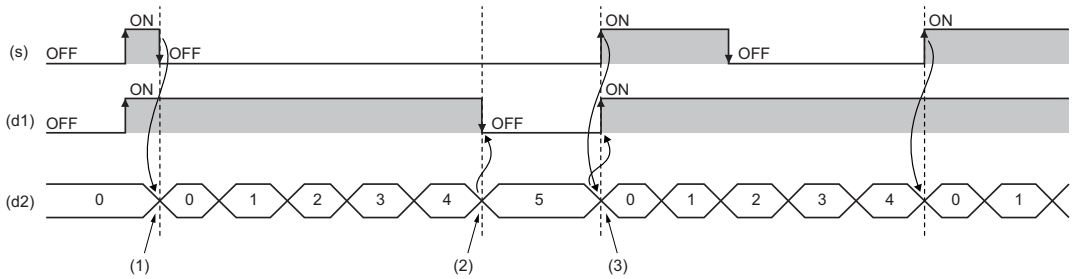
1. Function block without EN/ENO

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.

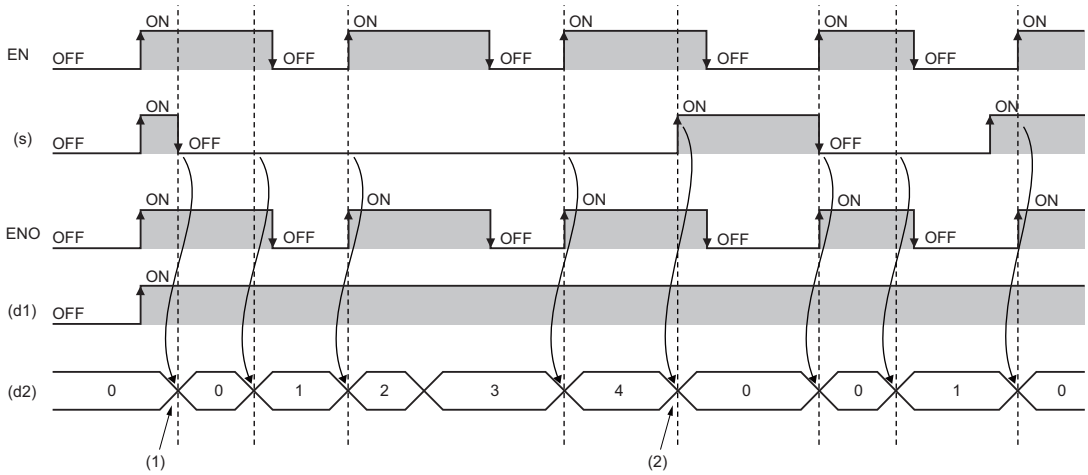
2. Function block with EN/ENO

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
3401H	The output time setting value exceeds the valid range.

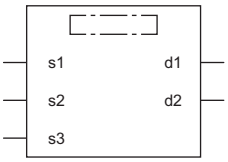
43.8 Timer Function Block

TIMER_□_M

RnCPU | RnENCPU | RnPCPU (Process) | RnPCPU (Redundant) | RnPSFCPU (Standard) | ~~RnPSFCPU (Safety)~~ | RnSFCPU (Standard) | ~~RnSFCPU (Safety)~~

- [Process CPU (redundant mode) and SIL2 Process CPU] If these instructions are used in a program executed in both systems, there are restrictions on their operation when the systems are switched. (📖 MELSEC iQ-R CPU Module User's Manual (Application))

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	Structured text
 (□ 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

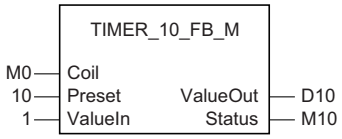
Processing details

■TIMER_10_FB_M

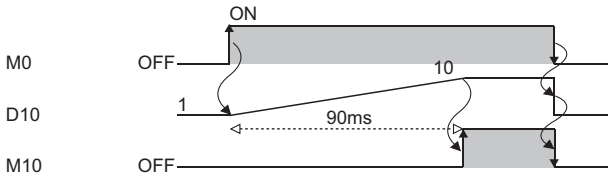
- When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×10ms. When the value reaches (s2)×10ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to 0, and (d2) also turns off.
- If the unit of measurement 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]

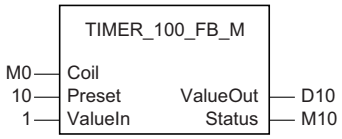


■TIMER_100_FB_M

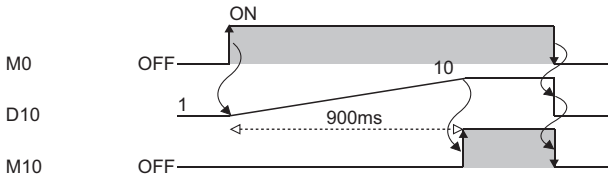
- When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×100ms. When the value reaches (s2)×100ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to 0, and (d2) also turns off.
- If the unit of measurement 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]

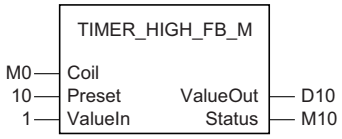


■TIMER_HIGH_FB_M

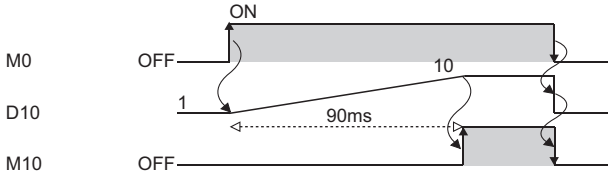
- This is a high-speed timer whose unit of measurement is 0.1 to 100ms. When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×0.1 to 100ms (variable; set in parameter). When the value reaches (s2)×0.1 to 100ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to 0, 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]

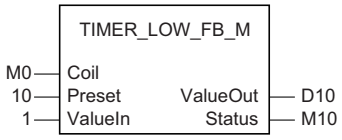


■TIMER_LOW_FB_M

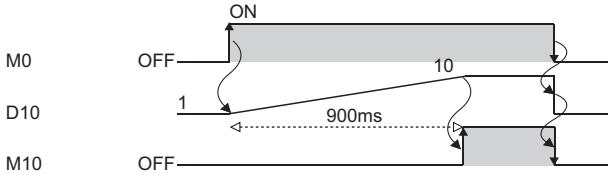
- This is a low-speed timer whose unit of measurement is 1 to 1000ms. When (s1) turns on, measurement of the current value starts. The measurement starts from (s3)×1 to 1000ms (variable; set in parameter). When the value reaches (s2)×1 to 1000ms, (d2) turns on. The measured current value is output to (d1).
- When (s1) turns off, the current value returns to 0, 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]



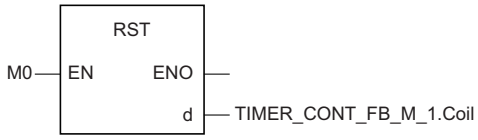
■TIMER_CONT_FB_M/TIMER_CONTHFB_M

- 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

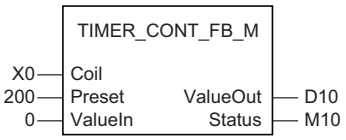
[Ladder program]



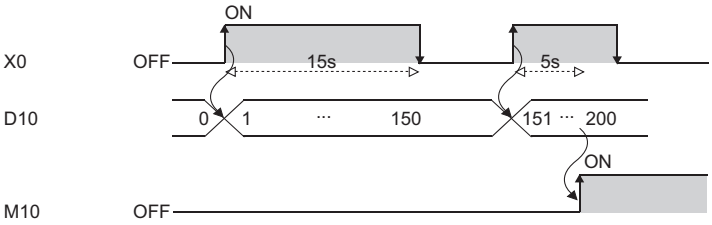
[ST program]

RST(M0,TIMER_CONT_FB_M_1.Coil)

[Ladder example]



[Timing chart]



Operation error

There is no operation error.

APPENDICES

Appendix 1 Instruction Processing Time

The following table lists the processing time of each instruction.
The processing time varies slightly depending on the contents of the source and destination. Assume that the values in the table are reference processing time.

Point

When using the file register (R/ZR), module access device (U□\G□), link direct device (J□\□), or module label (only the ones with the label name includes _D), add extra time described in the section below to each instruction processing time.
☞ Page 1585 Time added to instruction processing time

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
LD	When executed	0.03136		0.00392		0.00098	
LDI	When executed	0.03136		0.00392		0.00098	
AND	When executed	0.03136		0.00392		0.00098	
ANI	When executed	0.03136		0.00392		0.00098	
OR	When executed	0.03136		0.00392		0.00098	
ORI	When executed	0.03136		0.00392		0.00098	
LDP	When executed	0.09408		0.01176		0.00294	
LDF	When executed	0.09408		0.01176		0.00294	
ANDP	When executed	0.09408		0.01176		0.00294	
ANDF	When executed	0.09408		0.01176		0.00294	
ORP	When executed	0.09408		0.01176		0.00294	
ORF	When executed	0.09408		0.01176		0.00294	
LDPI	When executed	0.09408		0.01176		0.00294	
LDFI	When executed	0.09408		0.01176		0.00294	
ANDPI	When executed	0.09408		0.01176		0.00294	
ANDFI	When executed	0.09408		0.01176		0.00294	
ORPI	When executed	0.09408		0.01176		0.00294	
ORFI	When executed	0.09408		0.01176		0.00294	
ANB	—	0.03136		0.00392		0.00098	
ORB	—	0.03136		0.00392		0.00098	
MPS	—	0.03136		0.00392		0.00098	
MRD	—	0.03136		0.00392		0.00098	
MPP	—	0.03136		0.00392		0.00098	
INV	When not executed/When executed	0.03136		0.00392		0.00098	
MEP	When not executed/When executed	0.03136		0.00392		0.00098	
MEF	When not executed/When executed	0.03136		0.00392		0.00098	
EGP	When not executed/When executed	0.06272		0.00784		0.00196	
EGF	When not executed/When executed	0.06272		0.00784		0.00196	
OUT	When not executed/When executed	0.06272		0.00784		0.00196	
OUT (F)	When not executed	0.22272		0.02784		0.00696	
	When executed	52.400	83.100	51.400	82.100	50.900	81.600
OUT (T/ST/C)	When not executed	0.38272		0.04784		0.01196	
	When executed: when counting/after timeout	0.38272		0.04784		0.01196	

A

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
OUT (LT/LST)	When not executed	0.31872		0.03984		0.00996	
	When executed: when counting/after timeout	0.31872		0.03984		0.00996	
OUT (LC)	When not executed	0.38272		0.04784		0.01196	
	When executed: when counting/after timeout	0.38272		0.04784		0.01196	
SET	When not executed	0.06272		0.00784		0.00196	
	When executed: when the value is changed/when the value is not changed	0.06272		0.00784		0.00196	
SET (F)	When not executed	0.22272		0.02784		0.00696	
	When executed	51.800	83.200	50.800	82.200	50.300	81.700
RST	When not executed	0.06272		0.00784		0.00196	
	When executed: when the value is changed/when the value is not changed	0.06272		0.00784		0.00196	
RST (F)	When not executed	0.06272		0.00784		0.00196	
	When executed	17.400	30.800	16.400	29.800	15.900	29.300
RST (T/ST/C)	When not executed	0.31872		0.03984		0.00996	
	When executed	0.31872		0.03984		0.00996	
RST (LT/LST)	When not executed	0.19072		0.02384		0.00596	
	When executed	0.19072		0.02384		0.00596	
RST (LC)	When not executed	0.31872		0.03984		0.00996	
	When executed	0.31872		0.03984		0.00996	
PLS	—	0.06272		0.00784		0.00196	
PLF	—	0.06272		0.00784		0.00196	
FF	When not executed/When executed	0.06272		0.00784		0.00196	
DELTA	When not executed	0.12544		0.01568		0.00392	
	When executed	3.200	7.800	2.200	6.800	1.700	6.300
SFT	When not executed	0.12544		0.01568		0.00392	
	When executed	2.400	4.800	1.400	3.800	0.900	3.300
MC	—	0.06272		0.00784		0.00196	
MCR	—	0.06272		0.00784		0.00196	
PHASE	Continuity	0.59584		0.09800		0.06272	
	Non-continuity/non-processing	0.59584		0.09800		0.06076	
PHASECHG	—	0.18816		0.02352		0.01176	
PHASEEND	—	0.12544		0.01568		0.00784	
FEND	—	Refer to the following. 📖 MELSEC iQ-R CPU Module User's Manual (Application)					
END	—						
STOP	—	—		—		—	
NOP	—	0.03136		0.00392		0.00098	
LD=	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LD<>	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LD>	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LD<=	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LD<	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LD>=	Continuity/Non-continuity	0.18816		0.02352		0.00588	
AND=	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
AND<>	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
AND>	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
AND<=	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
AND<	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
AND>=	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
OR=	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
OR<>	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
OR>	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
OR<=	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
OR<	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
OR>=	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
LD=_U	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LD<>_U	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LD>_U	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LD<=_U	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LD<_U	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LD>=_U	Continuity/Non-continuity	0.18816		0.02352		0.00588	
AND=_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
AND<>_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
AND>_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
AND<=_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
AND<_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
AND>=_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	

A

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
OR=_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
OR<>_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
OR>_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
OR<=_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
OR<_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
OR>=_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
LDD=	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LDD<>	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LDD>	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LDD<=	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LDD<	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LDD>=	Continuity/Non-continuity	0.18816		0.02352		0.00588	
ANDD=	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ANDD<>	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ANDD>	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ANDD<=	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ANDD<	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ANDD>=	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORD=	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORD<>	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORD>	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORD<=	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
ORD<	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORD>=	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
LDD=_U	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LDD<>_U	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LDD>_U	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LDD<=_U	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LDD<_U	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LDD>=_U	Continuity/Non-continuity	0.18816		0.02352		0.00588	
ANDD=_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ANDD<>_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ANDD>_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ANDD<=_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ANDD<_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ANDD>=_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORD=_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORD<>_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORD>_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORD<=_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORD<_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORD>=_U	When not executed	0.12544		0.01568		0.00392	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
CMP	—	4.400	6.900	3.400	5.900	2.900	5.400
CMP_U	—	4.400	6.900	3.400	5.900	2.900	5.400
DCMP	—	4.400	6.900	3.400	5.900	2.900	5.400
DCMP_U	—	4.400	6.900	3.400	5.900	2.900	5.400
ZCP	—	4.800	7.700	3.800	6.700	3.300	6.200
ZCP_U	—	4.800	7.700	3.800	6.700	3.300	6.200

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
DZCP	—	4.800	7.700	3.800	6.700	3.300	6.200
DZCP_U	—	4.800	7.700	3.800	6.700	3.300	6.200
BKCMP=	(n)=1	5.100	12.000	4.100	11.000	3.600	10.500
	(n)=96	18.500	24.900	17.500	23.900	17.000	23.400
BKCMP<>	(n)=1	5.200	11.400	4.200	10.400	3.700	9.900
	(n)=96	18.900	25.100	17.900	24.100	17.400	23.600
BKCMP>	(n)=1	5.100	12.000	4.100	11.000	3.600	10.500
	(n)=96	18.300	26.100	17.300	25.100	16.800	24.600
BKCMP<=	(n)=1	5.000	11.800	4.000	10.800	3.500	10.300
	(n)=96	19.000	24.800	18.000	23.800	17.500	23.300
BKCMP<	(n)=1	5.100	11.800	4.100	10.800	3.600	10.300
	(n)=96	19.000	26.000	18.000	25.000	17.500	24.500
BKCMP>=	(n)=1	5.100	11.500	4.100	10.500	3.600	10.000
	(n)=96	19.200	25.600	18.200	24.600	17.700	24.100
BKCMP=_U	(n)=1	5.100	11.800	4.100	10.800	3.600	10.300
	(n)=96	18.400	25.400	17.400	24.400	16.900	23.900
BKCMP<>_U	(n)=1	5.200	11.300	4.200	10.300	3.700	9.800
	(n)=96	18.800	25.100	17.800	24.100	17.300	23.600
BKCMP>_U	(n)=1	5.000	11.700	4.000	10.700	3.500	10.200
	(n)=96	18.400	25.100	17.400	24.100	16.900	23.600
BKCMP<=_U	(n)=1	5.100	11.600	4.100	10.600	3.600	10.100
	(n)=96	18.200	24.800	17.200	23.800	16.700	23.300
BKCMP<_U	(n)=1	5.200	11.600	4.200	10.600	3.700	10.100
	(n)=96	18.300	25.600	17.300	24.600	16.800	24.100
BKCMP>=_U	(n)=1	5.100	11.400	4.100	10.400	3.600	9.900
	(n)=96	18.500	26.900	17.500	25.900	17.000	25.400
DBKCMP=	(n)=1	5.200	11.900	4.200	10.900	3.700	10.400
	(n)=96	18.500	25.000	17.500	24.000	17.000	23.500
DBKCMP<>	(n)=1	5.200	11.500	4.200	10.500	3.700	10.000
	(n)=96	18.900	25.200	17.900	24.200	17.400	23.700
DBKCMP>	(n)=1	5.100	11.600	4.100	10.600	3.600	10.100
	(n)=96	18.900	25.600	17.900	24.600	17.400	24.100
DBKCMP<=	(n)=1	5.300	11.600	4.300	10.600	3.800	10.100
	(n)=96	18.600	25.100	17.600	24.100	17.100	23.600
DBKCMP<	(n)=1	5.200	11.400	4.200	10.400	3.700	9.900
	(n)=96	18.800	25.300	17.800	24.300	17.300	23.800
DBKCMP>=	(n)=1	5.100	11.700	4.100	10.700	3.600	10.200
	(n)=96	18.600	24.400	17.600	23.400	17.100	22.900
DBKCMP=_U	(n)=1	5.200	11.700	4.200	10.700	3.700	10.200
	(n)=96	18.500	25.100	17.500	24.100	17.000	23.600
DBKCMP<>_U	(n)=1	5.200	11.500	4.200	10.500	3.700	10.000
	(n)=96	18.900	25.100	17.900	24.100	17.400	23.600
DBKCMP>_U	(n)=1	5.200	11.700	4.200	10.700	3.700	10.200
	(n)=96	25.800	32.300	24.800	31.300	24.300	30.800
DBKCMP<=_U	(n)=1	5.200	11.700	4.200	10.700	3.700	10.200
	(n)=96	25.400	31.800	24.400	30.800	23.900	30.300
DBKCMP<_U	(n)=1	5.300	11.500	4.300	10.500	3.800	10.000
	(n)=96	25.800	32.500	24.800	31.500	24.300	31.000
DBKCMP>=_U	(n)=1	5.300	12.000	4.300	11.000	3.800	10.500
	(n)=96	25.900	32.500	24.900	31.500	24.400	31.000
+ (s) (d)	When executed	0.18816		0.02352		0.00588	

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
+ (s1) (s2) (d)	When executed	0.18816		0.02352		0.00588	
+_U (s) (d)	When executed	0.18816		0.02352		0.00588	
+_U (s1) (s2) (d)	When executed	0.18816		0.02352		0.00588	
- (s) (d)	When executed	0.18816		0.02352		0.00588	
- (s1) (s2) (d)	When executed	0.18816		0.02352		0.00588	
-_U (s) (d)	When executed	0.18816		0.02352		0.00588	
-_U (s1) (s2) (d)	When executed	0.18816		0.02352		0.00588	
D+ (s) (d)	When executed	0.18816		0.02352		0.00588	
D+ (s1) (s2) (d)	When executed	0.18816		0.02352		0.00588	
D+_U (s) (d)	When executed	0.18816		0.02352		0.00588	
D+_U (s1) (s2) (d)	When executed	0.18816		0.02352		0.00588	
D- (s) (d)	When executed	0.18816		0.02352		0.00588	
D- (s1) (s2) (d)	When executed	0.18816		0.02352		0.00588	
D-_U (s) (d)	When executed	0.18816		0.02352		0.00588	
D-_U (s1) (s2) (d)	When executed	0.18816		0.02352		0.00588	
*	When executed	0.37632		0.04704		0.01176	
*_U	When executed	0.37632		0.04704		0.01176	
/	When executed	0.62720		0.07840		0.01960	
/_U	When executed	0.62720		0.07840		0.01960	
D*	When executed	0.62720		0.07840		0.01960	
D*_U	When executed	0.62720		0.07840		0.01960	
D/	When executed	0.94080		0.11760		0.02940	
D/_U	When executed	0.94080		0.11760		0.02940	
B+ (s) (d)	When executed	2.800	5.200	1.800	4.200	1.300	3.700
B+ (s1) (s2) (d)	When executed	3.300	6.600	2.300	5.600	1.800	5.100
B- (s) (d)	When executed	2.800	5.300	1.800	4.300	1.300	3.800
B- (s1) (s2) (d)	When executed	3.400	6.500	2.400	5.500	1.900	5.000
DB+ (s) (d)	When executed	3.500	7.700	2.500	6.700	2.000	6.200
DB+ (s1) (s2) (d)	When executed	3.600	7.600	2.600	6.600	2.100	6.100
DB- (s) (d)	When executed	3.500	7.600	2.500	6.600	2.000	6.100
DB- (s1) (s2) (d)	When executed	3.600	7.400	2.600	6.400	2.100	5.900
B*	When executed	2.900	6.400	1.900	5.400	1.400	4.900
B/	When executed	3.000	6.300	2.000	5.300	1.500	4.800
DB*	When executed	3.900	9.500	2.900	8.500	2.400	8.000
DB/	When executed	3.700	9.400	2.700	8.400	2.200	7.900
BK+	(n)=1	5.000	9.600	4.000	8.600	3.500	8.100
	(n)=96	18.700	23.000	17.700	22.000	17.200	21.500
BK+_U	(n)=1	4.700	9.800	3.700	8.800	3.200	8.300
	(n)=96	19.000	23.100	18.000	22.100	17.500	21.600
BK-	(n)=1	4.600	9.700	3.600	8.700	3.100	8.200
	(n)=96	18.900	22.900	17.900	21.900	17.400	21.400
BK-_U	(n)=1	4.700	9.800	3.700	8.800	3.200	8.300
	(n)=96	19.000	23.000	18.000	22.000	17.500	21.500
DBK+	(n)=1	4.800	8.400	3.800	7.400	3.300	6.900
	(n)=96	18.900	23.400	17.900	22.400	17.400	21.900
DBK+_U	(n)=1	4.500	9.300	3.500	8.300	3.000	7.800
	(n)=96	18.900	23.400	17.900	22.400	17.400	21.900
DBK-	(n)=1	4.500	9.400	3.500	8.400	3.000	7.900
	(n)=96	18.900	23.500	17.900	22.500	17.400	22.000

A

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
DBK-_U	(n)=1	4.500	9.400	3.500	8.400	3.000	7.900
	(n)=96	18.900	23.500	17.900	22.500	17.400	22.000
INC	When executed	0.12544		0.01568		0.00392	
INC_U	When executed	0.12544		0.01568		0.00392	
DEC	When executed	0.12544		0.01568		0.00392	
DEC_U	When executed	0.12544		0.01568		0.00392	
DINC	When executed	0.12544		0.01568		0.00392	
DINC_U	When executed	0.12544		0.01568		0.00392	
DDEC	When executed	0.12544		0.01568		0.00392	
DDEC_U	When executed	0.12544		0.01568		0.00392	
WAND (s) (d)	When executed	0.12544		0.01568		0.00392	
WAND (s1) (s2) (d)	When executed	0.12544		0.01568		0.00392	
DAND (s) (d)	When executed	0.12544		0.01568		0.00392	
DAND (s1) (s2) (d)	When executed	0.12544		0.01568		0.00392	
BKAND	(n)=1	4.800	9.900	3.800	8.900	3.300	8.400
	(n)=96	19.000	23.900	18.000	22.900	17.500	22.400
WOR (s) (d)	When executed	0.12544		0.01568		0.00392	
WOR (s1) (s2) (d)	When executed	0.12544		0.01568		0.00392	
DOR (s) (d)	When executed	0.12544		0.01568		0.00392	
DOR (s1) (s2) (d)	When executed	0.12544		0.01568		0.00392	
BKOR	(n)=1	4.800	9.800	3.800	8.800	3.300	8.300
	(n)=96	19.000	24.300	18.000	23.300	17.500	22.800
WXOR (s) (d)	When executed	0.12544		0.01568		0.00392	
WXOR (s1) (s2) (d)	When executed	0.12544		0.01568		0.00392	
DXOR (s) (d)	When executed	0.12544		0.01568		0.00392	
DXOR (s1) (s2) (d)	When executed	0.12544		0.01568		0.00392	
BKXOR	(n)=1	4.800	9.800	3.800	8.800	3.300	8.300
	(n)=96	19.000	23.900	18.000	22.900	17.500	22.400
WXNR (s) (d)	When executed	0.12544		0.01568		0.00392	
WXNR (s1) (s2) (d)	When executed	0.12544		0.01568		0.00392	
DXNR (s) (d)	When executed	0.12544		0.01568		0.00392	
DXNR (s1) (s2) (d)	When executed	0.12544		0.01568		0.00392	
BKXNR	(n)=1	4.800	9.700	3.800	8.700	3.300	8.200
	(n)=96	19.400	24.300	18.400	23.300	17.900	22.800
BSET	(n)=1	0.09408		0.01176		0.00294	
	(n)=15	0.09408		0.01176		0.00294	
BRST	(n)=1	0.09408		0.01176		0.00294	
	(n)=15	0.09408		0.01176		0.00294	
TEST	When executed	0.38400		0.04800		0.0120	
DTEST	When executed	0.60800		0.07600		0.0190	
BKRST	(n)=1	3.000	3.500	1.100	2.300	0.900	2.000
	(n)=96	4.600	5.300	1.600	2.500	1.200	2.200
SFR	(n)=1	0.28224		0.03528		0.00882	
	(n)=15	0.28224		0.03528		0.00882	

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
SFL	(n)=1	0.28224		0.03528		0.00882	
	(n)=15	0.28224		0.03528		0.00882	
BSFR	(n)=1	3.000	3.500	1.100	2.000	0.800	1.800
	(n)=96	6.200	6.700	1.900	3.000	1.500	2.600
BSFL	(n)=1	3.000	3.500	1.100	2.200	0.800	1.900
	(n)=96	6.200	6.700	1.900	3.000	1.500	2.600
DSFR	(n)=1	2.600	4.800	1.600	3.800	1.100	3.300
	(n)=96	9.700	12.300	8.700	11.300	8.200	10.800
DSFL	(n)=1	2.600	4.900	1.600	3.900	1.100	3.400
	(n)=96	9.700	12.200	8.700	11.200	8.200	10.700
DDSFR	(n)=1	3.700	5.300	2.700	4.300	2.200	5.400
	(n)=96	4.500	6.700	3.500	5.700	3.200	6.800
DDSFL	(n)=1	3.600	5.100	2.600	4.100	2.200	5.400
	(n)=96	4.600	6.600	3.600	5.600	3.200	6.800
ESFR	(n)=1	3.700	5.300	2.700	4.300	2.200	5.400
	(n)=96	4.600	6.800	3.600	5.800	3.200	6.800
ESFL	(n)=1	3.700	5.000	2.700	4.000	2.200	5.400
	(n)=96	4.700	6.600	3.700	5.600	3.200	6.800
EDSFR	(n)=1	3.800	5.700	2.800	4.700	2.800	5.800
	(n)=96	5.100	7.500	4.100	6.500	4.300	7.800
EDSFL	(n)=1	3.800	5.600	2.800	4.600	2.800	5.800
	(n)=96	5.100	7.500	4.100	6.500	4.300	7.800
SFTBR	(n1)=16, (n2)=1	7.600	8.200	1.900	2.900	0.900	2.100
	(n1)=16, (n2)=15	7.600	8.200	1.900	2.900	0.900	2.100
SFTR	(n1)=16, (n2)=1	5.900	12.800	4.900	11.800	4.400	11.300
	(n1)=16, (n2)=15	5.900	12.800	4.900	11.800	4.400	11.300
SFTBL	(n1)=16, (n2)=1	7.600	8.200	1.900	2.900	0.900	2.100
	(n1)=16, (n2)=15	7.600	8.200	1.900	2.900	0.900	2.100
SFTL	(n1)=16, (n2)=1	5.900	12.800	4.900	11.800	4.400	11.300
	(n1)=16, (n2)=15	5.900	12.800	4.900	11.800	4.400	11.300
SFTWR	(n1)=16, (n2)=1	3.900	6.900	2.900	5.900	2.400	5.400
	(n1)=16, (n2)=15	3.900	7.200	2.900	6.200	2.400	5.700
WSFR	(n1)=16, (n2)=1	6.600	11.100	5.600	10.100	5.100	9.600
	(n1)=16, (n2)=15	6.600	11.100	5.600	10.100	5.100	9.600
SFTWL	(n1)=16, (n2)=1	3.900	7.300	2.900	6.300	2.400	5.800
	(n1)=16, (n2)=15	3.900	6.900	2.900	5.900	2.400	5.400
WSFL	(n1)=16, (n2)=1	6.600	11.100	5.600	10.100	5.100	9.600
	(n1)=16, (n2)=15	6.600	11.100	5.600	10.100	5.100	9.600
SFTDWR	(n1)=16, (n2)=1	5.200	7.500	4.200	6.500	3.900	7.700
	(n1)=16, (n2)=15	5.300	7.600	4.300	6.600	3.900	7.700
DWSFTR	(n1)=16, (n2)=1	6.300	11.200	5.300	10.200	5.100	11.500
	(n1)=16, (n2)=15	6.300	11.100	5.300	10.100	5.100	11.500
SFTDWL	(n1)=16, (n2)=1	5.200	7.500	4.200	6.500	3.900	7.700
	(n1)=16, (n2)=15	5.200	7.500	4.200	6.500	3.900	7.700
DWSFTL	(n1)=16, (n2)=1	6.700	11.400	5.700	10.400	5.100	11.500
	(n1)=16, (n2)=15	6.800	11.300	5.800	10.300	5.100	11.500
SFTER	(n1)=16, (n2)=1	5.300	7.500	4.300	6.500	3.900	7.700
	(n1)=16, (n2)=15	5.300	7.600	4.300	6.600	3.900	7.700
ESFTR	(n1)=16, (n2)=1	6.400	11.200	5.400	10.200	5.100	11.500
	(n1)=16, (n2)=15	6.300	11.200	5.300	10.200	5.100	11.500

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
SFTEL	(n1)=16, (n2)=1	5.200	7.500	4.200	6.500	3.900	7.700
	(n1)=16, (n2)=15	5.200	7.500	4.200	6.500	3.900	7.700
ESFTL	(n1)=16, (n2)=1	6.800	11.300	5.800	10.300	5.100	11.500
	(n1)=16, (n2)=15	6.800	11.300	5.800	10.300	5.100	11.500
SFTEDR	(n1)=16, (n2)=1	5.100	7.600	4.100	6.600	4.100	7.900
	(n1)=16, (n2)=15	5.200	7.700	4.200	6.700	4.100	7.900
EDSFTR	(n1)=16, (n2)=1	6.300	11.300	5.300	10.300	5.800	11.800
	(n1)=16, (n2)=15	6.400	11.200	5.400	10.200	5.800	11.800
SFTEDL	(n1)=16, (n2)=1	5.200	7.400	4.200	6.400	4.100	7.900
	(n1)=16, (n2)=15	5.300	7.500	4.300	6.500	4.100	7.900
EDSFTL	(n1)=16, (n2)=1	6.400	11.200	5.400	10.200	5.800	11.800
	(n1)=16, (n2)=15	6.500	11.200	5.500	10.200	5.800	11.800
BCD	When executed	0.34496		0.04312		0.01078	
DBCD	When executed	0.59584		0.07448		0.01862	
BIN	When executed	0.21952		0.02744		0.00686	
DBIN	When executed	0.21952		0.02744		0.00686	
FLT2INT	(s)=0	0.21952		0.02744		0.00686	
	(s)=32766.5	0.21952		0.02744		0.00686	
FLT2UINT	(s)=0	0.21952		0.02744		0.00686	
	(s)=65534.5	0.21952		0.02744		0.00686	
FLT2DINT	(s)=0	0.21952		0.02744		0.00686	
	(s)=1234567890.3	0.21952		0.02744		0.00686	
FLT2UDINT	(s)=0	0.21952		0.02744		0.00686	
	(s)=1234567890.3	0.21952		0.02744		0.00686	
DBL2INT	(s)=0	2.900	4.900	1.900	3.900	1.400	3.400
	(s)=32766.5	3.000	5.600	2.000	4.600	1.500	4.100
DBL2UINT	(s)=0	2.800	5.100	1.800	4.100	1.300	3.600
	(s)=65534.5	3.000	5.500	2.000	4.500	1.500	4.000
DBL2DINT	(s)=0	2.900	4.900	1.900	3.900	1.400	3.400
	(s)=1234567890.3	2.900	5.700	1.900	4.700	1.400	4.200
DBL2UDINT	(s)=0	3.000	4.900	2.000	3.900	1.500	3.400
	(s)=1234567890.3	3.000	5.600	2.000	4.600	1.500	4.100
INT2UINT	When executed	0.09408		0.01176		0.00294	
INT2DINT	When executed	0.09408		0.01176		0.00294	
INT2UDINT	When executed	0.09408		0.01176		0.00294	
UINT2INT	When executed	0.09408		0.01176		0.00294	
UINT2DINT	When executed	0.09408		0.01176		0.00294	
UINT2UDINT	When executed	0.09408		0.01176		0.00294	
DINT2INT	When executed	0.09408		0.01176		0.00294	
DINT2UINT	When executed	0.09408		0.01176		0.00294	
DINT2UDINT	When executed	0.09408		0.01176		0.00294	
UDINT2INT	When executed	0.09408		0.01176		0.00294	
UDINT2UINT	When executed	0.09408		0.01176		0.00294	
UDINT2DINT	When executed	0.09408		0.01176		0.00294	
GRY	When executed	0.15680		0.01960		0.00490	
GRY_U	When executed	0.15680		0.01960		0.00490	
DGRY	When executed	0.15680		0.01960		0.00490	
DGRY_U	When executed	0.15680		0.01960		0.00490	
GBIN	When executed	0.15680		0.01960		0.00490	
GBIN_U	When executed	0.15680		0.01960		0.00490	
DGBIN	When executed	0.15680		0.01960		0.00490	

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
DGBIN_U	When executed	0.15680		0.01960		0.00490	
BKBCD	(n)=1	3.900	9.300	2.900	8.300	2.400	7.800
	(n)=96	22.200	27.200	21.200	26.200	20.700	25.700
BKBIN	(n)=1	3.800	8.900	2.800	7.900	2.300	7.400
	(n)=96	18.600	23.400	17.600	22.400	17.100	21.900
DABIN	(s)=1	4.500	12.300	3.500	11.300	3.000	10.800
	(s)=-32768	4.500	12.300	3.500	11.300	3.000	10.800
DABIN_U	(s)=1	4.500	12.300	3.500	11.300	3.000	10.800
	(s)=65535	4.500	12.300	3.500	11.300	3.000	10.800
DDABIN	(s)=1	4.900	12.600	3.900	11.600	3.400	11.100
	(s)=-2147483648	4.900	12.600	3.900	11.600	3.400	11.100
DDABIN_U	(s)=1	4.900	12.600	3.900	11.600	3.400	11.100
	(s)=4294967295	4.900	12.600	3.900	11.600	3.400	11.100
HABIN	(s)=1	4.400	10.200	3.400	9.200	2.900	8.700
	(s)=FFFFH	4.400	10.200	3.400	9.200	2.900	8.700
DHABIN	(s)=1	4.600	10.500	3.600	9.500	3.100	9.000
	(s)=FFFFFFFFH	4.600	10.500	3.600	9.500	3.100	9.000
DABCD	(s)=1	4.300	10.400	3.300	9.400	2.800	8.900
	(s)=9999	4.300	10.400	3.300	9.400	2.800	8.900
DDABCD	(s)=1	4.500	10.600	3.500	9.600	3.000	9.100
	(s)=99999999	4.500	10.600	3.500	9.600	3.000	9.100
VAL	—	5.300	13.400	4.300	12.400	3.800	11.900
VAL_U	—	5.600	13.400	4.600	12.400	4.100	11.900
DVAL	—	6.400	14.600	5.400	13.600	4.900	13.100
DVAL_U	—	6.700	14.100	5.700	13.100	5.200	12.600
ASC2INT	(n)=1	4.100	8.900	3.100	7.900	2.600	7.400
	(n)=96	9.800	15.200	8.800	14.200	8.300	13.700
EMOD	—	4.500	9.700	3.500	8.700	3.000	8.200
NEG	When executed	0.15680		0.01960		0.00490	
DNEG	When executed	0.15680		0.01960		0.00490	
DECO	(n)=2	3.900	7.400	2.900	6.400	2.400	5.900
	(n)=8	3.900	7.800	2.900	6.800	2.400	6.300
ENCO	(n)=2, M1=ON	4.000	7.900	3.000	6.900	2.500	6.400
	(n)=2, M4=ON	4.000	7.900	3.000	6.900	2.500	6.400
	(n)=8, M1=ON	5.000	9.300	4.000	8.300	3.500	7.800
	(n)=8, M256=ON	3.900	7.700	2.900	6.700	2.400	6.200
SEG	When executed	1.900	3.100	0.900	2.100	0.400	1.600
DIS	(n)=1	3.700	5.700	2.700	4.700	2.200	4.200
	(n)=4	3.700	5.800	2.700	4.800	2.200	4.300
UNI	(n)=1	3.900	6.800	2.900	5.800	2.400	5.300
	(n)=4	4.000	6.700	3.000	5.700	2.500	5.200
NDIS	When executed	3.700	6.500	2.700	5.500	2.200	5.000
NUNI	When executed	3.700	6.500	2.700	5.500	2.200	5.000
WTOB	(n)=1	4.000	7.000	3.000	6.000	2.500	5.500
	(n)=96	14.500	17.500	13.500	16.500	13.000	16.000
BTOW	(n)=1	4.100	7.000	3.100	6.000	2.600	5.500
	(n)=96	11.300	14.100	10.300	13.100	9.800	12.600
MOV	—	0.06272		0.00784		0.00196	
DMOV	—	0.06272		0.00784		0.00196	
CML	—	0.06272		0.00784		0.00196	
DCML	—	0.06272		0.00784		0.00196	

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
SMOV	SM773=OFF	5.000	10.000	4.000	9.000	3.500	8.500
	SM773=ON	4.700	7.600	3.700	6.600	3.200	6.100
CMLB	—	0.06272		0.00784		0.00196	
BMOV	(n)=1	4.800	5.000	1.500	1.600	1.100	1.300
	(n)=96	6.600	6.700	2.500	2.700	2.000	2.200
BMOVL	(n)=1	5.000	5.500	1.700	2.900	1.300	2.500
	(n)=96	6.900	7.400	2.700	4.100	2.200	3.600
FMOV	(n)=1	3.800	4.000	1.100	1.400	0.700	1.000
	(n)=96	5.700	6.000	2.300	2.500	1.800	2.000
FMOVL	(n)=1	4.100	4.600	1.300	2.300	1.000	2.000
	(n)=96	6.000	6.600	2.400	3.700	2.000	3.300
DFMOV	(n)=1	4.000	4.500	1.300	2.300	1.000	2.000
	(n)=96	6.200	6.600	2.600	3.800	2.200	3.400
DFMOVL	(n)=1	4.200	4.600	1.400	2.400	1.000	2.000
	(n)=96	6.300	6.700	2.700	4.000	2.200	3.500
XCH	—	3.000	3.400	1.100	1.700	0.900	1.500
DXCH	—	3.100	3.400	1.200	1.800	0.900	1.500
BXCH	(n)=1	4.200	8.300	3.200	7.300	2.700	6.800
	(n)=96	18.400	22.200	17.400	21.200	16.900	20.700
SWAP	—	2.700	3.700	1.700	2.700	1.200	2.200
DSWAP	—	2.700	3.700	1.700	2.700	1.200	2.200
MOVB	—	0.06272		0.00784		0.00196	
BLKMOVB	(n)=1	4.600	9.100	3.600	8.100	3.100	7.600
	(n)=96	5.100	10.100	4.100	9.100	3.600	8.600
ROR	(n)=1	0.28224		0.03528		0.00882	
	(n)=15	0.28224		0.03528		0.00882	
RCR	(n)=1	0.28224		0.03528		0.00882	
	(n)=15	0.28224		0.03528		0.00882	
DROR	(n)=1	0.28224		0.03528		0.00882	
	(n)=31	0.28224		0.03528		0.00882	
DRCR	(n)=1	0.28224		0.03528		0.00882	
	(n)=31	0.28224		0.03528		0.00882	
ROL	(n)=1	0.28224		0.03528		0.00882	
	(n)=15	0.28224		0.03528		0.00882	
RCL	(n)=1	0.28224		0.03528		0.00882	
	(n)=15	0.28224		0.03528		0.00882	
DROL	(n)=1	0.28224		0.03528		0.00882	
	(n)=31	0.28224		0.03528		0.00882	
DRCL	(n)=1	0.28224		0.03528		0.00882	
	(n)=31	0.28224		0.03528		0.00882	
CJ	—	4.000	4.700	1.300	2.300	1.000	2.000
SCJ	—	1.300	2.400	1.100	2.200	1.000	2.000
JMP	—	4.100	4.900	1.400	2.500	1.000	2.100
GOEND	—	28.80000		3.60000		0.9000	
DI	—	6.000	8.300	3.700	6.000	2.800	4.200
DI (s)	—	6.600	8.900	3.800	6.100	3.000	4.200
EI	—	6.000	11.000	4.600	10.500	4.100	9.200
IMASK	—	3.500	4.000	1.300	2.000	1.000	1.800
SIMASK	—	3.100	3.700	1.000	2.000	0.800	1.700
IRET	—	3.200	3.700	2.200	2.700	1.700	2.200
WDT	—	6.400	17.900	5.400	16.900	4.900	16.400

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
FOR	—	0.06272		0.00784		0.00196	
NEXT	—	1.25440		0.15680		0.03920	
BREAK	—	5.900	11.500	4.900	10.500	4.400	10.000
CALL Pn	Local pointer	3.700	4.300	1.100	1.800	0.800	1.500
	Global pointer	8.200	17.700	4.900	15.500	3.900	13.500
CALL Pn (s1) to (s5)	Local pointer	18.900	42.100	17.900	41.100	17.400	40.600
RET	Return to the own program	6.40000		0.80000		0.2000	
	Return to the another program	64.00000		8.00000		2.0000	
FCALL Pn	Local pointer	5.500	6.200	1.800	2.500	1.000	2.000
	Global pointer	14.000	34.000	9.000	30.000	4.300	22.900
FCALL Pn (s1) to (s5)	Local pointer	17.900	39.300	16.900	38.300	16.400	37.800
ECALL Pn	Local pointer, file name="P1"	90.000	145.500	83.500	134.000	74.700	118.400
ECALL Pn (s1) to (s5)	Local pointer, file name="P1"	96.000	149.900	95.000	148.900	94.500	148.400
EFCALL Pn	Local pointer, file name="P1"	85.000	130.000	82.000	127.000	72.400	114.300
EFCALL Pn (s1) to (s5)	Local pointer, file name="P1"	92.600	133.700	91.600	132.700	91.100	132.200
XCALL Pn	—	6.800	20.700	4.500	19.000	3.800	16.700
FIFR	Number of data stored = 1	3.600	6.300	2.600	5.300	2.100	4.800
	Number of data stored = 96	8.600	11.800	7.600	10.800	7.100	10.300
FPOP	Number of data stored = 1	3.500	6.100	2.500	5.100	2.000	4.600
	Number of data stored = 96	3.600	6.100	2.600	5.100	2.100	4.600
FIFW	Number of data stored = 0	3.600	6.300	2.600	5.300	2.100	4.800
	Number of data stored = 96	3.600	6.200	2.600	5.200	2.100	4.700
FINS	Number of data stored = 0	4.000	7.300	3.000	6.300	2.500	5.800
	Number of data stored = 96	10.700	15.200	9.700	14.200	9.200	13.700
FDEL	Number of data stored = 1	3.800	7.500	2.800	6.500	2.300	6.000
	Number of data stored = 96	8.900	13.000	7.900	12.000	7.400	11.500
S.DEVLD	—	5.800	8.800	4.800	7.800	4.300	7.300
SP.DEVST	—	21.700	26.300	20.700	25.300	20.200	24.800
SP.FREAD	—	38.300	44.300	37.300	43.300	36.800	42.800
SP.FWRITE	—	43.000	48.900	42.000	47.900	41.500	47.400
SP.FDELETE	—	26.700	30.200	25.700	29.200	25.200	28.700
SP.FCOPY	—	31.100	34.300	30.100	33.300	29.600	32.800
SP.FMOVE	—	31.100	34.300	30.100	33.300	29.600	32.800
SP.FRENAME	—	32.500	35.900	31.500	34.900	31.000	34.400
SP.FSTATUS	—	28.700	32.000	27.700	31.000	27.200	30.500
LEDR	No self-diagnostic error	3.600	6.700	2.600	5.700	2.100	5.200
	Self-diagnostic error (continuation error, annunciator ON)	16.500	27.400	15.500	26.400	15.000	25.900
PALERT(P)	—	—	—	—	—	77.100	146.000
PABORT	—	—	—	—	—	—	—
LD$=	Continuity/Non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
LD$<>	Continuity/Non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
LD$>	Continuity/Non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
LD$<=	Continuity/Non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
LD$<	Continuity/Non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
LD$>=	Continuity/Non-continuity	3.100	6.000	2.100	5.000	1.600	4.500

A

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
AND$=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
AND$<>	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
AND$>	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
AND$<=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
AND$<	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
AND$>=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
OR$=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
OR$<>	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
OR$>	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
OR$<=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
OR$<	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
OR$>=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.100	6.000	2.100	5.000	1.600	4.500
$+ (s) (d)	When executed	3.900	8.200	2.900	7.200	2.400	6.700
$+ (s1) (s2) (d)	When executed	4.400	9.700	3.400	8.700	2.900	8.200
$MOV	0 characters	3.900	10.700	2.900	9.700	2.400	9.200
	32 characters	5.900	13.600	4.900	12.600	4.400	12.100
$MOV_WS	0 characters	3.900	11.600	2.900	10.600	2.400	10.100
	32 characters	7.600	15.700	6.600	14.700	6.100	14.200
BINDA	(s)=1	4.100	7.400	3.100	6.400	2.600	5.900
	(s)=-32768	4.500	8.300	3.500	7.300	3.000	6.800
BINDA_U	(s)=1	4.100	7.400	3.100	6.400	2.600	5.900
	(s)=65535	4.500	8.300	3.500	7.300	3.000	6.800
DBINDA	(s)=1	4.200	7.600	3.200	6.600	2.700	6.100
	(s)=-2147483648	4.600	8.900	3.600	7.900	3.100	7.400
DBINDA_U	(s)=1	4.200	7.600	3.200	6.600	2.700	6.100
	(s)=4294967295	4.600	8.900	3.600	7.900	3.100	7.400
BINHA	(s)=1	4.100	7.000	3.100	6.000	2.600	5.500
	(s)=FFFFH	4.100	7.100	3.100	6.100	2.600	5.600
DBINHA	(s)=1	4.100	6.800	3.100	5.800	2.600	5.300
	(s)=FFFFFFFFH	4.100	7.200	3.100	6.200	2.600	5.700

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
STR	—	4.300	8.800	3.300	7.800	2.800	7.300
STR_U	—	4.400	8.800	3.400	7.800	2.900	7.300
DSTR	—	4.700	8.900	3.700	7.900	3.200	7.400
DSTR_U	—	4.800	9.400	3.800	8.400	3.300	7.900
BCDDA	(s)=1	4.100	7.700	3.100	6.700	2.600	6.200
	(s)=9999	4.200	7.800	3.200	6.800	2.700	6.300
DBCDDA	(s)=1	4.100	7.900	3.100	6.900	2.600	6.400
	(s)=99999999	4.200	8.200	3.200	7.200	2.700	6.700
ESTR	—	6.200	18.600	5.200	17.600	4.7000	17.1000
INT2ASC	(n)=1	4.100	8.400	3.100	7.400	2.600	6.900
	(n)=96	7.600	12.500	6.600	11.500	6.100	11.000
WS2SJIS	Number of characters = 1	4.800	11.800	3.800	10.800	3.300	10.300
	Number of characters = 96	54.000	60.700	53.000	59.700	52.500	59.200
SJIS2WS	Number of characters = 1	4.600	11.400	3.600	10.400	3.100	9.900
	Number of characters = 96	49.800	56.700	48.800	55.700	48.300	55.200
SJIS2WSB	Number of characters = 1	4.700	11.300	3.700	10.300	3.200	9.800
	Number of characters = 96	49.800	56.600	48.800	55.600	48.300	55.100
LEN	1 characters	2.900	5.300	1.900	4.300	1.400	3.800
	96 characters	11.100	13.400	10.100	12.400	9.600	11.900
RIGHT	Number of characters to be extracted = 1	4.700	12.700	3.700	11.700	3.200	11.200
	Number of characters to be extracted = 96	17.300	24.900	16.300	23.900	15.800	23.400
LEFT	Number of characters to be extracted = 1	4.700	12.600	3.700	11.600	3.200	11.100
	Number of characters to be extracted = 96	17.200	24.700	16.200	23.700	15.700	23.200
MIDR	—	5.100	13.700	4.100	12.700	3.600	12.200
MIDW	—	5.700	13.800	4.700	12.800	4.200	12.300
INSTR	No match	9.000	16.500	8.000	15.500	7.500	15.000
	Match: Start	6.800	14.500	5.800	13.500	5.300	13.000
	Match: Last	9.000	16.800	8.000	15.800	7.500	15.300
STRINS	(s)=128, (d)=40, (n)=1	19.400	28.700	18.400	27.700	17.900	27.200
	(s)=128, (d)=40, (n)=48	21.900	31.600	20.900	30.600	20.400	30.100
STRDEL	(s)=128, (d)=40, (n)=1	17.300	24.500	16.300	23.500	15.800	23.000
	(s)=128, (d)=40, (n)=48	15.100	22.400	14.100	21.400	13.600	20.900
LDE=	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LDE<>	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LDE>	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LDE<=	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LDE<	Continuity/Non-continuity	0.18816		0.02352		0.00588	
LDE>=	Continuity/Non-continuity	0.18816		0.02352		0.00588	
ANDE=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ANDE<>	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ANDE>	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
ANDE<=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ANDE<	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ANDE>=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORE=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORE<>	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORE>	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORE<=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORE<	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
ORE>=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	0.18816		0.02352		0.00588	
LDED=	Continuity/Non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
LDED<>	Continuity/Non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
LDED>	Continuity/Non-continuity	3.300	5.900	2.300	4.900	1.800	4.400
LDED<=	Continuity/Non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
LDED<	Continuity/Non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
LDED>=	Continuity/Non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
ANDED=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
ANDED<>	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
ANDED>	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
ANDED<=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
ANDED<	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
ANDED>=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
ORED=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.400	6.000	2.400	5.000	1.900	4.500

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
ORED<>	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
ORED>	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
ORED<=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
ORED<	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
ORED>=	When not executed	0.18816		0.02352		0.00588	
	When executed: continuity/non-continuity	3.400	6.000	2.400	5.000	1.900	4.500
ECMP	—	4.400	7.400	3.400	6.400	2.900	5.900
EDCMP	—	4.600	8.900	3.600	7.900	3.100	7.400
EZCP	—	4.800	8.100	3.800	7.100	3.300	6.600
EDZCP	—	5.200	10.100	4.200	9.100	3.700	8.600
E+ (s) (d)	(s)=0, (d)=0	0.31360		0.03920		0.0098	
	(s)=2^{127}, (d)=2^{127}	0.31360		0.03920		0.0098	
E+ (s1) (s2) (d)	(s1)=0, (s2)=0	0.31360		0.03920		0.0098	
	(s1)=2^{127}, (s2)=2^{127}	0.31360		0.03920		0.0098	
E- (s) (d)	(s)=0, (d)=0	0.31360		0.03920		0.0098	
	(s)=2^{127}, (d)=2^{127}	0.31360		0.03920		0.0098	
E- (s1) (s2) (d)	(s1)=0, (s2)=0	0.31360		0.03920		0.0098	
	(s1)=2^{127}, (s2)=2^{127}	0.31360		0.03920		0.0098	
ED+ (s) (d)	(s)=0, (d)=0	3.100	7.300	2.100	6.300	1.600	5.800
	(s)=2^{1023}, (d)=2^{1023}	3.500	8.500	2.500	7.500	2.000	7.000
ED+ (s1) (s2) (d)	(s1)=0, (s2)=0	3.500	7.200	2.500	6.200	2.000	5.700
	(s1)=2^{1023}, (s2)=2^{1023}	3.700	8.500	2.700	7.500	2.200	7.000
ED- (s) (d)	(s)=0, (d)=0	3.300	7.000	2.300	6.000	1.800	5.500
	(s)=2^{1023}, (d)=2^{1023}	3.400	7.200	2.400	6.200	1.900	5.700
ED- (s1) (s2) (d)	(s1)=0, (s2)=0	3.500	7.000	2.500	6.000	2.000	5.500
	(s1)=2^{1023}, (s2)=2^{1023}	3.600	7.400	2.600	6.400	2.100	5.900
E*	(s1)=0, (s2)=0	0.31360		0.03920		0.0098	
	(s1)=2^{127}, (s2)=2^{127}	0.31360		0.03920		0.0098	
E/	(s1)=2^{127}, (s2)=2^{127}	18.18880		2.27360		0.05684	
ED*	(s1)=0, (s2)=0	3.500	7.400	2.500	6.400	2.000	5.900
	(s1)=2^{1023}, (s2)=2^{1023}	3.800	8.800	2.800	7.800	2.300	7.300
ED/	(s1)=2^{1023}, (s2)=2^{1023}	3.800	8.800	2.800	7.800	2.300	7.300
INT2FLT	(s)=0	0.21952		0.02744		0.00686	
	(s)=7FFFH	0.21952		0.02744		0.00686	
UINT2FLT	(s)=0	0.21952		0.02744		0.00686	
	(s)=FFFFH	0.21952		0.02744		0.00686	
DINT2FLT	(s)=0	0.21952		0.02744		0.00686	
	(s)=7FFFFFFFH	0.21952		0.02744		0.00686	
UDINT2FLT	(s)=0	0.21952		0.02744		0.00686	
	(s)=FFFFFFFFH	0.21952		0.02744		0.00686	
DBL2FLT	—	3.100	5.500	2.100	4.500	1.600	4.000
INT2DBL	(s)=0	2.900	4.600	1.900	3.600	1.400	3.100
	(s)=7FFFH	2.900	4.600	1.900	3.600	1.400	3.100

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
UINT2DBL	(s)=0	2.900	4.800	1.900	3.800	1.400	3.300
	(s)=FFFFH	2.900	4.700	1.900	3.700	1.400	3.200
DINT2DBL	(s)=0	2.900	4.600	1.900	3.600	1.400	3.100
	(s)=7FFFFFFFH	2.900	4.600	1.900	3.600	1.400	3.100
UDINT2DBL	(s)=0	2.900	4.700	1.900	3.700	1.400	3.200
	(s)=FFFFFFFFH	2.900	4.700	1.900	3.700	1.400	3.200
FLT2DBL	—	2.900	6.100	1.900	5.100	1.400	4.600
EVAL	Decimal point format 2-digit full specification	5.200	13.800	4.200	12.800	3.700	12.300
	Number of digits format 6-digit full specification	5.600	13.500	4.600	12.500	4.100	12.000
EREXP	—	4.600	10.900	3.600	9.900	3.100	9.400
ENEG	(d)=0	2.700	3.700	1.700	2.700	1.200	2.200
	(d)=-1.0	2.800	4.400	1.800	3.400	1.300	2.900
EDNEG	(d)=0	2.800	5.800	1.800	4.800	1.300	4.300
	(d)=-1.0	2.600	5.900	1.600	4.900	1.100	4.400
EMOV	—	0.06272		0.00784		0.00196	
EDMOV	—	0.06272		0.00784		0.00196	
SIN	—	2.900	5.800	1.900	4.800	1.400	4.300
COS	—	2.900	5.700	1.900	4.700	1.400	4.200
TAN	—	2.900	5.700	1.900	4.700	1.400	4.200
ASIN	—	2.900	6.000	1.900	5.000	1.400	4.500
ACOS	—	2.900	6.000	1.900	5.000	1.400	4.500
ATAN	—	2.900	4.900	1.900	3.900	1.400	3.400
SIND	—	3.900	14.200	2.900	13.200	2.400	12.700
COSD	—	3.800	14.200	2.800	13.200	2.300	12.700
TAND	—	4.300	15.400	3.300	14.400	2.800	13.900
ASIND	—	3.900	12.000	2.900	11.000	2.400	10.500
ACOSD	—	3.700	11.200	2.700	10.200	2.200	9.700
ATAND	—	3.500	10.700	2.500	9.700	2.000	9.200
BSIN	—	4.100	10.800	3.100	9.800	2.600	9.300
BCOS	—	4.100	10.400	3.100	9.400	2.600	8.900
BTAN	—	4.200	11.100	3.200	10.100	2.700	9.600
BASIN	—	3.900	8.800	2.900	7.800	2.400	7.300
BACOS	—	4.000	8.900	3.000	7.900	2.500	7.400
BATAN	—	3.900	8.900	2.900	7.900	2.400	7.400
RAD	—	2.900	4.200	1.900	3.200	1.400	2.700
DEG	—	2.900	4.400	1.900	3.400	1.400	2.900
RADD	—	3.100	8.800	2.100	7.800	1.600	7.300
DEGD	—	3.100	8.600	2.100	7.600	1.600	7.100
ESQRT	—	2.600	4.400	1.600	3.400	1.100	2.900
EDSQRT	—	3.200	8.700	2.200	7.700	1.700	7.200
EXP	(s)=-10	2.900	6.000	1.900	5.000	1.400	4.500
	(s)=1	3.000	6.100	2.000	5.100	1.500	4.600
EXPD	(s)=-10	3.700	12.000	2.700	11.000	2.200	10.500
	(s)=1	3.600	11.800	2.600	10.800	2.100	10.300
LOG	(s)=1	2.800	5.600	1.800	4.600	1.300	4.100
	(s)=10	2.900	6.200	1.900	5.200	1.400	4.700
LOGD	(s)=1	3.300	10.100	2.300	9.100	1.800	8.600
	(s)=10	3.800	12.400	2.800	11.400	2.300	10.900

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
BSQRT	(s)=0	3.000	4.600	2.000	3.600	1.500	3.100
	(s)=9999	3.700	8.400	2.700	7.400	2.200	6.900
BDSQRT	(s)=0	2.900	4.100	1.900	3.100	1.400	2.600
	(s)=99999999	3.600	7.500	2.600	6.500	2.100	6.000
POW	(s1)=1.23E+5, (s2)=3.45E+0	4.500	10.400	3.500	9.400	3.000	8.900
POWD	(s1)=1.23E+5, (s2)=3.45E+0	5.900	20.800	4.900	19.800	4.400	19.300
LOG10	(s)=1.23E+20	3.000	6.200	2.000	5.200	1.500	4.700
LOG10D	(s)=1.23E+20	3.900	14.400	2.900	13.400	2.400	12.900
EMAX	(n)=1	4.000	6.600	3.000	5.600	2.500	5.100
	(n)=96	16.200	19.400	15.200	18.400	14.700	17.900
EDMAX	(n)=1	4.200	8.400	3.200	7.400	2.700	6.900
	(n)=96	29.200	34.000	28.200	33.000	27.700	32.500
EMIN	(n)=1	4.000	6.600	3.000	5.600	2.500	5.100
	(n)=96	16.200	19.400	15.200	18.400	14.700	17.900
EDMIN	(n)=1	4.200	8.300	3.200	7.300	2.700	6.800
	(n)=96	28.700	33.900	27.700	32.900	27.200	32.400
RND	—	2.300	3.200	1.300	2.200	0.800	1.700
SRND	—	2.600	3.300	1.600	2.300	1.100	1.800
ZPUSH (d)	—	3.300	3.700	1.200	1.900	0.900	1.700
ZPUSH (s) (d)	All areas of Z and LZ	6.300	6.900	2.000	3.700	1.600	3.200
	All area of Z (default: 20 points)	6.300	6.900	2.000	3.500	1.600	3.100
	All area of LZ (default: 2 points)	5.100	5.800	2.000	3.600	1.300	3.100
ZPOP (d)	—	3.300	3.800	1.800	2.700	0.900	1.800
ZPOP (s) (d)	All areas of Z and LZ	5.800	6.700	1.900	3.700	1.500	3.200
	All area of Z (default: 20 points)	5.800	6.600	1.900	3.800	1.500	3.100
	All area of LZ (default: 2 points)	4.900	5.700	1.800	3.800	1.300	3.100
LIMIT	—	3.700	4.300	1.600	2.800	1.000	1.800
LIMIT_U	—	3.700	4.300	1.600	2.800	1.000	1.800
DLIMIT	—	3.700	4.300	1.600	2.800	1.000	1.800
DLIMIT_U	—	3.700	4.300	1.600	2.800	1.000	1.800
BAND	—	3.300	4.600	2.300	3.600	1.800	3.100
BAND_U	—	3.300	4.600	2.300	3.600	1.800	3.100
DBAND	—	3.400	4.500	2.400	3.500	1.900	3.000
DBAND_U	—	3.400	4.400	2.400	3.400	1.900	2.900
ZONE	—	3.300	4.500	2.300	3.500	1.800	3.000
ZONE_U	—	3.300	4.500	2.300	3.500	1.800	3.000
DZONE	—	3.400	4.400	2.400	3.400	1.900	2.900
DZONE_U	—	3.400	4.500	2.400	3.500	1.900	3.000
SCL	SM755=ON, point No.1<(s1)<point No.2	4.800	10.400	3.800	9.400	3.300	8.900
	SM755=ON, point No.9<(s1)<point No.10	4.800	10.400	3.800	9.400	3.300	8.900
	SM755=OFF, point No.1<(s1)<point No.2	4.600	10.200	3.600	9.200	3.100	8.700
	SM755=OFF, point No.9<(s1)<point No.10	5.200	10.400	4.200	9.400	3.700	8.900

A

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
SCL_U	SM755=ON, point No.1<(s1)<point No.2	4.700	10.200	3.700	9.200	3.200	8.700
	SM755=ON, point No.9<(s1)<point No.10	4.700	10.300	3.700	9.300	3.200	8.800
	SM755=OFF, point No.1<(s1)<point No.2	4.500	10.000	3.500	9.000	3.000	8.500
	SM755=OFF, point No.9<(s1)<point No.10	4.900	10.400	3.900	9.400	3.400	8.900
DSCL	SM755=ON, point No.1<(s1)<point No.2	4.800	11.400	3.800	10.400	3.300	9.900
	SM755=ON, point No.9<(s1)<point No.10	4.800	11.300	3.800	10.300	3.300	9.800
	SM755=OFF, point No.1<(s1)<point No.2	4.500	10.800	3.500	9.800	3.000	9.300
	SM755=OFF, point No.9<(s1)<point No.10	5.100	11.200	4.100	10.200	3.600	9.700
DSCL_U	SM755=ON, point No.1<(s1)<point No.2	4.700	11.700	3.700	10.700	3.200	10.200
	SM755=ON, point No.9<(s1)<point No.10	4.700	11.700	3.700	10.700	3.200	10.200
	SM755=OFF, point No.1<(s1)<point No.2	4.400	11.000	3.400	10.000	2.900	9.500
	SM755=OFF, point No.9<(s1)<point No.10	5.000	11.400	4.000	10.400	3.500	9.900
SCL2	SM755=ON, point No.1<(s1)<point No.2	4.800	10.700	3.800	9.700	3.300	9.200
	SM755=ON, point No.9<(s1)<point No.10	4.800	10.600	3.800	9.600	3.300	9.100
	SM755=OFF, point No.1<(s1)<point No.2	4.800	10.200	3.800	9.200	3.300	8.700
	SM755=OFF, point No.9<(s1)<point No.10	5.000	10.300	4.000	9.300	3.500	8.800
SCL2_U	SM755=ON, point No.1<(s1)<point No.2	4.700	10.300	3.700	9.300	3.200	8.800
	SM755=ON, point No.9<(s1)<point No.10	4.800	10.400	3.800	9.400	3.300	8.900
	SM755=OFF, point No.1<(s1)<point No.2	4.400	10.000	3.400	9.000	2.900	8.500
	SM755=OFF, point No.9<(s1)<point No.10	5.000	10.500	4.000	9.500	3.500	9.000
DSCL2	SM755=ON, point No.1<(s1)<point No.2	4.800	11.500	3.800	10.500	3.300	10.000
	SM755=ON, point No.9<(s1)<point No.10	4.800	11.500	3.800	10.500	3.300	10.000
	SM755=OFF, point No.1<(s1)<point No.2	4.500	11.000	3.500	10.000	3.000	9.500
	SM755=OFF, point No.9<(s1)<point No.10	4.800	11.600	3.800	10.600	3.300	10.100
DSCL2_U	SM755=ON, point No.1<(s1)<point No.2	4.700	11.600	3.700	10.600	3.200	10.100
	SM755=ON, point No.9<(s1)<point No.10	4.800	11.600	3.800	10.600	3.300	10.100
	SM755=OFF, point No.1<(s1)<point No.2	4.400	11.100	3.400	10.100	2.900	9.600
	SM755=OFF, point No.9<(s1)<point No.10	5.000	11.500	4.000	10.500	3.500	10.000
UDCNT1	—	2.300	3.100	1.300	2.100	0.800	1.600
UDCNT2	—	2.300	3.100	1.300	2.100	0.800	1.600

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
TTMR	—	3.500	7.400	2.500	6.400	2.000	5.900
STMR	—	4.200	9.600	3.200	8.600	2.700	8.100
ROTC	—	6.300	10.100	5.300	9.100	4.800	8.600
RAMPQ	—	4.900	9.700	3.900	8.700	3.400	8.200
SPD	—	2.200	3.100	1.200	2.100	0.700	1.600
PLSY	—	2.300	2.900	1.300	1.900	0.800	1.400
PWM	—	2.300	2.900	1.300	1.900	0.800	1.400
MTR	—	4.900	13.100	3.900	12.100	3.400	11.600
CCD	SM772=OFF, (n)=1	4.800	8.600	3.800	7.600	3.300	7.100
	SM772=OFF, (n)=96	13.200	17.000	12.200	16.000	11.700	15.500
	SM772=ON, (n)=1	4.800	8.600	3.800	7.600	3.300	7.100
	SM772=ON, (n)=96	13.200	17.000	12.200	16.000	11.700	15.500
SERDATA	(n)=1: All match	4.200	8.200	3.200	7.200	2.700	6.700
	(n)=1: All mismatch	4.200	8.100	3.200	7.100	2.700	6.600
	(n)=96: All match	11.200	16.500	10.200	15.500	9.700	15.000
	(n)=96: All mismatch	11.200	16.600	10.200	15.600	9.700	15.100
DSERDATA	(n)=1: All match	4.500	8.800	3.500	7.800	3.000	7.300
	(n)=1: All mismatch	4.500	8.800	3.500	7.800	3.000	7.300
	(n)=96: All match	16.000	20.400	15.000	19.400	14.500	18.900
	(n)=96: All mismatch	16.000	20.400	15.000	19.400	14.500	18.900
SERMM	(n)=1, no matched data	5.600	10.800	4.600	9.800	4.100	9.300
	(n)=1, number of the matched data = 1	5.600	10.800	4.600	9.800	4.100	9.300
	(n)=96, no matched data	14.600	19.700	13.600	18.700	13.100	18.200
	(n)=96, number of the matched data = 96	14.600	19.700	13.600	18.700	13.100	18.200
DSERMM	(n)=1, no matched data	5.600	10.800	4.600	9.800	4.100	9.300
	(n)=1, number of the matched data = 1	5.600	10.800	4.600	9.800	4.100	9.300
	(n)=96, no matched data	14.600	19.700	13.600	18.700	13.100	18.200
	(n)=96, number of the matched data = 96	14.600	19.700	13.600	18.700	13.100	18.200
SUM	(s)=0	5.000	5.500	1.600	2.200	1.300	1.900
	(s)=FFFFH	5.000	5.500	1.600	2.200	1.300	1.900
DSUM	(s)=0	7.600	8.000	2.000	2.700	1.700	2.400
	(s)=FFFFFFFFH	7.500	7.900	2.000	2.700	1.700	2.400
BON	(s)=0	3.900	6.400	2.900	5.400	2.400	4.900
	(s)=FFFFH	3.900	6.400	2.900	5.400	2.400	4.900
DBON	(s)=0	3.900	6.400	2.900	5.400	2.400	4.900
	(s)=FFFFFFFFH	3.900	6.400	2.900	5.400	2.400	4.900
MAX	(n)=1	3.900	6.500	2.900	5.500	2.400	5.000
	(n)=96	11.000	13.800	10.000	12.800	9.500	12.300
MAX_U	(n)=1	4.000	7.600	3.000	6.600	2.500	6.100
	(n)=96	12.200	16.200	11.200	15.200	10.700	14.700
DMAX	(n)=1	4.000	6.800	3.000	5.800	2.500	5.300
	(n)=96	18.900	21.900	17.900	20.900	17.400	20.400
DMAX_U	(n)=1	4.000	7.300	3.000	6.300	2.500	5.800
	(n)=96	11.800	15.600	10.800	14.600	10.300	14.100
MIN	(n)=1	3.900	6.500	2.900	5.500	2.400	5.000
	(n)=96	11.000	13.900	10.000	12.900	9.500	12.400
MIN_U	(n)=1	4.000	7.400	3.000	6.400	2.500	5.900
	(n)=96	12.200	15.900	11.200	14.900	10.700	14.400

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
DMIN	(n)=1	4.000	6.900	3.000	5.900	2.500	5.400
	(n)=96	19.000	22.000	18.000	21.000	17.500	20.500
DMIN_U	(n)=1	3.900	7.400	2.900	6.400	2.400	5.900
	(n)=96	11.800	15.500	10.800	14.500	10.300	14.000
SORTD	(n)=1, (s1)=1	4.600	7.100	3.600	6.100	3.100	5.600
	(n)=96, (s1)=16	9.900	13.400	8.900	12.400	8.400	11.900
SORTD_U	(n)=1, (s1)=1	4.600	7.100	3.600	6.100	3.100	5.600
	(n)=96, (s1)=16	10.300	14.400	9.300	13.400	8.800	12.900
DSORTD	(n)=1, (s1)=1	4.500	7.000	3.500	6.000	3.000	5.500
	(n)=96, (s1)=16	11.400	15.000	10.400	14.000	9.900	13.500
DSORTD_U	(n)=1, (s1)=1	4.600	7.300	3.600	6.300	3.100	5.800
	(n)=96, (s1)=16	12.600	17.300	11.600	16.300	11.100	15.800
SORTTBL	(n1)=1, (n2)=1	13.100	15.500	12.100	14.500	11.600	14.000
	(n1)=32, (n2)=6	13.100	15.500	12.100	14.500	11.600	14.000
SORTTBL_U	(n1)=1, (n2)=1	13.100	15.500	12.100	14.500	11.600	14.000
	(n1)=32, (n2)=6	13.100	15.500	12.100	14.500	11.600	14.000
SORTTBL2	(n1)=1, (n2)=1	13.100	15.500	12.100	14.500	11.600	14.000
	(n1)=32, (n2)=6	13.100	15.500	12.100	14.500	11.600	14.000
SORTTBL2_U	(n1)=1, (n2)=1	13.100	15.500	12.100	14.500	11.600	14.000
	(n1)=32, (n2)=6	13.100	15.500	12.100	14.500	11.600	14.000
DSORTTBL2	(n1)=1, (n2)=1	13.100	15.500	12.100	14.500	11.600	14.000
	(n1)=32, (n2)=6	13.100	15.500	12.100	14.500	11.600	14.000
DSORTTBL2_U	(n1)=1, (n2)=1	13.100	15.500	12.100	14.500	11.600	14.000
	(n1)=32, (n2)=6	13.100	15.500	12.100	14.500	11.600	14.000
WSUM	(n)=1	3.700	4.100	1.300	2.200	0.900	1.800
	(n)=96	24.500	25.000	6.000	6.700	4.700	5.600
WSUM_U	(n)=1	3.600	4.000	1.200	2.200	0.900	1.800
	(n)=96	16.100	16.600	5.500	6.300	4.700	5.600
DWSUM	(n)=1	3.800	8.500	2.800	7.500	2.300	7.000
	(n)=96	10.900	15.600	9.900	14.600	9.400	14.100
DWSUM_U	(n)=1	3.900	9.300	2.900	8.300	2.400	7.800
	(n)=96	11.000	16.400	10.000	15.400	9.500	14.900
MEAN	(n)=1	3.400	6.400	2.400	5.400	1.900	4.900
	(n)=96	6.600	10.300	5.600	9.300	5.100	8.800
MEAN_U	(n)=1	3.400	6.300	2.400	5.300	1.900	4.800
	(n)=96	6.200	10.200	5.200	9.200	4.700	8.700
DMEAN	(n)=1	4.100	9.300	3.100	8.300	2.600	7.800
	(n)=96	10.400	15.700	9.400	14.700	8.900	14.200
DMEAN_U	(n)=1	3.700	9.000	2.700	8.000	2.200	7.500
	(n)=96	9.800	14.900	8.800	13.900	8.300	13.400
SQRT	—	3.300	4.800	2.300	3.800	1.800	3.300
DSQRT	—	3.300	4.800	2.300	3.800	1.800	3.300
CRC	SM772=OFF, (n)=1	4.700	9.400	3.700	8.400	3.200	7.900
	SM772=OFF, (n)=96	13.700	20.300	12.700	19.300	12.200	18.800
	SM772=ON, (n)=1	4.700	9.400	3.700	8.400	3.200	7.900
	SM772=ON, (n)=96	13.700	20.300	12.700	19.300	12.200	18.800
DBOPEN	When executed (for the programmable controller CPU with firmware version earlier than "28")	—	—	—	—	18.300	20.900
	When executed (for the programmable controller CPU with firmware version "28" or later)	—	—	—	—	19.300	24.500

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
DBCLOSE	When executed (for the programmable controller CPU with firmware version earlier than "28")	—	—	—	—	11.500	13.100
	When executed (for the programmable controller CPU with firmware version "28" or later)	—	—	—	—	13.300	18.300
DBINSERT	(s3)=1 (for the programmable controller CPU with firmware version earlier than "28")	—	—	—	—	25.400	28.900
	(s3)=16 (for the programmable controller CPU with firmware version earlier than "28")	—	—	—	—	46.400	50.100
	(s3)=1 (for the programmable controller CPU with firmware version "28" or later)	—	—	—	—	23.300	29.100
	(s3)=16 (for the programmable controller CPU with firmware version "28" or later)	—	—	—	—	40.200	45.800
DBUPDATE	(s3)=1 (for the programmable controller CPU with firmware version earlier than "28")	—	—	—	—	31.700	34.700
	(s3)=16 (for the programmable controller CPU with firmware version earlier than "28")	—	—	—	—	49.300	52.900
	(s3)=1 (for the programmable controller CPU with firmware version "28" or later)	—	—	—	—	29.900	35.900
	(s3)=16 (for the programmable controller CPU with firmware version "28" or later)	—	—	—	—	46.900	53.200
DBSELECT	(s3)=1 (for the programmable controller CPU with firmware version earlier than "28")	—	—	—	—	32.100	35.300
	(s3)=16 (for the programmable controller CPU with firmware version earlier than "28")	—	—	—	—	51.600	55.000
	(s3)=1 (for the programmable controller CPU with firmware version "28" or later)	—	—	—	—	32.800	38.300
	(s3)=16 (for the programmable controller CPU with firmware version "28" or later)	—	—	—	—	48.900	55.200
DBDELETE	(s3)=1 (for the programmable controller CPU with firmware version earlier than "28")	—	—	—	—	26.000	29.900
	(s3)=2 (for the programmable controller CPU with firmware version earlier than "28")	—	—	—	—	27.400	31.200
	(s3)=1 (for the programmable controller CPU with firmware version "28" or later)	—	—	—	—	24.800	30.900
	(s3)=2 (for the programmable controller CPU with firmware version "28" or later)	—	—	—	—	26.600	32.500
DBIMPORT	When executed (for the programmable controller CPU with firmware version earlier than "28")	—	—	—	—	16.200	19.200
	When executed (for the programmable controller CPU with firmware version "28" or later)	—	—	—	—	17.800	22.400

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
DBEXPORT	When executed (for the programmable controller CPU with firmware version earlier than "28")	—	—	—	—	16.000	19.000
	When executed (for the programmable controller CPU with firmware version "28" or later)	—	—	—	—	17.300	22.500
DBTRANS	When executed (for the programmable controller CPU with firmware version earlier than "28")	—	—	—	—	11.400	13.000
	When executed (for the programmable controller CPU with firmware version "28" or later)	—	—	—	—	12.800	18.500
DBCOMMIT	When executed (for the programmable controller CPU with firmware version earlier than "28")	—	—	—	—	11.600	13.000
	When executed (for the programmable controller CPU with firmware version "28" or later)	—	—	—	—	12.800	18.500
DBROLBAK	When executed (for the programmable controller CPU with firmware version earlier than "28")	—	—	—	—	11.500	12.900
	When executed (for the programmable controller CPU with firmware version "28" or later)	—	—	—	—	12.800	18.400
RSET	—	3.300	7.800	2.300	6.800	1.800	6.300
QDRSET	—	66.600	93.600	65.600	92.600	65.100	92.100
ZRRDB	—	3.100	4.400	2.100	3.400	1.600	2.900
ZRWRB	—	3.200	4.700	2.200	3.700	1.700	3.200
ADRSET	—	2.600	3.500	1.600	2.500	1.100	2.000
DATERD	—	4.900	12.200	3.900	11.200	3.400	10.700
DATEWR	—	13.700	38.900	12.700	37.900	12.200	37.400
DATE+	No carry	4.800	8.300	3.800	7.300	3.300	6.800
	Carry	4.900	8.000	3.900	7.000	3.400	6.500
DATE-	No carry	4.900	8.000	3.900	7.000	3.400	6.500
	Carry	4.900	8.100	3.900	7.100	3.400	6.600
TIME2SEC	—	3.400	5.200	2.400	4.200	1.900	3.700
SEC2TIME	—	3.300	5.400	2.300	4.400	1.800	3.900
DATE2SEC	(s)+0=2067, (s)+1=12, (s)+2=31, (s)+3=23, (s)+4=59, (s)+5=59	4.700	8.600	3.700	7.600	3.300	7.400
DATE2SEC_U	(s)+0=2099, (s)+1=12, (s)+2=31, (s)+3=23, (s)+4=59, (s)+5=59	4.700	8.600	3.700	7.600	3.300	7.400
SEC2DATE	(s)=2145916799	4.700	12.200	3.700	11.200	3.100	10.800
SEC2DATE_U	(s)=3155759999	4.700	12.200	3.700	11.200	3.100	10.800
LDDT=	When compared with the specified date: Continuity	4.400	11.000	3.400	10.000	2.900	9.500
	When compared with the specified date: Non-continuity	4.400	10.900	3.400	9.900	2.900	9.400
	When compared with the current date: Continuity	6.200	16.500	5.200	15.500	4.700	15.000
	When compared with the current date: Non-continuity	6.300	16.600	5.300	15.600	4.800	15.100

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
LDDT<>	When compared with the specified date: Continuity	4.100	10.800	3.100	9.800	2.600	9.300
	When compared with the specified date: Non-continuity	4.100	10.900	3.100	9.900	2.600	9.400
	When compared with the current date: Continuity	6.100	17.100	5.100	16.100	4.600	15.600
	When compared with the current date: Non-continuity	6.300	17.000	5.300	16.000	4.800	15.500
LDDT>	When compared with the specified date: Continuity	4.200	11.000	3.200	10.000	2.700	9.500
	When compared with the specified date: Non-continuity	4.200	11.200	3.200	10.200	2.700	9.700
	When compared with the current date: Continuity	6.100	17.000	5.100	16.000	4.600	15.500
	When compared with the current date: Non-continuity	6.100	16.800	5.100	15.800	4.600	15.300
LDDT<=	When compared with the specified date: Continuity	4.200	11.400	3.200	10.400	2.700	9.900
	When compared with the specified date: Non-continuity	4.300	11.300	3.300	10.300	2.800	9.800
	When compared with the current date: Continuity	6.300	16.100	5.300	15.100	4.800	14.600
	When compared with the current date: Non-continuity	6.400	15.900	5.400	14.900	4.900	14.400
LDDT<	When compared with the specified date: Continuity	4.300	11.400	3.300	10.400	2.800	9.900
	When compared with the specified date: Non-continuity	4.200	11.400	3.200	10.400	2.700	9.900
	When compared with the current date: Continuity	6.300	16.500	5.300	15.500	4.800	15.000
	When compared with the current date: Non-continuity	6.300	16.400	5.300	15.400	4.800	14.900
LDDT>=	When compared with the specified date: Continuity	4.200	11.300	3.200	10.300	2.700	9.800
	When compared with the specified date: Non-continuity	4.300	11.300	3.300	10.300	2.800	9.800
	When compared with the current date: Continuity	6.500	16.000	5.500	15.000	5.000	14.500
	When compared with the current date: Non-continuity	6.400	16.200	5.400	15.200	4.900	14.700
ANDDT=	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified date: Continuity	3.900	11.500	2.900	10.500	2.400	10.000
	When compared with the specified date: Non-continuity	4.000	11.600	3.000	10.600	2.500	10.100
	When compared with the current date: Continuity	6.100	17.400	5.100	16.400	4.600	15.900
	When compared with the current date: Non-continuity	6.000	17.400	5.000	16.400	4.500	15.900
ANDDT<>	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified date: Continuity	4.400	10.900	3.400	9.900	2.900	9.400
	When compared with the specified date: Non-continuity	4.400	10.900	3.400	9.900	2.900	9.400
	When compared with the current date: Continuity	6.300	17.300	5.300	16.300	4.800	15.800
	When compared with the current date: Non-continuity	6.400	17.300	5.400	16.300	4.900	15.800

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
ANDDT>	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified date: Continuity	4.200	11.200	3.200	10.200	2.700	9.700
	When compared with the specified date: Non-continuity	4.200	11.500	3.200	10.500	2.700	10.000
	When compared with the current date: Continuity	6.200	17.100	5.200	16.100	4.700	15.600
	When compared with the current date: Non-continuity	6.200	17.100	5.200	16.100	4.700	15.600
ANDDT<=	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified date: Continuity	4.200	11.300	3.200	10.300	2.700	9.800
	When compared with the specified date: Non-continuity	4.300	11.400	3.300	10.400	2.800	9.900
	When compared with the current date: Continuity	6.400	16.100	5.400	15.100	4.900	14.600
	When compared with the current date: Non-continuity	6.400	16.000	5.400	15.000	4.900	14.500
ANDDT<	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified date: Continuity	4.100	10.900	3.100	9.900	2.600	9.400
	When compared with the specified date: Non-continuity	4.100	11.100	3.100	10.100	2.600	9.600
	When compared with the current date: Continuity	6.300	17.700	5.300	16.700	4.800	16.200
	When compared with the current date: Non-continuity	6.400	16.700	5.400	15.700	4.900	15.200
ANDDT>=	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified date: Continuity	4.200	11.400	3.200	10.400	2.700	9.900
	When compared with the specified date: Non-continuity	4.200	11.200	3.200	10.200	2.700	9.700
	When compared with the current date: Continuity	6.400	16.100	5.400	15.100	4.900	14.600
	When compared with the current date: Non-continuity	6.400	16.400	5.400	15.400	4.900	14.900
ORDT=	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified date: Continuity	4.200	11.300	3.200	10.300	2.700	9.800
	When compared with the specified date: Non-continuity	4.100	11.300	3.100	10.300	2.600	9.800
	When compared with the current date: Continuity	6.300	17.100	5.300	16.100	4.800	15.600
	When compared with the current date: Non-continuity	6.300	16.900	5.300	15.900	4.800	15.400
ORDT<>	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified date: Continuity	4.300	11.500	3.300	10.500	2.800	10.000
	When compared with the specified date: Non-continuity	4.200	11.600	3.200	10.600	2.700	10.100
	When compared with the current date: Continuity	6.200	17.400	5.200	16.400	4.700	15.900
	When compared with the current date: Non-continuity	6.300	17.100	5.300	16.100	4.800	15.600

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
ORDT>	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified date: Continuity	4.300	11.500	3.300	10.500	2.800	10.000
	When compared with the specified date: Non-continuity	4.200	11.500	3.200	10.500	2.700	10.000
	When compared with the current date: Continuity	6.300	17.100	5.300	16.100	4.800	15.600
	When compared with the current date: Non-continuity	6.300	17.200	5.300	16.200	4.800	15.700
ORDT<=	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified date: Continuity	4.200	11.300	3.200	10.300	2.700	9.800
	When compared with the specified date: Non-continuity	4.100	11.500	3.100	10.500	2.600	10.000
	When compared with the current date: Continuity	6.400	16.500	5.400	15.500	4.900	15.000
	When compared with the current date: Non-continuity	6.400	16.500	5.400	15.500	4.900	15.000
ORDT<	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified date: Continuity	4.300	11.700	3.300	10.700	2.800	10.200
	When compared with the specified date: Non-continuity	4.200	11.600	3.200	10.600	2.700	10.100
	When compared with the current date: Continuity	6.300	17.000	5.300	16.000	4.800	15.500
	When compared with the current date: Non-continuity	6.400	16.900	5.400	15.900	4.900	15.400
ORDT>=	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified date: Continuity	4.300	11.000	3.300	10.000	2.800	9.500
	When compared with the specified date: Non-continuity	4.300	11.100	3.300	10.100	2.800	9.600
	When compared with the current date: Continuity	6.200	16.000	5.200	15.000	4.700	14.500
	When compared with the current date: Non-continuity	6.200	16.400	5.200	15.400	4.700	14.900
LDTM=	When compared with the specified time: Continuity	4.400	11.000	3.400	10.000	2.900	9.500
	When compared with the specified time: Non-continuity	4.400	11.000	3.400	10.000	2.900	9.500
	When compared with the current time: Continuity	6.200	17.100	5.200	16.100	4.700	15.600
	When compared with the current time: Non-continuity	6.200	16.400	5.200	15.400	4.700	14.900
LDTM<>	When compared with the specified time: Continuity	4.100	10.700	3.100	9.700	2.600	9.200
	When compared with the specified time: Non-continuity	4.100	10.800	3.100	9.800	2.600	9.300
	When compared with the current time: Continuity	6.400	16.000	5.400	15.000	4.900	14.500
	When compared with the current time: Non-continuity	6.300	16.000	5.300	15.000	4.800	14.500

A

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
LDTM>	When compared with the specified time: Continuity	4.100	10.700	3.100	9.700	2.600	9.200
	When compared with the specified time: Non-continuity	4.100	10.800	3.100	9.800	2.600	9.300
	When compared with the current time: Continuity	6.300	16.100	5.300	15.100	4.800	14.600
	When compared with the current time: Non-continuity	6.400	16.200	5.400	15.200	4.900	14.700
LDTM<=	When compared with the specified time: Continuity	4.100	10.700	3.100	9.700	2.600	9.200
	When compared with the specified time: Non-continuity	4.200	10.700	3.200	9.700	2.700	9.200
	When compared with the current time: Continuity	6.400	16.000	5.400	15.000	4.900	14.500
	When compared with the current time: Non-continuity	6.400	16.100	5.400	15.100	4.900	14.600
LDTM<	When compared with the specified time: Continuity	4.100	10.800	3.100	9.800	2.600	9.300
	When compared with the specified time: Non-continuity	4.100	10.800	3.100	9.800	2.600	9.300
	When compared with the current time: Continuity	6.200	16.300	5.200	15.300	4.700	14.800
	When compared with the current time: Non-continuity	6.400	16.200	5.400	15.200	4.900	14.700
LDTM>=	When compared with the specified time: Continuity	4.000	10.600	3.000	9.600	2.500	9.100
	When compared with the specified time: Non-continuity	4.100	10.700	3.100	9.700	2.600	9.200
	When compared with the current time: Continuity	6.400	16.000	5.400	15.000	4.900	14.500
	When compared with the current time: Non-continuity	6.300	16.100	5.300	15.100	4.800	14.600
ANDTM=	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified time: Continuity	3.900	10.600	2.900	9.600	2.400	9.100
	When compared with the specified time: Non-continuity	4.000	10.700	3.000	9.700	2.500	9.200
	When compared with the current time: Continuity	6.400	16.100	5.400	15.100	4.900	14.600
	When compared with the current time: Non-continuity	6.400	16.200	5.400	15.200	4.900	14.700
ANDTM<>	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified time: Continuity	3.900	10.600	2.900	9.600	2.400	9.100
	When compared with the specified time: Non-continuity	4.100	10.900	3.100	9.900	2.600	9.400
	When compared with the current time: Continuity	6.400	16.300	5.400	15.300	4.900	14.800
	When compared with the current time: Non-continuity	6.400	16.300	5.400	15.300	4.900	14.800

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
ANDTM>	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified time: Continuity	4.100	10.900	3.100	9.900	2.600	9.400
	When compared with the specified time: Non-continuity	4.100	11.000	3.100	10.000	2.600	9.500
	When compared with the current time: Continuity	6.300	16.200	5.300	15.200	4.800	14.700
	When compared with the current time: Non-continuity	6.400	16.300	5.400	15.300	4.900	14.800
ANDTM<=	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified time: Continuity	4.100	10.900	3.100	9.900	2.600	9.400
	When compared with the specified time: Non-continuity	4.100	10.900	3.100	9.900	2.600	9.400
	When compared with the current time: Continuity	6.400	16.500	5.400	15.500	4.900	15.000
	When compared with the current time: Non-continuity	6.400	16.400	5.400	15.400	4.900	14.900
ANDTM<	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified time: Continuity	4.100	11.000	3.100	10.000	2.600	9.500
	When compared with the specified time: Non-continuity	4.100	11.100	3.100	10.100	2.600	9.600
	When compared with the current time: Continuity	6.400	16.200	5.400	15.200	4.900	14.700
	When compared with the current time: Non-continuity	6.400	16.300	5.400	15.300	4.900	14.800
ANDTM>=	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified time: Continuity	4.100	10.900	3.100	9.900	2.600	9.400
	When compared with the specified time: Non-continuity	4.100	10.900	3.100	9.900	2.600	9.400
	When compared with the current time: Continuity	6.400	16.400	5.400	15.400	4.900	14.900
	When compared with the current time: Non-continuity	6.300	16.400	5.300	15.400	4.800	14.900
ORTM=	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified time: Continuity	4.200	10.700	3.200	9.700	2.700	9.200
	When compared with the specified time: Non-continuity	4.200	10.600	3.200	9.600	2.700	9.100
	When compared with the current time: Continuity	6.500	16.400	5.500	15.400	5.000	14.900
	When compared with the current time: Non-continuity	6.500	16.300	5.500	15.300	5.000	14.800
ORTM<>	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified time: Continuity	4.200	10.700	3.200	9.700	2.700	9.200
	When compared with the specified time: Non-continuity	4.200	10.700	3.200	9.700	2.700	9.200
	When compared with the current time: Continuity	6.500	16.500	5.500	15.500	5.000	15.000
	When compared with the current time: Non-continuity	6.500	16.500	5.500	15.500	5.000	15.000

A

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
ORTM>	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified time: Continuity	4.200	10.800	3.200	9.800	2.700	9.300
	When compared with the specified time: Non-continuity	4.300	10.800	3.300	9.800	2.800	9.300
	When compared with the current time: Continuity	6.600	16.600	5.600	15.600	5.100	15.100
	When compared with the current time: Non-continuity	6.600	16.600	5.600	15.600	5.100	15.100
ORTM<=	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified time: Continuity	4.200	10.700	3.200	9.700	2.700	9.200
	When compared with the specified time: Non-continuity	4.100	10.800	3.100	9.800	2.600	9.300
	When compared with the current time: Continuity	6.500	16.400	5.500	15.400	5.000	14.900
	When compared with the current time: Non-continuity	6.400	16.500	5.400	15.500	4.900	15.000
ORTM<	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified time: Continuity	4.200	10.900	3.200	9.900	2.700	9.400
	When compared with the specified time: Non-continuity	4.200	10.900	3.200	9.900	2.700	9.400
	When compared with the current time: Continuity	6.500	16.300	5.500	15.300	5.000	14.800
	When compared with the current time: Non-continuity	6.400	16.400	5.400	15.400	4.900	14.900
ORTM>=	When not executed	0.18816		0.02352		0.00588	
	When compared with the specified time: Continuity	4.200	10.700	3.200	9.700	2.700	9.200
	When compared with the specified time: Non-continuity	4.000	10.700	3.000	9.700	2.500	9.200
	When compared with the current time: Continuity	6.400	16.400	5.400	15.400	4.900	14.900
	When compared with the current time: Non-continuity	6.400	16.400	5.400	15.400	4.900	14.900
TCMP	—	5.700	9.300	4.700	8.300	4.200	7.800
TZCP	—	6.200	10.500	5.200	9.500	4.700	9.000
S.DATERD	—	5.000	13.100	4.000	12.100	3.500	11.600
S.DATE+	No carry	5.000	9.100	4.000	8.100	3.500	7.600
	Carry	5.100	9.100	4.100	8.100	3.600	7.600
S.DATE-	No borrow	5.100	9.500	4.100	8.500	3.600	8.000
	Borrow	5.000	9.300	4.000	8.300	3.500	7.800
DUTY	—	3.900	8.100	2.900	7.100	2.400	6.600
TIMCHK	—	3.700	6.300	2.700	5.300	2.200	4.800
HOURM	—	4.400	7.700	3.400	6.700	2.900	6.200
DHOURM	—	4.400	7.700	3.400	6.700	2.900	6.200
RFS (X)	(n)=1	7.000	17.000	6.000	16.000	5.500	15.500
	(n)=64	14.500	35.300	13.500	34.300	13.000	33.800
RFS (Y)	(n)=1	6.200	16.400	5.200	15.400	4.700	14.900
	(n)=64	10.500	30.800	9.500	29.800	9.000	29.300

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
COM	When only I/O refresh is selected	8.000	21.200	7.000	20.200	6.500	19.700
	When only CC-Link IE Controller Network refresh is selected (control station side)	17.900	47.200	16.900	46.200	16.400	45.700
	When only CC-Link IE Field Network refresh is selected (master station side)	16.100	41.500	15.100	40.500	14.600	40.000
	When only MELSECNET/H network refresh is selected (control station side)	20.200	51.000	19.200	50.000	18.700	49.500
	When only CC-Link IE Controller Network refresh is selected (normal station side)	18.300	45.800	17.300	44.800	16.800	44.300
	When only CC-Link IE Field Network refresh is selected (local station side)	16.500	40.100	15.500	39.100	15.000	38.600
	When only MELSECNET/H network refresh is selected (normal station side)	20.600	49.600	19.600	48.600	19.100	48.100
	When only CC-Link IE Field Network Basic refresh is selected	9.200	21.900	8.200	20.900	7.7	20.4
	When only an intelligent function module is selected	8.300	13.000	7.300	12.000	6.800	11.500
	Refresh using CPU buffer memory in the multiple CPU system (during END processing)	5.300	13.600	4.300	12.600	3.800	12.100
	Import of input/output outside the group of multiple CPU system Input: 64 points + output: 64 points	5.300	14.200	4.300	13.200	3.800	12.700
	Device/label access service processing (Communication with the engineering tool, GOT, or other external devices)	8.600	25.100	7.600	24.100	7.100	23.600
S.ZCOM	When only CC-Link IE TSN refresh is selected (master station side)	30.800	64.200	29.800	63.200	29.300	62.700
	When only CC-Link IE Controller Network refresh is selected (control station side)	25.400	57.200	24.400	56.200	23.900	55.700
	When only CC-Link IE Field Network refresh is selected (master station side)	23.100	51.500	22.100	50.500	21.600	50.000
	When only MELSECNET/H network refresh is selected (control station side)	28.700	61.400	27.700	60.400	27.200	59.900
	When only CC-Link IE TSN refresh is selected (local station side)	31.200	62.800	30.200	61.800	29.700	61.300
	When only CC-Link IE Controller Network refresh is selected (normal station side)	25.800	55.800	24.800	54.800	24.300	54.300
	When only CC-Link IE Field Network refresh is selected (local station side)	23.500	50.100	22.500	49.100	22.000	48.600
	When only MELSECNET/H network refresh is selected (normal station side)	29.100	60.000	28.100	59.000	27.600	58.500
FROM	Reading buffer memory, (n)=1	8.100	8.600	4.400	5.000	3.700	4.300
	Reading buffer memory, (n)=1000	39.500	57.100	38.500	56.100	38.000	55.600
	Reading host CPU buffer memory, (n)=1	4.200	4.600	1.600	2.000	1.100	1.500
	Reading host CPU buffer memory, (n)=320	15.600	28.500	14.600	27.500	14.100	27.000
	Reading another CPU buffer memory, (n)=1	8.000	8.500	4.400	5.000	3.700	4.300
	Reading another CPU buffer memory, (n)=320	23.400	41.000	22.400	40.000	21.900	39.500

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
DFROM	Reading buffer memory, (n)=1	8.100	8.600	4.500	6.200	4.000	5.500
	Reading buffer memory, (n)=500	40.900	70.900	39.900	69.900	39.400	69.400
	Reading host CPU buffer memory, (n)=1	4.100	4.700	1.500	3.200	1.200	2.800
	Reading host CPU buffer memory, (n)=320	23.400	44.400	22.400	43.400	21.900	42.900
	Reading another CPU buffer memory, (n)=1	8.000	8.700	4.300	6.200	3.800	5.500
	Reading another CPU buffer memory, (n)=320	33.800	67.800	32.800	66.800	32.300	66.300
TO	Writing to buffer memory, (n)=1	6.600	7.100	2.600	3.200	2.100	2.700
	Writing to buffer memory, (n)=1000	46.500	67.000	45.500	66.000	45.000	65.500
	Writing to host CPU buffer memory, (n)=1	4.200	4.300	1.300	1.600	0.900	1.200
	Writing to host CPU buffer memory, (n)=320	12.700	28.100	11.700	27.100	11.200	26.600
DTO	Writing to buffer memory, (n)=1	6.800	7.400	2.800	4.500	2.400	4.000
	Writing to buffer memory, (n)=500	45.900	72.500	44.900	71.500	44.400	71.000
	Writing to host CPU buffer memory, (n)=1	4.200	4.600	1.300	2.900	1.000	2.600
	Writing to host CPU buffer memory, (n)=320	17.000	39.400	16.000	38.400	15.500	37.900
FROMD	Reading buffer memory, (n)=1	8.400	8.900	4.500	6.400	4.000	5.700
	Reading buffer memory, (n)=1000	42.500	65.100	41.500	64.100	41.000	63.600
	Reading host CPU buffer memory, (n)=1	4.400	4.900	1.700	3.300	1.400	2.900
	Reading host CPU buffer memory, (n)=320	17.000	36.200	16.000	35.200	15.500	34.700
	Reading another CPU buffer memory, (n)=1	8.400	9.000	4.500	6.400	4.000	5.700
	Reading another CPU buffer memory, (n)=320	26.000	49.100	25.000	48.100	24.500	47.600
DFROMD	Reading buffer memory, (n)=1	8.300	8.900	4.500	6.400	4.000	5.700
	Reading buffer memory, (n)=500	42.900	71.900	41.900	70.900	41.400	70.400
	Reading host CPU buffer memory, (n)=1	4.100	4.700	1.500	3.200	1.200	2.800
	Reading host CPU buffer memory, (n)=320	23.900	46.800	22.900	45.800	22.400	45.300
	Reading another CPU buffer memory, (n)=1	8.200	8.800	4.400	6.300	3.900	5.600
	Reading another CPU buffer memory, (n)=320	35.500	64.300	34.500	63.300	34.000	62.800
TOD	Writing to buffer memory, (n)=1	6.800	7.400	2.700	4.600	2.300	4.100
	Writing to buffer memory, (n)=1000	48.600	77.500	47.600	76.500	47.100	76.000
	Writing to host CPU buffer memory, (n)=1	4.300	4.800	1.500	3.000	1.200	2.700
	Writing to host CPU buffer memory, (n)=320	15.200	39.100	14.200	38.100	13.700	37.600
DTOD	Writing to buffer memory, (n)=1	6.800	7.400	2.700	4.600	2.300	4.100
	Writing to buffer memory, (n)=500	47.800	82.900	46.800	81.900	46.300	81.400
	Writing to host CPU buffer memory, (n)=1	3.900	4.700	1.300	3.200	1.000	2.800
	Writing to host CPU buffer memory, (n)=320	18.700	48.000	17.700	47.000	17.200	46.500
TYPERD	—	12.000	26.900	11.000	25.900	10.500	25.400

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
UNIINFRD	(n)=1	14.900	21.500	13.900	20.500	13.400	20.000 (When the Process CPU is used: 22.900)
	(n)=16	27.600	34.600	26.600	33.600	26.100	33.100 (When the Process CPU is used: 36.800)
S.RTREAD	—	3.900	7.400	2.900	6.400	2.400	5.900
S.RTWRITE	—	15.000	45.300	14.000	44.300	13.500	43.800
LOGTRG	—	57.200	76.000	56.200	75.000	55.700	74.500
LOGTRGR	—	12.500	20.100	11.500	19.100	11.000	18.600
DATATRG	Number of collection target device points = 32	—	—	—	—	65.000	65.000
SP.SOCOPEN	TCP: Active	17.100	35.300	16.100	34.300	15.600	33.800
	TCP: Unpassive	17.100	35.300	16.100	34.300	15.600	33.800
	TCP: Fullpassive	17.100	35.300	16.100	34.300	15.600	33.800
	UDP	17.100	35.300	16.100	34.300	15.600	33.800
SP.SOCCLOSE	TCP: Executed from own device	16.800	35.500	15.800	34.500	15.300	34.000
	TCP: Executed from external device	16.800	35.500	15.800	34.500	15.300	34.000
	UDP	16.800	35.500	15.800	34.500	15.300	34.000
SP.SOCRCV	TCP: Minimum amount of data (1 byte)	16.500	38.200	15.500	37.200	15.000	36.700
	TCP: Maximum amount of data (10238 bytes)	16.500	38.200	15.500	37.200	15.000	36.700
	UDP: Minimum amount of data (1 byte)	16.700	38.300	15.700	37.300	15.200	36.800
	UDP: Maximum amount of data (10238 bytes)	16.700	38.300	15.700	37.300	15.200	36.800
S.SOCRCVS	TCP: Minimum amount of data (1 byte)	17.300	33.500	16.300	32.500	15.800	32.000
	TCP: Maximum amount of data (10238 bytes)	142.500	181.500	141.500	180.500	141.000	180.000
	UDP: Minimum amount of data (1 byte)	17.300	33.500	16.300	32.500	15.800	32.000
	UDP: Maximum amount of data (10238 bytes)	146.500	183.500	145.500	182.500	145.000	182.000
SP.SOCSND	TCP: Minimum amount of data (1 byte)	19.500	36.900	18.500	35.900	18.000	35.400
	TCP: Maximum amount of data (10238 bytes)	156.500	181.500	155.500	180.500	155.000	180.000
	UDP: Minimum amount of data (1 byte)	19.500	36.900	18.500	35.900	18.000	35.400
	UDP: Maximum amount of data (10238 bytes)	156.500	181.500	155.500	180.500	155.000	180.000
SP.SOCCINF	—	4.600	10.700	3.600	9.700	3.100	9.200
SP.SOCCSET	—	4.800	16.000	3.800	15.000	3.300	14.500
SP.SOCRMODE	Switching from normal mode to fixed-length mode	6.300	15.700	5.300	14.700	4.800	14.200
	Switching from fixed-length mode to normal mode	6.300	15.400	5.300	14.400	4.800	13.900
S.SOCRDATA	Minimum amount of data (1 word)	5.700	11.600	4.700	10.600	4.200	10.100
	Maximum amount of data (5120 words)	115.900	164.300	114.900	163.300	114.400	162.800
SP.ECPRTCL	—	34.700	41.300	33.700	40.300	33.200	39.800
SP.SLMPSND	"Read (command: 0401H)" (reading in units of words): Number of read points = 1 point	17.900	42.900	16.900	41.900	16.400	47.400

A

Instruction name	Condition		Processing time (μs)					
			R00CPU, R01CPU		R02CPU		Other CPU modules	
			Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
SP.FTPPUT	When the FTP server is specified with the module parameter	File name + server path string = 32 characters	89.300	92.200	88.300	91.200	87.800	90.700
		File name + server path string = 64 characters	105.600	108.600	104.600	107.600	104.100	107.100
	When the FTP server is specified with the control data	File name + server path string = 32 characters	91.800	94.900	90.800	93.900	90.300	93.400
		File name + server path string = 64 characters	108.400	111.500	107.400	110.500	106.900	110.000
SP.FTPGET	When the FTP server is specified with the module parameter	File name + server path string = 32 characters	87.900	90.400	86.900	89.400	86.400	88.900
		File name + server path string = 64 characters	104.400	106.900	103.400	105.900	102.900	105.400
	When the FTP server is specified with the control data	File name + server path string = 32 characters	91.200	93.200	90.200	92.200	89.700	91.700
		File name + server path string = 64 characters	107.600	109.700	106.600	108.700	106.100	108.200
PSTOP	File name: "P1"		44.500	79.500	43.500	78.500	43.000	78.000
POFF	File name: "P1"		43.800	78.500	42.800	77.500	42.300	77.000
PSCAN	File name: "P1"		45.200	79.700	44.200	78.700	43.700	78.200
PID	—		5.400	11.700	4.400	10.700	3.900	10.200
S.PIDINIT	1 loop		5.000	8.500	4.000	7.500	3.500	7.000
	32 loop		54.800	58.400	53.800	57.400	53.300	56.900
S.PIDCONT	1 loops (first time)		18.100	19.600	17.100	18.600	16.600	18.100
	1 loops (second and later)		15.400	21.000	14.400	20.000	13.900	19.500
	32 loops (first time)		206.500	210.200	205.500	209.200	205.000	208.700
	32 loops (second and later)		194.800	215.600	193.800	214.600	193.300	214.100
S.PIDSTOP	1 loop		2.600	4.100	1.600	3.100	1.100	2.600
S.PIDRUN	1 loop		3.000	4.200	2.000	3.200	1.500	2.700
S.PIDPRMW	1 loop		5.200	8.000	4.200	7.000	3.700	6.500
PIDINIT	1 loop		4.500	8.200	3.500	7.200	3.000	6.700
	32 loop		38.500	42.000	37.500	41.000	37.000	40.500
PIDCONT	1 loops (first time)		18.100	19.400	17.100	18.400	16.600	17.900
	1 loops (second and later)		15.000	16.600	14.000	15.600	13.500	15.100
	32 loops (first time)		201.100	203.100	200.100	202.100	199.600	201.600
	32 loops (second and later)		192.300	203.200	191.300	202.200	190.800	201.700
PIDSTOP	1 loop		2.700	4.000	1.700	3.000	1.200	2.500
PIDRUN	1 loop		2.700	4.000	1.700	3.000	1.200	2.500
PIDPRMW	1 loop		4.500	8.000	3.500	7.000	3.000	6.500
D.DDRD	Number of read data points = 1		76.800	123.000	75.800	122.000	75.300	121.500
	Number of read data points = 16		77.000	123.000	76.000	122.000	75.500	121.500
	Number of read data points = 96		79.500	125.500	78.500	124.500	78.000	124.000
	Number of read data points = 8192		154.600	198.300	153.600	197.300	153.100	196.800
D.DDWR	Number of read data points = 1		75.400	122.600	74.400	121.600	73.900	121.100
	Number of read data points = 16		75.800	122.700	74.800	121.700	74.300	121.200
	Number of read data points = 96		75.900	122.900	74.900	121.900	74.400	121.400
	Number of read data points = 8192		76.000	122.900	75.000	121.900	74.500	121.400

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
M.DDRD	Number of write data points = 1	66.200	113.900	65.200	112.900	64.700	112.400
	Number of write data points = 16	67.200	114.200	66.200	113.200	65.700	112.700
	Number of write data points = 96	66.900	114.600	65.900	113.600	65.400	113.100
	Number of write data points = 8192	146.700	189.700	145.700	188.700	145.200	188.200
M.DDWR	Number of write data points = 1	65.100	112.900	64.100	111.900	63.600	111.400
	Number of write data points = 16	65.100	113.900	64.100	112.900	63.600	112.400
	Number of write data points = 96	65.400	115.000	64.400	114.000	63.900	113.500
	Number of write data points = 8192	66.200	115.100	65.200	114.100	64.700	113.600
LD [S□]	Continuity/Non-continuity	1.500	1.900	0.700	1.200	0.500	1.100
LD [BL□\S□]	Continuity/Non-continuity	4.300	6.500	3.300	5.500	2.800	5.000
LD [BL□]	Continuity/Non-continuity	3.300	5.100	2.300	4.100	1.800	3.600
AND [S□]	Continuity/Non-continuity	1.500	1.900	0.700	1.200	0.500	1.100
AND [BL□\S□]	Continuity/Non-continuity	4.300	6.500	3.300	5.500	2.800	5.000
AND [BL□]	Continuity/Non-continuity	3.300	5.100	2.300	4.100	1.800	3.600
OR [S□]	Continuity/Non-continuity	1.500	1.900	0.700	1.200	0.500	1.100
OR [BL□\S□]	Continuity/Non-continuity	4.300	6.500	3.300	5.500	2.800	5.000
OR [BL□]	Continuity/Non-continuity	3.300	5.100	2.300	4.100	1.800	3.600
LDI [S□]	Continuity/Non-continuity	1.500	1.900	0.700	1.200	0.500	1.100
LDI [BL□\S□]	Continuity/Non-continuity	4.300	6.500	3.300	5.500	2.800	5.000
LDI [BL□]	Continuity/Non-continuity	3.300	5.100	2.300	4.100	1.800	3.600
ANI [S□]	Continuity/Non-continuity	1.500	1.900	0.700	1.200	0.500	1.100
ANI [BL□\S□]	Continuity/Non-continuity	4.300	6.500	3.300	5.500	2.800	5.000
ANI [BL□]	Continuity/Non-continuity	3.300	5.100	2.300	4.100	1.800	3.600
ORI [S□]	Continuity/Non-continuity	1.500	1.900	0.700	1.200	0.500	1.100
ORI [BL□\S□]	Continuity/Non-continuity	4.300	6.500	3.300	5.500	2.800	5.000
ORI [BL□]	Continuity/Non-continuity	3.300	5.100	2.300	4.100	1.800	3.600
MOV(P) [K4S□]	—	4.100	7.700	3.100	6.700	2.600	6.200
MOV(P) [BL□\K4S□]	—	5.300	10.700	4.300	9.700	3.800	9.200
DMOV(P) [K8S□]	—	4.100	7.700	3.100	6.700	2.600	6.200
DMOV(P) [BL□\K8S□]	—	5.300	10.700	4.300	9.700	3.800	9.200
BMOV(P) [K4S□]	(n)=1	5.700	15.600	4.700	14.600	4.200	14.100
	(n)=96	6.400	16.600	5.400	15.600	4.900	15.100
BMOV(P) [BL□\K4S□]	(n)=1	7.100	19.100	6.100	18.100	5.600	17.600
	(n)=96	7.800	19.700	6.800	18.700	6.300	18.200
SET [S□]	No change time	4.900	9.900	3.900	8.900	3.400	8.400
	Change time	7.100	15.100	5.900	14.100	5.100	13.600
SET [BL□\S□]	No change time	4.900	9.900	3.900	8.900	3.400	8.400
	Change time	6.700	15.100	5.300	14.100	4.700	13.600
SET [BL□]	No change time	3.300	5.000	2.300	4.000	1.800	3.500
	Change time	4.600	5.000	3.200	4.000	2.700	3.500
RST [S□]	No change time	3.800	7.900	2.800	6.900	2.300	6.400
	Change time	6.200	9.000	5.100	8.000	4.400	7.500
RST [BL□\S□]	No change time	3.800	7.900	2.800	6.900	2.300	6.400
	Change time	5.800	9.000	4.700	8.000	4.200	7.500
RST [BL□]	No change time	3.900	5.900	2.900	4.900	2.400	4.400
	Change time	4.800	5.900	3.400	4.900	2.900	4.400
PAUSE [BL□]	—	3.300	4.600	2.300	3.600	1.800	3.100
RSTART [BL□]	—	3.300	4.600	2.300	3.600	1.800	3.100

A

Instruction name	Condition	Processing time (μs)					
		R00CPU, R01CPU		R02CPU		Other CPU modules	
		Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
BRSET [S□]	—	3.600	5.000	2.600	4.000	2.100	3.500
SP.CONTSW	SM1646=ON	—	—	—	—	10.600	16.000
	SM1646=OFF	—	—	—	—	137.200	169.700
DCONTSW	Enabled → Disabled	—	—	—	—	1.400	1.900
ECONTSW	Disabled → Enabled	—	—	—	—	1.400	2.000
CONTWR	Number of write data points = 1	—	—	—	—	18.300	21.200
	Number of write data points = 16	—	—	—	—	18.700	21.400
	Number of write data points = 96	—	—	—	—	19.200	21.900
	Number of write data points = 1024	—	—	—	—	27.200	29.900
SP.SIDRD	—	—	—	—	—	30.800	37.400

Time added to instruction processing time

When using the file register (R/ZR), the module access device (U□\G□) and module label (ones with the label name including _D), or the link direct device (J□\□) and module label (link devices with the label name including _D), add the additional time described in the section below to each instruction processing time.

Device name		Data type	Device part specification	Additional time (μs)	
				R04CPU, R04ENCPU	R08CPU, R08ENCPU, R16CPU, R16ENCPU, R32CPU, R32ENCPU, R120CPU, R120ENCPU
File register (R/ZR)	When an extended SRAM cassette and a battery-less option cassette is not inserted	Bit	Source	0.074	0.043
			Destination	0.023	0.023
		Word	Source	0.074	0.043
			Destination	0.023	0.023
		Double word	Source	0.148	0.085
			Destination	0.044	0.044
	When an extended SRAM cassette or a battery-less option cassette is inserted	Bit	Source	0.099	0.099
			Destination	0.028	0.028
		Word	Source	0.099	0.099
			Destination	0.028	0.028
		Double word	Source	0.198	0.198
			Destination	0.054	0.054
Module access device (U□\G□) and module label (ones with the label name including _D)		Bit	Source	13.000	13.000
			Destination	14.000	14.000
		Word	Source	13.000	13.000
			Destination	14.000	14.000
		Double word	Source	13.000	13.000
			Destination	14.000	14.000
Link direct device (J□\□) and module label (link devices with the label name including _D)		Bit	Source	51.000	51.000
			Destination	52.000	52.000
		Word	Source	51.000	51.000
			Destination	52.000	52.000
		Double word	Source	51.000	51.000
			Destination	52.000	52.000

A

Appendix 2 Number of Basic Steps and Availability of Subset Processing

The number of basic steps and the availability of subset processing are shown below.

Instruction name	Number of basic steps	Subset availability
LD	1	○
LDI	1	○
AND	1	○
ANI	1	○
OR	1	○
ORI	1	○
LDP	2	○
LDF	2	○
ANDP	2	○
ANDF	2	○
ORP	2	○
ORF	2	○
LDPI	2	○
LDFI	2	○
ANDPI	2	○
ANDFI	2	○
ORPI	2	○
ORFI	2	○
ANB	1	—
ORB	1	—
MPS	1	—
MRD	1	—
MPP	1	—
INV	1	—
MEP	1	—
MEF	1	—
EGP	1	—
EGF	1	—
OUT	1	○
OUT T/ST	4	—
OUT LT/LST	2	—
OUT C	4	—
OUT LC	4	—
OUT F	2	—
OUTH T/ST	4	—
SET	1	○
RST	1	○
SET F	3	—
RST F	3	—
PLS	2	○
PLF	2	○
FF	2	○
DELTA	2	—
DELTAP	3	—
SFT	2	—
SFTP	3	—
MC	2	○
MCR	1	—

Instruction name	Number of basic steps	Subset availability
PHASE	19[*1]	—
PHASECHG	6	—
PHASEEND	4	—
FEND	2	—
END	2	—
STOP	1	—
NOP	1	—
LD=	3	○
LD<>	3	○
LD>	3	○
LD<=	3	○
LD<	3	○
LD>=	3	○
AND=	3	○
AND<>	3	○
AND>	3	○
AND<=	3	○
AND<	3	○
AND>=	3	○
OR=	3	○
OR<>	3	○
OR>	3	○
OR<=	3	○
OR<	3	○
OR>=	3	○
LD=_U	3	○
LD<>_U	3	○
LD>_U	3	○
LD<=_U	3	○
LD<_U	3	○
LD>=_U	3	○
AND=_U	3	○
AND<>_U	3	○
AND>_U	3	○
AND<=_U	3	○
AND<_U	3	○
AND>=_U	3	○
OR=_U	3	○
OR<>_U	3	○
OR>_U	3	○
OR<=_U	3	○
OR<_U	3	○
OR>=_U	3	○
LDD=	3	○
LDD<>	3	○
LDD>	3	○
LDD<=	3	○
LDD<	3	○
LDD>=	3	○
ANDD=	3	○
ANDD<>	3	○
ANDD>	3	○
ANDD<=	3	○

Instruction name	Number of basic steps	Subset availability
ANDD<	3	○
ANDD>=	3	○
ORD=	3	○
ORD<>	3	○
ORD>	3	○
ORD<=	3	○
ORD<	3	○
ORD>=	3	○
LDD=_U	3	○
LDD<>_U	3	○
LDD>_U	3	○
LDD<=_U	3	○
LDD<_U	3	○
LDD>=_U	3	○
ANDD=_U	3	○
ANDD<>_U	3	○
ANDD>_U	3	○
ANDD<=_U	3	○
ANDD<_U	3	○
ANDD>=_U	3	○
ORD=_U	3	○
ORD<>_U	3	○
ORD>_U	3	○
ORD<=_U	3	○
ORD<_U	3	○
ORD>=_U	3	○
CMP	4	—
CMPP	5	—
CMP_U	4	—
CMPP_U	5	—
DCMP	4	—
DCMPP	5	—
DCMP_U	4	—
DCMPP_U	5	—
ZCP	5	—
ZCPP	6	—
ZCP_U	5	—
ZCPP_U	6	—
DZCP	5	—
DZCPP	6	—
DZCP_U	5	—
DZCPP_U	6	—
BKCMP=	5	—
BKCMP<>	5	—
BKCMP>	5	—
BKCMP<=	5	—
BKCMP<	5	—
BKCMP>=	5	—
BKCMP=P	6	—
BKCMP<>P	6	—
BKCMP>P	6	—
BKCMP<=P	6	—
BKCMP<P	6	—

Instruction name	Number of basic steps	Subset availability
BKCMP>=P	6	—
BKCMP=_U	5	—
BKCMP<>_U	5	—
BKCMP>_U	5	—
BKCMP<=_U	5	—
BKCMP<_U	5	—
BKCMP>=_U	5	—
BKCMP=P_U	6	—
BKCMP<>P_U	6	—
BKCMP>P_U	6	—
BKCMP<=P_U	6	—
BKCMP<P_U	6	—
BKCMP>=P_U	6	—
DBKCMP=	5	—
DBKCMP<>	5	—
DBKCMP>	5	—
DBKCMP<=	5	—
DBKCMP<	5	—
DBKCMP>=	5	—
DBKCMP=P	6	—
DBKCMP<>P	6	—
DBKCMP>P	6	—
DBKCMP<=P	6	—
DBKCMP<P	6	—
DBKCMP>=P	6	—
DBKCMP=_U	5	—
DBKCMP<>_U	5	—
DBKCMP>_U	5	—
DBKCMP<=_U	5	—
DBKCMP<_U	5	—
DBKCMP>=_U	5	—
DBKCMP=P_U	6	—
DBKCMP<>P_U	6	—
DBKCMP>P_U	6	—
DBKCMP<=P_U	6	—
DBKCMP<P_U	6	—
DBKCMP>=P_U	6	—
+ (s) (d)	3	○
+P (s) (d)	4	○
+ (s1) (s2) (d)	3	○
+P (s1) (s2) (d)	4	○
+_U (s) (d)	3	○
+P_U (s) (d)	4	○
+_U (s1) (s2) (d)	3	○
+P_U (s1) (s2) (d)	4	○
- (s) (d)	3	○
-P (s) (d)	4	○
- (s1) (s2) (d)	3	○
-P (s1) (s2) (d)	4	○
-_U (s) (d)	3	○
-P_U (s) (d)	4	○
-_U (s1) (s2) (d)	3	○
-P_U (s1) (s2) (d)	4	○

Instruction name	Number of basic steps	Subset availability
D+ (s) (d)	3	○
D+P (s) (d)	4	○
D+ (s1) (s2) (d)	3	○
D+P (s1) (s2) (d)	4	○
D+_U (s) (d)	3	○
D+P_U (s) (d)	4	○
D+_U (s1) (s2) (d)	3	○
D+P_U (s1) (s2) (d)	4	○
D- (s) (d)	3	○
D-P (s) (d)	4	○
D- (s1) (s2) (d)	3	○
D-P (s1) (s2) (d)	4	○
D-_U (s) (d)	3	○
D-P_U (s) (d)	4	○
D-_U (s1) (s2) (d)	3	○
D-P_U (s1) (s2) (d)	4	○
*	3	○
*P	4	○
*_U	3	○
*P_U	4	○
/	3	○
/P	4	○
/_U	3	○
/P_U	4	○
D*	3	○
D*P	4	○
D*_U	3	○
D*P_U	4	○
D/	3	○
D/P	4	○
D/_U	3	○
D/P_U	4	○
B+ (s) (d)	3	○
B+P (s) (d)	4	○
B+ (s1) (s2) (d)	4	—
B+P (s1) (s2) (d)	5	—
B- (s) (d)	3	○
B-P (s) (d)	4	○
B- (s1) (s2) (d)	4	—
B-P (s1) (s2) (d)	5	—
DB+ (s) (d)	3	—
DB+P (s) (d)	4	—
DB+ (s1) (s2) (d)	4	—
DB+P (s1) (s2) (d)	5	—
DB- (s) (d)	3	—
DB-P (s) (d)	4	—
DB- (s1) (s2) (d)	4	—
DB-P (s1) (s2) (d)	5	—
B*	4	○
B*P	5	○
B/	4	○
B/P	5	○
DB*	4	—

Instruction name	Number of basic steps	Subset availability
DB*P	5	—
DB/	4	—
DB/P	5	—
BK+	5	—
BK+P	6	—
BK+_U	5	—
BK+P_U	6	—
BK-	5	—
BK-P	6	—
BK-_U	5	—
BK-P_U	6	—
DBK+	5	—
DBK+P	6	—
DBK+_U	5	—
DBK+P_U	6	—
DBK-	5	—
DBK-P	6	—
DBK-_U	5	—
DBK-P_U	6	—
INC	2	○
INCP	3	○
INC_U	2	○
INCP_U	3	○
DEC	2	○
DECP	3	○
DEC_U	2	○
DECP_U	3	○
DINC	2	○
DINCP	3	○
DINC_U	2	○
DINCP_U	3	○
DDEC	2	○
DDECP	3	○
DDEC_U	2	○
DDECP_U	3	○
WAND (s) (d)	3	○
WANDP (s) (d)	4	○
WAND (s1) (s2) (d)	3	○
WANDP (s1) (s2) (d)	4	○
DAND (s) (d)	3	○
DANDP (s) (d)	4	○
DAND (s1) (s2) (d)	3	○
DANDP (s1) (s2) (d)	4	○
BKAND	5	—
BKANDP	6	—
WOR (s) (d)	3	○
WORP (s) (d)	4	○
WOR (s1) (s2) (d)	3	○
WORP (s1) (s2) (d)	4	○
DOR (s) (d)	3	○
DORP (s) (d)	4	○
DOR (s1) (s2) (d)	3	○
DORP (s1) (s2) (d)	4	○

Instruction name	Number of basic steps	Subset availability
BKOR	5	—
BKORP	6	—
WXOR (s) (d)	3	○
WXORP (s) (d)	4	○
WXOR (s1) (s2) (d)	3	○
WXORP (s1) (s2) (d)	4	○
DXOR (s) (d)	3	○
DXORP (s) (d)	4	○
DXOR (s1) (s2) (d)	3	○
DXORP (s1) (s2) (d)	4	○
BKXOR	5	—
BKXORP	6	—
WXNR (s) (d)	3	○
WXNRP (s) (d)	4	○
WXNR (s1) (s2) (d)	3	○
WXNRP (s1) (s2) (d)	4	○
DXNR (s) (d)	3	○
DXNRP (s) (d)	4	○
DXNR (s1) (s2) (d)	3	○
DXNRP (s1) (s2) (d)	4	○
BKXNR	5	—
BKXNRP	6	—
BSET	3	○
BSETP	4	○
BRST	3	○
BRSTP	4	○
TEST	4	○
TESTP	5	○
DTEST	5	○
DTESTP	6	○
BKRST	3	○
BKRSTP	4	○
SFR	4	○
SFRP	5	○
SFL	4	○
SFLP	5	○
BSFR	3	○
BSFRP	4	○
BSFL	3	○
BSFLP	4	○
DSFR	3	○
DSFRP	4	○
DSFL	3	○
DSFLP	4	○
DDSFR	3	—
DDSFRP	4	—
DDSFL	3	—
DDSFLP	4	—
ESFR	3	—
ESFRP	4	—
ESFL	3	—
ESFLP	4	—
EDSFR	3	—

Instruction name	Number of basic steps	Subset availability
EDSFRP	4	—
EDSFL	3	—
EDSFLP	4	—
SFTBR	4	○
SFTBRP	5	○
SFTBL	4	○
SFTBLP	5	○
SFTWR	4	○
SFTWRP	5	○
SFTWL	4	○
SFTWLP	5	○
SFTDWR	4	—
SFTDWRP	5	—
DWSFTR	5	—
DWSFTRP	6	—
SFTDWL	4	—
SFTDWLP	5	—
DWSFTL	5	—
DWSFTLP	6	—
SFTER	4	—
SFTERP	5	—
ESFTR	5	—
ESFTRP	6	—
SFTEL	4	—
SFTELP	5	—
ESFTL	5	—
ESFTLP	6	—
SFTEDR	4	—
SFTEDRP	5	—
EDSFTR	5	—
EDSFTRP	6	—
SFTEDL	4	—
SFTEDLP	5	—
EDSFTL	5	—
EDSFTLP	6	—
BCD	2	○
BCDP	3	○
DBCD	2	○
DBCDP	3	○
BIN	2	○
BINP	3	○
DBIN	2	○
DBINP	3	○
FLT2INT	2	○
FLT2INTP	3	○
FLT2UINT	2	○
FLT2UINTP	3	○
FLT2DINT	2	○
FLT2DINTP	3	○
FLT2UDINT	2	○
FLT2UDINTP	3	○
DBL2INT	3	—
DBL2INTP	4	—

Instruction name	Number of basic steps	Subset availability
DBL2UINT	3	—
DBL2UINTP	4	—
DBL2DINT	3	—
DBL2DINTP	4	—
DBL2UDINT	3	—
DBL2UDINTP	4	—
INT2UINT	3	○
INT2UINTP	4	○
INT2DINT	3	○
INT2DINTP	4	○
INT2UDINT	3	○
INT2UDINTP	4	○
UINT2INT	3	○
UINT2INTP	4	○
UINT2DINT	3	○
UINT2DINTP	4	○
UINT2UDINT	3	○
UINT2UDINTP	4	○
DINT2INT	2	○
DINT2INTP	3	○
DINT2UINT	3	○
DINT2UINTP	4	○
DINT2UDINT	3	○
DINT2UDINTP	4	○
UDINT2INT	3	○
UDINT2INTP	4	○
UDINT2UINT	3	○
UDINT2UINTP	4	○
UDINT2DINT	3	○
UDINT2DINTP	4	○
GRY	2	○
GRYP	3	○
GRY_U	2	○
GRYP_U	3	○
DGRY	2	○
DGRYP	3	○
DGRY_U	2	○
DGRYP_U	3	○
GBIN	2	○
GBINP	3	○
GBIN_U	2	○
GBINP_U	3	○
DGBIN	2	○
DGBINP	3	○
DGBIN_U	2	○
DGBINP_U	3	○
BKBCD	4	—
BKBCDP	5	—
BKBIN	4	—
BKBINP	5	—
DABIN	3	—
DABINP	4	—
DABIN_U	3	—

Instruction name	Number of basic steps	Subset availability
DABINP_U	4	—
DDABIN	3	—
DDABINP	4	—
DDABIN_U	3	—
DDABINP_U	4	—
HABIN	3	—
HABINP	4	—
DHABIN	3	—
DHABINP	4	—
DABCD	3	—
DABCDP	4	—
DDABCD	3	—
DDABCDP	4	—
VAL	4	—
VALP	5	—
VAL_U	4	—
VALP_U	5	—
DVAL	4	—
DVALP	5	—
DVAL_U	4	—
DVALP_U	5	—
ASC2INT	4	—
ASC2INTP	5	—
EMOD	4	—
EMODP	5	—
NEG	2	○
NEGP	3	○
DNEG	2	○
DNEGP	3	○
DECO	4	—
DECOP	5	—
ENCO	4	—
ENCOP	5	—
SEG	3	○
SEGP	4	○
DIS	4	—
DISP	5	—
UNI	4	—
UNIP	5	—
NDIS	4	—
NDISP	5	—
NUNI	4	—
NUNIP	5	—
WTOB	4	—
WTOBP	5	—
BTOW	4	—
BTOWP	5	—
MOV	2	○
MOVP	3	○
DMOV	2	○
DMOVP	3	○
CML	2	○
CMLP	3	○

Instruction name	Number of basic steps	Subset availability
DCML	2	○
DCMLP	3	○
SMOV	6	—
SMOVP	7	—
CMLB	3	○
CMLBP	4	○
BMOV	4	○
BMOVP	5	○
BMOVL	4	○
BMOVLP	5	○
FMOV	4	○
FMOVP	5	○
FMOVL	4	○
FMOVLP	5	○
DFMOV	4	○
DFMOVP	5	○
DFMOVL	4	○
DFMOVLP	5	○
XCH	3	○
XCHP	4	○
DXCH	3	○
DXCHP	4	○
BXCH	4	—
BXCHP	5	—
SWAP	2	—
SWAPP	3	—
DSWAP	3	—
DSWAPP	4	—
MOVB	2	○
MOVBP	3	○
BLKMOVB	4	—
BLKMOVBP	5	—
ROR	4	○
RORP	5	○
RCR	4	○
RCRP	5	○
DROR	4	○
DRORP	5	○
DRCR	4	○
DRCRP	5	○
ROL	4	○
ROLP	5	○
RCL	4	○
RCLP	5	○
DROL	4	○
DROLP	5	○
DRCL	4	○
DRCLP	5	○
CJ	4	—
SCJ	5	—
JMP	4	—
GOEND	1	—
DI	1	—

Instruction name	Number of basic steps	Subset availability
DI (s)	2	○
EI	1	○
IMASK	2	○
SIMASK	3	○
IRET	1	—
WDT	1	—
WDTP	2	—
FOR	2	○
NEXT	1	—
BREAK	4	—
BREAKP	5	—
CALL (P)	3	—
CALL (P) (s1)	4	—
CALL (P) (s1) (s2)	5	—
CALL (P) (s1) (s2) (s3)	6	—
CALL (P) (s1) (s2) (s3) (s4)	7	—
CALL (P) (s1) (s2) (s3) (s4) (s5)	8	—
CALLP (P)	4	—
CALLP (P) (s1)	5	—
CALLP (P) (s1) (s2)	6	—
CALLP (P) (s1) (s2) (s3)	7	—
CALLP (P) (s1) (s2) (s3) (s4)	8	—
CALLP (P) (s1) (s2) (s3) (s4) (s5)	9	—
RET	1	—
FCALL (P)	3	—
FCALL (P) (s1)	4	—
FCALL (P) (s1) (s2)	5	—
FCALL (P) (s1) (s2) (s3)	6	—
FCALL (P) (s1) (s2) (s3) (s4)	7	—
FCALL (P) (s1) (s2) (s3) (s4) (s5)	8	—
FCALLP (P)	4	—
FCALLP (P) (s1)	5	—
FCALLP (P) (s1) (s2)	6	—
FCALLP (P) (s1) (s2) (s3)	7	—
FCALLP (P) (s1) (s2) (s3) (s4)	8	—
FCALLP (P) (s1) (s2) (s3) (s4) (s5)	9	—
ECALL (file name) (P)	4 + Number of characters in the file name	—
ECALL (file name) (P) (s1)	5 + Number of characters in the file name	—
ECALL (file name) (P) (s1) (s2)	6 + Number of characters in the file name	—
ECALL (file name) (P) (s1) (s2) (s3)	7 + Number of characters in the file name	—
ECALL (file name) (P) (s1) (s2) (s3) (s4)	8 + Number of characters in the file name	—
ECALL (file name) (P) (s1) (s2) (s3) (s4) (s5)	9 + Number of characters in the file name	—
ECALLP (file name) (P)	5 + Number of characters in the file name	—
ECALLP (file name) (P) (s1)	6 + Number of characters in the file name	—
ECALLP (file name) (P) (s1) (s2)	7 + Number of characters in the file name	—
ECALLP (file name) (P) (s1) (s2) (s3)	8 + Number of characters in the file name	—
ECALLP (file name) (P) (s1) (s2) (s3) (s4)	9 + Number of characters in the file name	—
ECALLP (file name) (P) (s1) (s2) (s3) (s4) (s5)	10 + Number of characters in the file name	—
EFCALL (file name) (P)	4 + Number of characters in the file name	—
EFCALL (file name) (P) (s1)	5 + Number of characters in the file name	—
EFCALL (file name) (P) (s1) (s2)	6 + Number of characters in the file name	—
EFCALL (file name) (P) (s1) (s2) (s3)	7 + Number of characters in the file name	—
EFCALL (file name) (P) (s1) (s2) (s3) (s4)	8 + Number of characters in the file name	—

Instruction name	Number of basic steps	Subset availability
EFCALL (file name) (P) (s1) (s2) (s3) (s4) (s5)	9 + Number of characters in the file name	—
EFCALLP (file name) (P)	5 + Number of characters in the file name	—
EFCALLP (file name) (P) (s1)	6 + Number of characters in the file name	—
EFCALLP (file name) (P) (s1) (s2)	7 + Number of characters in the file name	—
EFCALLP (file name) (P) (s1) (s2) (s3)	8 + Number of characters in the file name	—
EFCALLP (file name) (P) (s1) (s2) (s3) (s4)	9 + Number of characters in the file name	—
EFCALLP (file name) (P) (s1) (s2) (s3) (s4) (s5)	10 + Number of characters in the file name	—
XCALL (P)	4	—
XCALL (P) (s1)	5	—
XCALL (P) (s1) (s2)	6	—
XCALL (P) (s1) (s2) (s3)	7	—
XCALL (P) (s1) (s2) (s3) (s4)	8	—
XCALL (P) (s1) (s2) (s3) (s4) (s5)	9	—
FIFR	3	—
FIFRP	4	—
FPOP	3	—
FPOPP	4	—
FIFW	3	—
FIFWP	4	—
FINS	4	—
FINSP	5	—
FDEL	4	—
FDELP	5	—
S.DEVLD	4	—
SP.DEVLD	5	—
SP.DEVST	6	—
SP.FREAD	8	—
SP.FWRITE	8	—
SP.FDELETE	7	—
SP.FCOPY	9	—
SP.FMOVE	9	—
SP.FRENAME	8	—
SP.FSTATUS	8	—
LEDR	2	—
PALERT	2	—
PALERTP	3	—
PABORT	2	—
LD$=	3	○
LD$<>	3	○
LD$>	3	○
LD$<=	3	○
LD$<	3	○
LD$>=	3	○
AND$=	3	○
AND$<>	3	○
AND$>	3	○
AND$<=	3	○
AND$<	3	○
AND$>=	3	○
OR$=	3	○
OR$<>	3	○
OR$>	3	○
OR$<=	3	○

Instruction name	Number of basic steps	Subset availability
OR$<	3	○
OR$>=	3	○
$+ (s) (d)	3	○
$+P (s) (d)	4	○
$+ (s1) (s2) (d)	4	○
$+P (s1) (s2) (d)	5	○
$MOV	3	—
$MOVP	4	—
$MOV_WS	3	—
$MOVP_WS	4	—
BINDA	3	—
BINDAP	4	—
BINDA_U	3	—
BINDAP_U	4	—
DBINDA	3	—
DBINDAP	4	—
DBINDA_U	3	—
DBINDAP_U	4	—
BINHA	3	—
BINHAP	4	—
DBINHA	3	—
DBINHAP	4	—
STR	4	—
STRP	5	—
STR_U	4	—
STRP_U	5	—
DSTR	4	—
DSTRP	5	—
DSTR_U	4	—
DSTRP_U	5	—
BCDDA	3	—
BCDDAP	4	—
DBCDDA	3	—
DBCDDAP	4	—
ESTR	4	—
ESTRP	5	—
INT2ASC	4	—
INT2ASCP	5	—
WS2SJIS	3	—
WS2SJISP	4	—
SJIS2WS	3	—
SJIS2WSP	4	—
SJIS2WSB	3	—
SJIS2WSBP	4	—
LEN	3	○
LENP	4	○
RIGHT	4	—
RIGHTP	5	—
LEFT	4	—
LEFTP	5	—
MIDR	4	—
MIDRP	5	—
MIDW	4	—

Instruction name	Number of basic steps	Subset availability
MIDWP	5	—
INSTR	5	—
INSTRP	6	—
STRINS	4	—
STRINSP	5	—
STRDEL	4	—
STRDELP	5	—
LDE=	3	○
LDE<>	3	○
LDE>	3	○
LDE<=	3	○
LDE<	3	○
LDE>=	3	○
ANDE=	3	○
ANDE<>	3	○
ANDE>	3	○
ANDE<=	3	○
ANDE<	3	○
ANDE>=	3	○
ORE=	3	○
ORE<>	3	○
ORE>	3	○
ORE<=	3	○
ORE<	3	○
ORE>=	3	○
LDED=	3	—
LDED<>	3	—
LDED>	3	—
LDED<=	3	—
LDED<	3	—
LDED>=	3	—
ANDED=	3	—
ANDED<>	3	—
ANDED>	3	—
ANDED<=	3	—
ANDED<	3	—
ANDED>=	3	—
ORED=	3	—
ORED<>	3	—
ORED>	3	—
ORED<=	3	—
ORED<	3	—
ORED>=	3	—
ECMP	4	—
ECMPP	5	—
EDCMP	4	—
EDCMPP	5	—
EZCP	5	—
EZCPP	6	—
EDZCP	5	—
EDZCPP	6	—
E+ (s) (d)	3	○
E+P (s) (d)	4	○

Instruction name	Number of basic steps	Subset availability
E+ (s1) (s2) (d)	3	○
E+P (s1) (s2) (d)	4	○
E- (s) (d)	3	○
E-P (s) (d)	4	○
E- (s1) (s2) (d)	3	○
E-P (s1) (s2) (d)	4	○
ED+ (s) (d)	3	—
ED+P (s) (d)	4	—
ED+ (s1) (s2) (d)	4	—
ED+P (s1) (s2) (d)	5	—
ED- (s) (d)	3	—
ED-P (s) (d)	4	—
ED- (s1) (s2) (d)	4	—
ED-P (s1) (s2) (d)	5	—
E*	3	○
E*P	4	○
E/	3	○
E/P	4	○
ED*	4	—
ED*P	5	—
ED/	4	—
ED/P	5	—
INT2FLT	2	○
INT2FLTP	3	○
UINT2FLT	2	○
UINT2FLTP	3	○
DINT2FLT	2	○
DINT2FLTP	3	○
UDINT2FLT	2	○
UDINT2FLTP	3	○
DBL2FLT	3	—
DBL2FLTP	4	—
INT2DBL	3	—
INT2DBLP	4	—
UINT2DBL	3	—
UINT2DBLP	4	—
DINT2DBL	3	—
DINT2DBLP	4	—
UDINT2DBL	3	—
UDINT2DBLP	4	—
FLT2DBL	3	—
FLT2DBLP	4	—
EVAL	3	—
EVALP	4	—
EREXP	4	—
EREXPP	5	—
ENEG	2	—
ENEGP	3	—
EDNEG	2	—
EDNEGP	3	—
EMOV	2	○
EMOVP	3	○
EDMOV	2	○

Instruction name	Number of basic steps	Subset availability
EDMOVP	3	○
SIN	3	—
SINP	4	—
COS	3	—
COSP	4	—
TAN	3	—
TANP	4	—
ASIN	3	—
ASINP	4	—
ACOS	3	—
ACOSP	4	—
ATAN	3	—
ATANP	4	—
SIND	3	—
SINDP	4	—
COSD	3	—
COSDP	4	—
TAND	3	—
TANDP	4	—
ASIND	3	—
ASINDP	4	—
ACOSD	3	—
ACOSDP	4	—
ATAND	3	—
ATANDP	4	—
BSIN	3	—
BSINP	4	—
BCOS	3	—
BCOSP	4	—
BTAN	3	—
BTANP	4	—
BASIN	3	—
BASINP	4	—
BACOS	3	—
BACOSP	4	—
BATAN	3	—
BATANP	4	—
RAD	3	—
RADP	4	—
DEG	3	—
DEGP	4	—
RADD	3	—
RADDP	4	—
DEGD	3	—
DEGDP	4	—
ESQRT	3	—
ESQRTP	4	—
EDSQRT	3	—
EDSQRTP	4	—
EXP	3	—
EXPP	4	—
EXPD	3	—
EXPDP	4	—

Instruction name	Number of basic steps	Subset availability
LOG	3	—
LOGP	4	—
LOGD	3	—
LOGDP	4	—
BSQRT	3	—
BSQRTP	4	—
BDSQRT	3	—
BDSQRTP	4	—
POW	4	—
POWP	5	—
POWD	4	—
POWDP	5	—
LOG10	3	—
LOG10P	4	—
LOG10D	3	—
LOG10DP	4	—
EMAX	4	—
EMAXP	5	—
EDMAX	4	—
EDMAXP	5	—
EMIN	4	—
EMINP	5	—
EDMIN	4	—
EDMINP	5	—
RND	2	—
RNDP	3	—
SRND	2	—
SRNDP	3	—
ZPUSH (d)	2	○
ZPUSHP (d)	3	○
ZPUSH (s) (d)	3	○
ZPUSHP (s) (d)	4	○
ZPOP (d)	2	○
ZPOPP (d)	3	○
ZPOP (s) (d)	3	○
ZPOPP (s) (d)	4	○
LIMIT	5	○
LIMITP	6	○
LIMIT_U	5	○
LIMITP_U	6	○
DLIMIT	5	○
DLIMITP	6	○
DLIMIT_U	5	○
DLIMITP_U	6	○
BAND	5	—
BANDP	6	—
BAND_U	5	—
BANDP_U	6	—
DBAND	5	—
DBANDP	6	—
DBAND_U	5	—
DBANDP_U	6	—
ZONE	5	—

Instruction name	Number of basic steps	Subset availability
ZONEP	6	—
ZONE_U	5	—
ZONEP_U	6	—
DZONE	5	—
DZONEP	6	—
DZONE_U	5	—
DZONEP_U	6	—
SCL	4	—
SCLP	5	—
SCL_U	4	—
SCLP_U	5	—
DSCL	4	—
DSCLP	5	—
DSCL_U	4	—
DSCLP_U	5	—
SCL2	4	—
SCL2P	5	—
SCL2_U	4	—
SCL2P_U	5	—
DSCL2	4	—
DSCL2P	5	—
DSCL2_U	4	—
DSCL2P_U	5	—
UDCNT1	5	—
UDCNT2	5	—
TTMR	4	—
STMR	4	—
ROTC	5	—
RAMPQ	7	—
SPD	5	—
PLSY	5	—
PWM	5	—
MTR	5	—
CCD	4	—
CCDP	5	—
SERDATA	5	—
SERDATAP	6	—
DSERDATA	5	—
DSERDATAP	6	—
SERMM	6	—
SERMMP	7	—
DSERMM	6	—
DSERMMP	7	—
SUM	3	○
SUMP	4	○
DSUM	3	○
DSUMP	4	○
BON	5	—
BONP	6	—
DBON	5	—
DBONP	6	—
MAX	4	—
MAXP	5	—

Instruction name	Number of basic steps	Subset availability
MAX_U	4	—
MAXP_U	5	—
DMAX	4	—
DMAXP	5	—
DMAX_U	4	—
DMAXP_U	5	—
MIN	4	—
MINP	5	—
MIN_U	4	—
MINP_U	5	—
DMIN	4	—
DMINP	5	—
DMIN_U	4	—
DMINP_U	5	—
SORTD	7	—
SORTD_U	7	—
DSORTD	7	—
DSORTD_U	7	—
SORTTBL	7	—
SORTTBL_U	7	—
SORTTBL2	7	—
SORTTBL2_U	7	—
DSORTTBL2	7	—
DSORTTBL2_U	7	—
WSUM	4	○
WSUMP	5	○
WSUM_U	4	○
WSUMP_U	5	○
DWSUM	4	—
DWSUMP	5	—
DWSUM_U	4	—
DWSUMP_U	5	—
MEAN	4	—
MEANP	5	—
MEAN_U	4	—
MEANP_U	5	—
DMEAN	4	—
DMEANP	5	—
DMEAN_U	4	—
DMEANP_U	5	—
SQRT	4	—
SQRTP	5	—
DSQRT	4	—
DSQRTP	5	—
CRC	4	—
CRCP	5	—
DBOPEN	5	—
DBOPENP	6	—
DBCLOSE	4	—
DBCLOSEP	5	—
DBINSERT	7	—
DBINSERTP	8	—
DBUPDATE	8	—

Instruction name	Number of basic steps	Subset availability
DBUPDATEP	9	—
DBSELECT	8	—
DBSELECTP	9	—
DBDELETE	6	—
DBDELETEP	7	—
DBIMPORT	4	—
DBIMPORTP	5	—
DBEXPORT	4	—
DBEXPORTP	5	—
DBTRANS	4	—
DBTRANSP	5	—
DBCOMMIT	4	—
DBCOMMITP	5	—
DBROLBAK	4	—
DBROLBAKP	5	—
RSET	2	—
RSETP	3	—
QDRSET	2 + Number of characters in the file name	—
QDRSETP	3 + Number of characters in the file name	—
ZRRDB	3	—
ZRRDBP	4	—
ZRWRB	3	—
ZRWRBP	4	—
ADRSET	3	—
ADRSETP	4	—
DATERD	2	—
DATERDP	3	—
DATEWR	2	—
DATEWRP	3	—
DATE+	4	—
DATE+P	5	—
DATE-	4	—
DATE-P	5	—
TIME2SEC	3	—
TIME2SECP	4	—
SEC2TIME	3	—
SEC2TIMEP	4	—
DATE2SEC	3	—
DATE2SECP	4	—
DATE2SEC_U	3	—
DATE2SECP_U	4	—
SEC2DATE	3	—
SEC2DATEP	4	—
SEC2DATE_U	3	—
SEC2DATEP_U	4	—
LDDT=	4	—
LDDT<>	4	—
LDDT>	4	—
LDDT<=	4	—
LDDT<	4	—
LDDT>=	4	—
ANDDT=	4	—
ANDDT<>	4	—

Instruction name	Number of basic steps	Subset availability
ANDDT>	4	—
ANDDT<=	4	—
ANDDT<	4	—
ANDDT>=	4	—
ORDT=	4	—
ORDT<>	4	—
ORDT>	4	—
ORDT<=	4	—
ORDT<	4	—
ORDT>=	4	—
LDTM=	4	—
LDTM<>	4	—
LDTM>	4	—
LDTM<=	4	—
LDTM<	4	—
LDTM>=	4	—
ANDTM=	4	—
ANDTM<>	4	—
ANDTM>	4	—
ANDTM<=	4	—
ANDTM<	4	—
ANDTM>=	4	—
ORTM=	4	—
ORTM<>	4	—
ORTM>	4	—
ORTM<=	4	—
ORTM<	4	—
ORTM>=	4	—
TCMP	6	—
TCMPP	7	—
TZCP	5	—
TZCPP	6	—
S.DATERD	2	—
SP.DATERD	3	—
S.DATE+	4	—
SP.DATE+	5	—
S.DATE-	4	—
SP.DATE-	5	—
DUTY	5	—
TIMCHK	5	—
HOURM	5	—
DHOURM	5	—
RFS	3	—
RFSP	4	—
COM	1	—
COMP	2	—
S.ZCOM	2	—
SP.ZCOM	3	—
FROM	5	○
FROMP	6	○
DFROM	5	○
DFROMP	6	○
TO	5	○

Instruction name	Number of basic steps	Subset availability
TOP	6	○
DTO	5	○
DTOP	6	○
FROMD	5	○
FROMDP	6	○
DFROMD	5	○
DFROMDP	6	○
TOD	5	○
TODP	6	○
DTOD	5	○
DTODP	6	○
TYPERD	3	—
TYPERDP	4	—
UNIINFRD	4	—
UNIINFRDP	5	—
S.RTREAD	3	—
SP.RTREAD	4	—
S.RTWRITE	3	—
SP.RTWRITE	4	—
LOGTRG	3	—
LOGTRGR	3	—
DATATRG	3	—
SP.SOCOPEN	5	—
SP.SOCCLOSE	6	—
SP.SOCRCV	7	—
S.SOCRCVS	4	—
SP.SOCSND	7	—
SP.SOCCINF	6	—
SP.SOCCSET	5	—
SP.SOCRMODE	5	—
S.SOCRDATA	6	—
SP.SOCRDATA	7	—
SP.ECPRTCL	7	—
SP.SLMPSND	6	—
SP.FTPPUT	7	—
SP.FTPGET	7	—
PSTOP	2 + Number of characters in the file name	—
PSTOPP	3 + Number of characters in the file name	—
POFF	2 + Number of characters in the file name	—
POFFP	3 + Number of characters in the file name	—
PSCAN	2 + Number of characters in the file name	—
PSCANP	3 + Number of characters in the file name	—
PID	5	—
S.PIDINIT	2	—
SP.PIDINIT	3	—
S.PIDCONT	2	—
SP.PIDCONT	3	—
S.PIDSTOP	2	—
SP.PIDSTOP	3	—
S.PIDRUN	2	—
SP.PIDRUN	3	—
S.PIDPRMW	3	—
SP.PIDPRMW	4	—

Instruction name	Number of basic steps	Subset availability
PIDINIT	2	—
PIDINITP	3	—
PIDCONT	2	—
PIDCONTP	3	—
PIDSTOP	2	—
PIDSTOPP	3	—
PIDRUN	2	—
PIDRUNP	3	—
PIDPRMW	3	—
PIDPRMWP	4	—
D.DDRD	9	—
DP.DDRD	10	—
D.DDWR	9	—
DP.DDWR	10	—
M.DDRD	9	—
MP.DDRD	10	—
M.DDWR	9	—
MP.DDWR	10	—
LD [S□]	2	—
LD [BL□\S□]	3	—
LDI [S□]	2	—
LDI [BL□\S□]	3	—
AND [S□]	2	—
AND [BL□\S□]	3	—
ANI [S□]	2	—
ANI [BL□\S□]	3	—
OR [S□]	2	—
OR [BL□\S□]	3	—
ORI [S□]	2	—
ORI [BL□\S□]	3	—
LD [BL□]	2	—
LDI [BL□]	2	—
AND [BL□]	2	—
ANI [BL□]	2	—
OR [BL□]	2	—
ORI [BL□]	2	—
MOV [K4S□]	3	—
MOV [BL□\K4S□]	4	—
MOVP [K4S□]	4	—
MOVP [BL□\K4S□]	5	—
DMOV [K8S□]	3	—
DMOV [BL□\K8S□]	4	—
DMOVP [K8S□]	4	—
DMOVP [BL□\K8S□]	5	—
BMOV [K4S□]	4	—
BMOV [BL□\K4S□]	5	—
BMOVP [K4S□]	5	—
BMOVP [BL□\K4S□]	6	—
SET [BL□]	2	—
RST [BL□]	2	—
PAUSE [BL□]	2	—
RSTART [BL□]	2	—
SET [S□]	2	—

Instruction name	Number of basic steps	Subset availability
SET [BL□\S□]	3	—
RST [S□]	2	—
RST [BL□\S□]	3	—
BRSET	2	—
SP.CONTSW	4	—
DCONTSW	1	—
ECONTSW	1	—
CONTWR	5	—
CONTWRP	6	—
SP.SIDRD	6	—

*1 Since the PHASE instruction adds two steps, when using a function that specifies the step number in the ladder program after the PHASE instruction, it is necessary to specify a step number that takes into account the two steps that are added (for a total of 15 steps).(☞ Page 201 Phase Processing Instructions)

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".

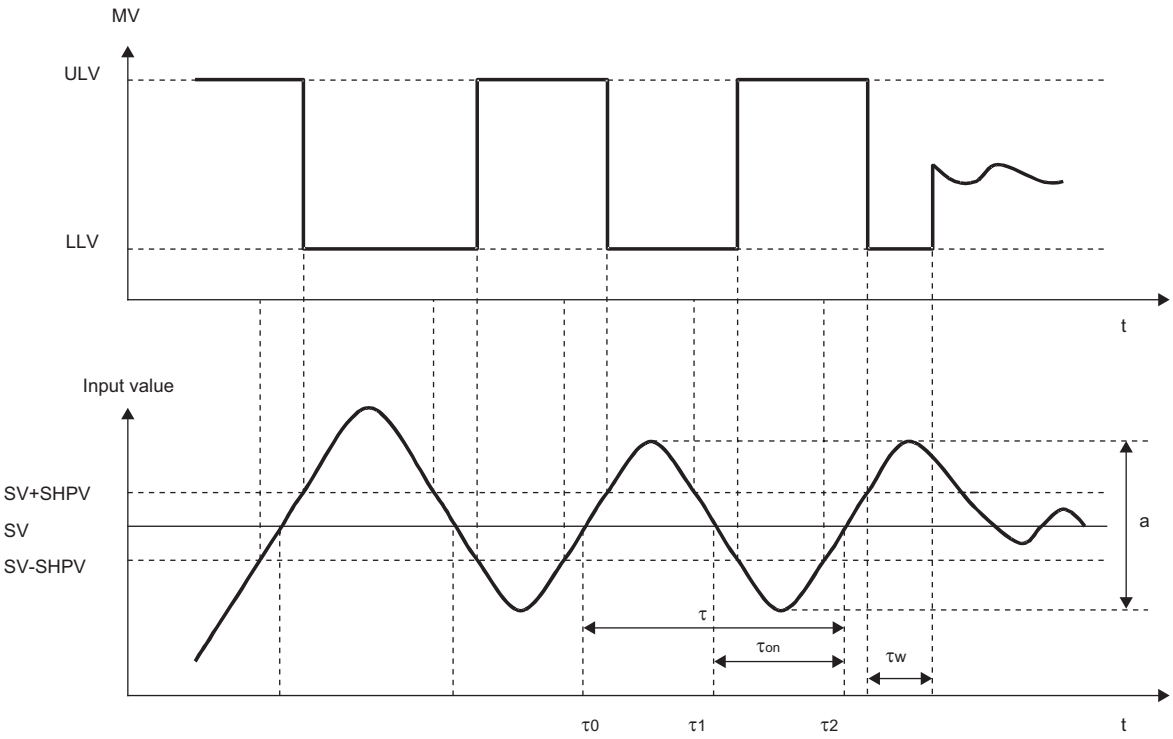
■Limit cycle method

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

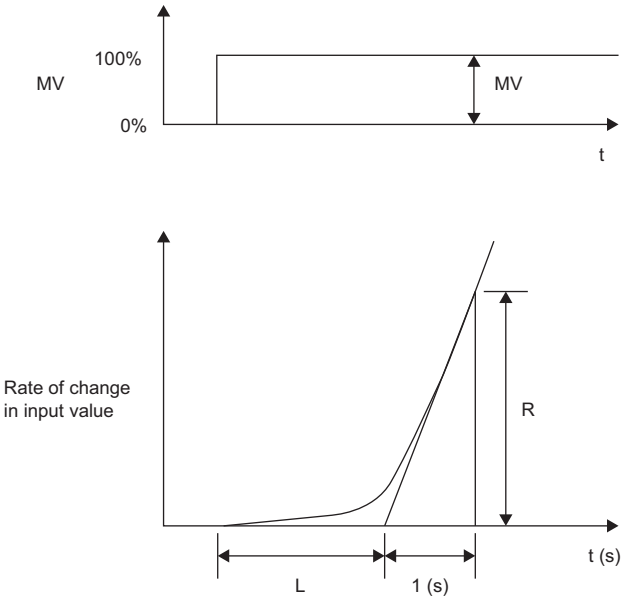
This section describes the step response method that is used to determine the optimum value of three PID constants (proportional gain (K_P), integral time (T_I), and derivative time (T_D)).

■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



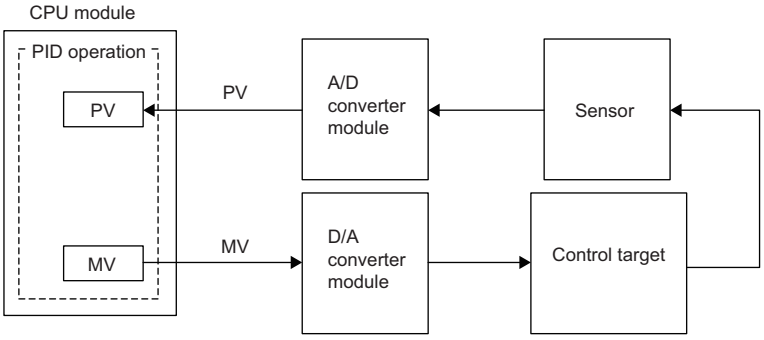
MV: Manipulated value
L: Dead time
R: Maximum ramp
t: Time

■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	$\frac{1}{RL} \times MV \times 100$	—	—
PI control (PI action)	$\frac{0.9}{RL} \times MV \times 100$	33L	—
PID control (PID action)	$\frac{1.2}{RL} \times MV \times 100$	20L	50L

Appendix 4 PID Operation Program Examples

This section provides examples of PID operation programs where the PID operation instruction is used.
The following system configuration is used.

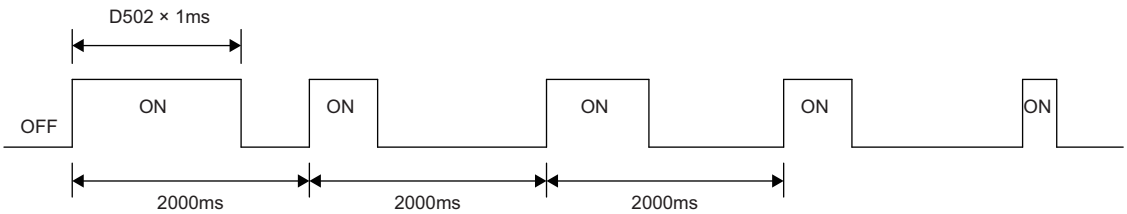


- X2: Auto tuning command
- X3: PID control command
- Y20: Error indication
- Y21: Heater output

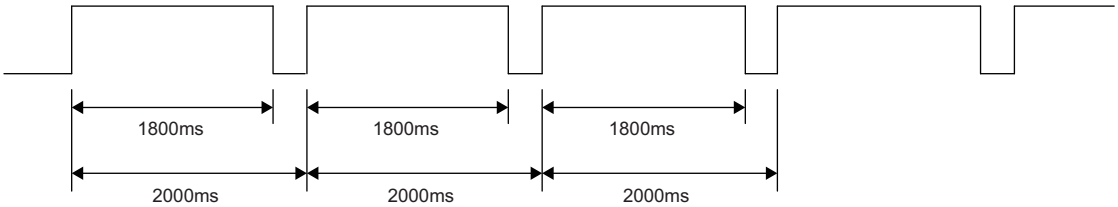
Item				During auto tuning	During PID control
Set value			(s1)	500 (+50℃)	500 (+50℃)
Control data	Sampling time (T_S)		(s3)	3000ms	500ms
	Input filter (α)		(s3)+2	70%	70%
	Derivative gain (K_D)		(s3)+5	0%	0%
	Output upper limit value		(s3)+22	2000 (2 seconds)	2000 (2 seconds)
	Output lower limit value		(s3)+23	0	0
	Action setting (ACT)	Input variation alarm	Bit 1 of (s3)+1	Disabled	Disabled
		Output variation alarm	Bit 2 of (s3)+1	Disabled	Disabled
		Output upper/lower limit value setting	Bit 5 of (s3)+1	Enabled	Enabled
Output value			(d)	1800	According to operation

Operation of output value

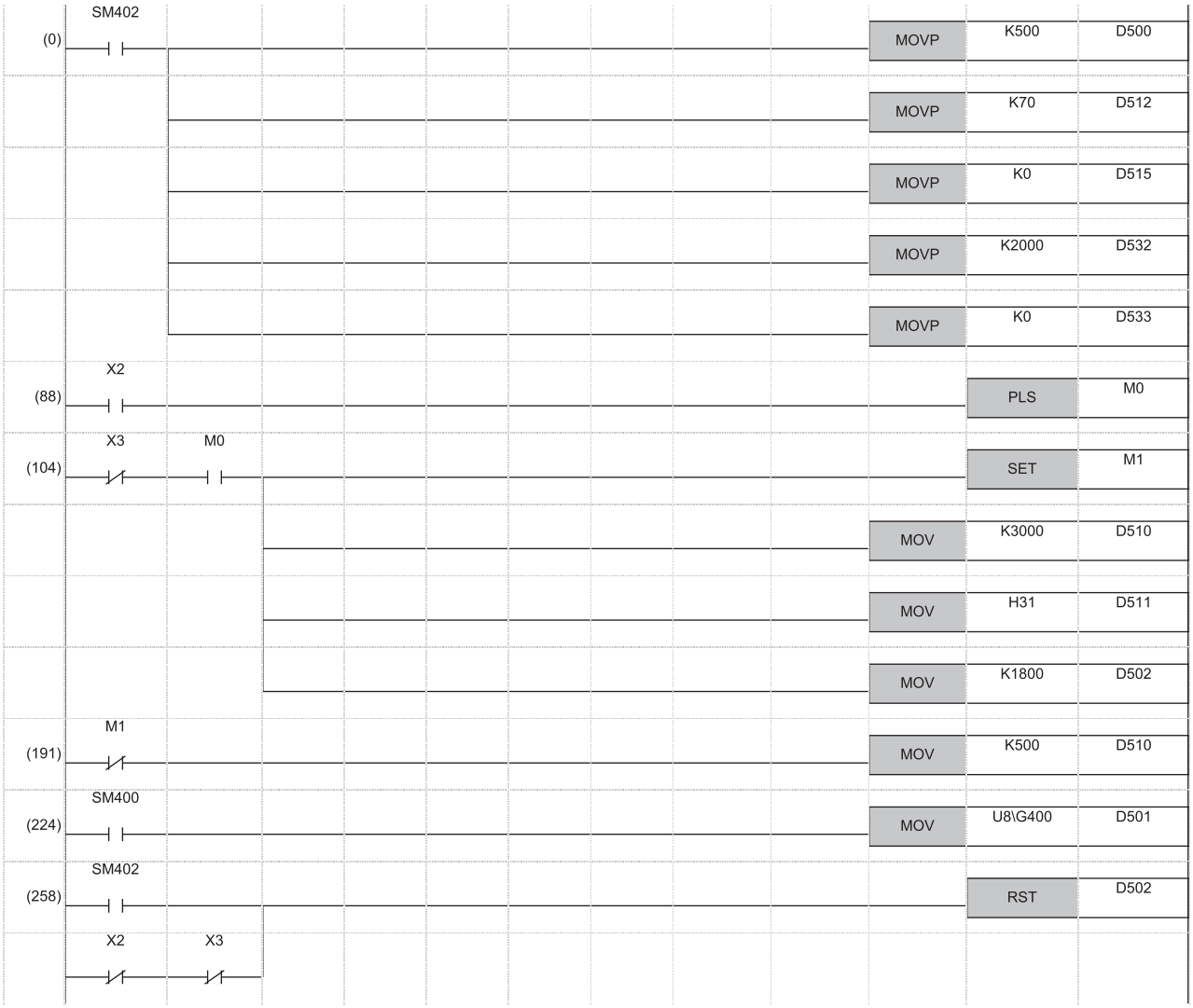
■PID control



■Auto Tuning: 90% of maximum output



Auto tuning (step response method) + PID control program example



(0) Set the set value. <50℃>
Set the input filter constant (α).
Set the derivative gain (K_D). <0%>
Set the output upper limit value. <ON for 2 seconds>
Set the output lower limit value. <ON for 0 seconds>

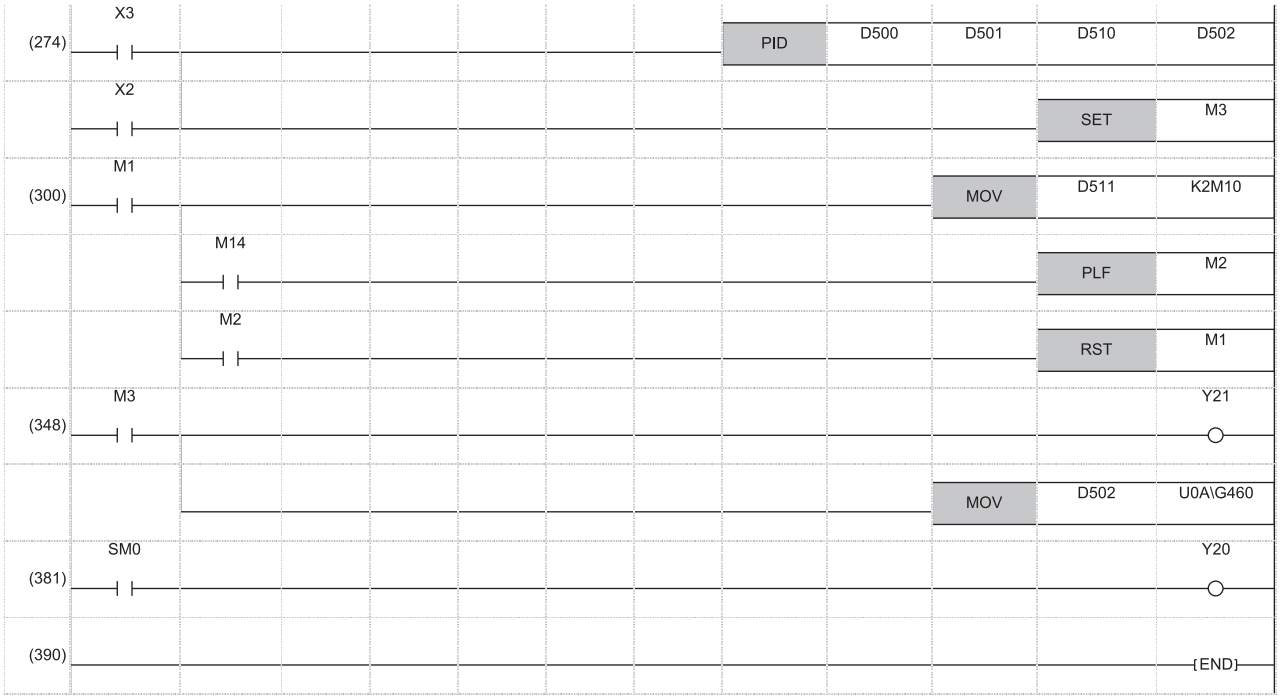
(88) Start auto tuning.

(104) Turn on the auto tuning action status flag.
Set the sampling time for auto tuning (T_S). <3 seconds>
Turn on the auto tuning execution bit in action setting (ACT) parameter.
Set the output value for auto tuning. <ON for 1.8 seconds>

(191) Set the sampling time (T_S) for normal PID action. <500ms>

(224) Input the process value (PV) to the input/output data area from the A/D converter module.

(258) Initialize the PID action.



(274) Execute the PID instruction.
PID action is in progress.

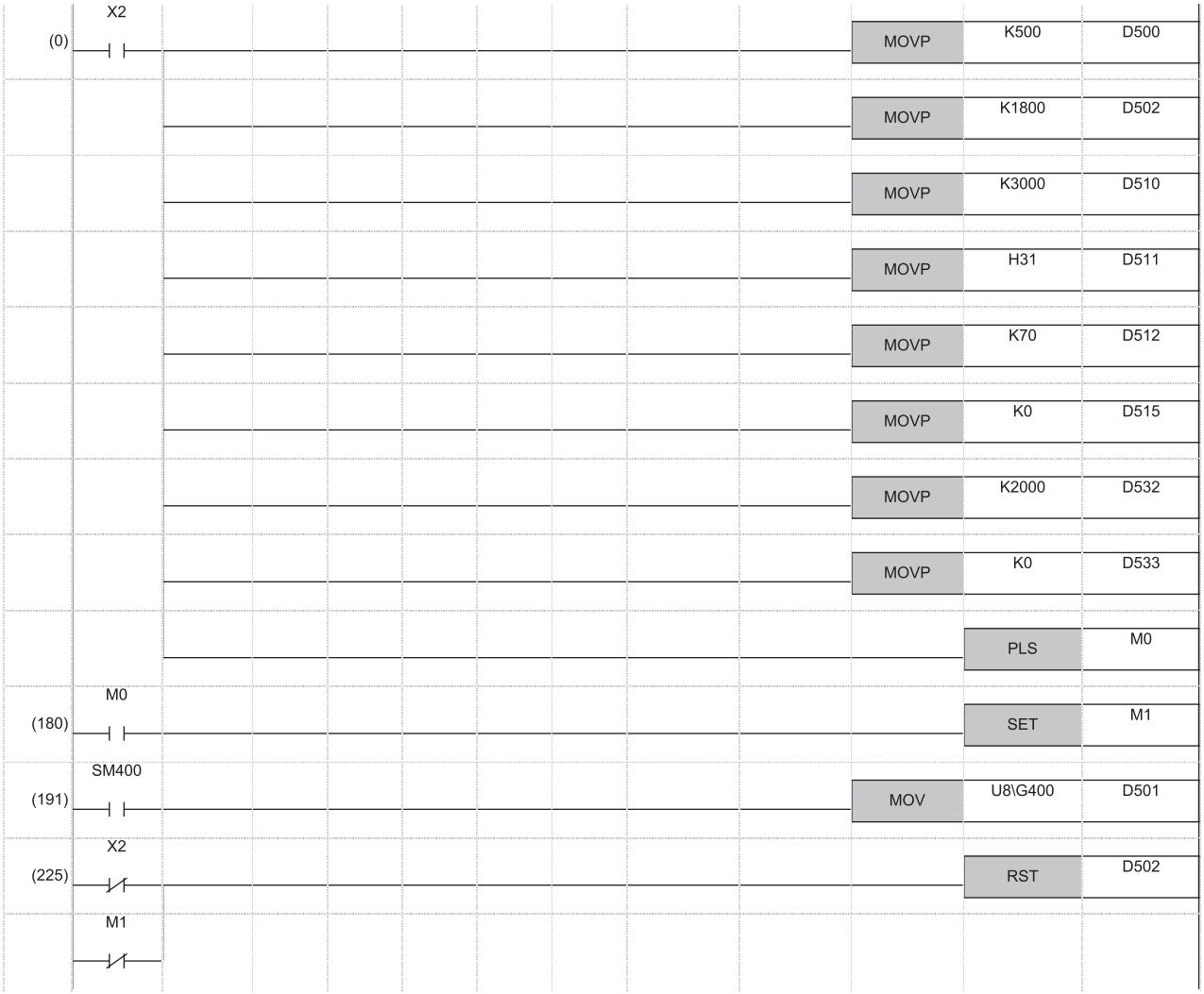
(300) Check the auto tuning operation.
End the auto tuning processing.
Shift to the normal PID action.

(348) Heater output
Write the manipulated value (MV) to D/A converter module.

(381) Error

A

Auto tuning (step response method) program example

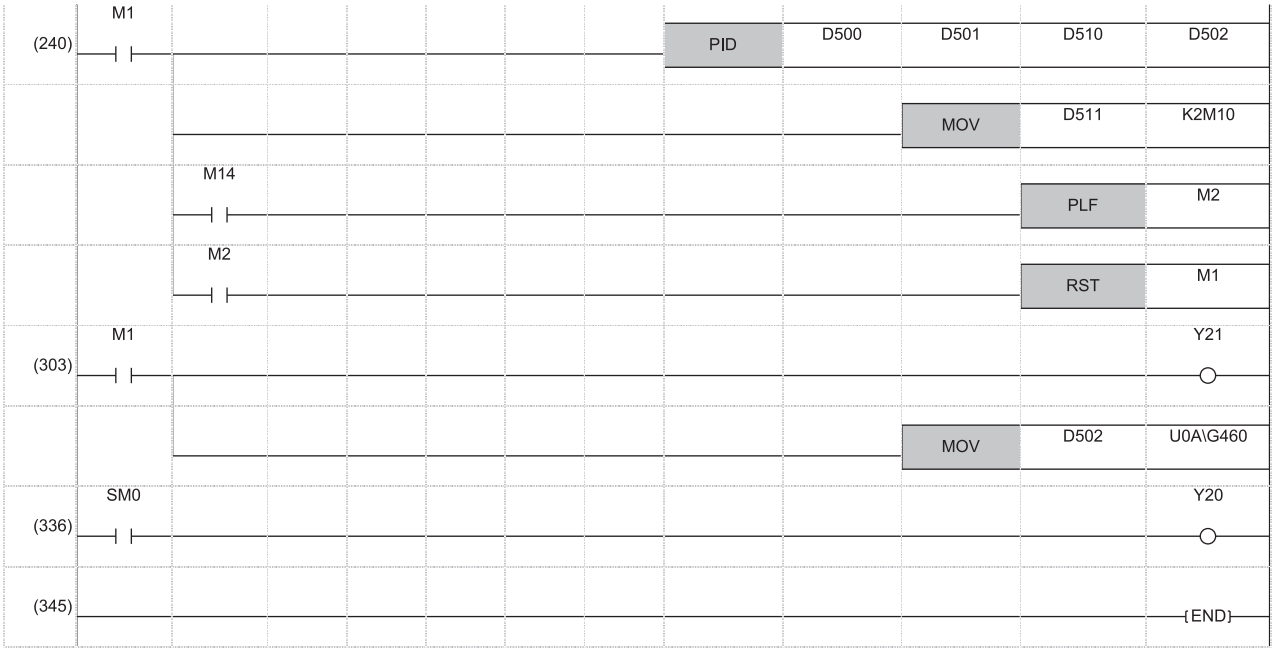


(0) Set the set value. <50℃>
Set the output value for auto tuning. <ON for 1.8 seconds>
Set the sampling time (T_S). <3 seconds>
Turn on the auto tuning execution bit in action setting (ACT) parameter.
Set the input filter constant (α).
Set the derivative gain (K_D). <0%>
Set the output upper limit value. <ON for 2 seconds>
Set the output lower limit value. <ON for 0 seconds>
Start auto tuning.

(180) Perform PID action.

(191) Input the process value (PV) to the input/output data area from the A/D converter module.

(225) Initialize the PID action.



(240) Execute the PID instruction.
Check the auto tuning operation.
End the auto tuning processing.
Shift to the normal PID action.

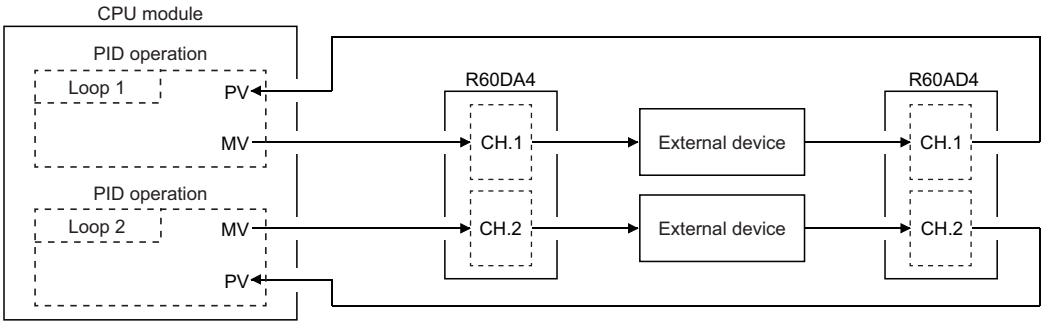
(303) Heater output
Write the manipulated value (MV) to D/A converter module.

(336) Error

A

Appendix 5 PID Control Program Examples

This section provides examples of PID control programs where PID control instructions are used.
The following system configuration is used.



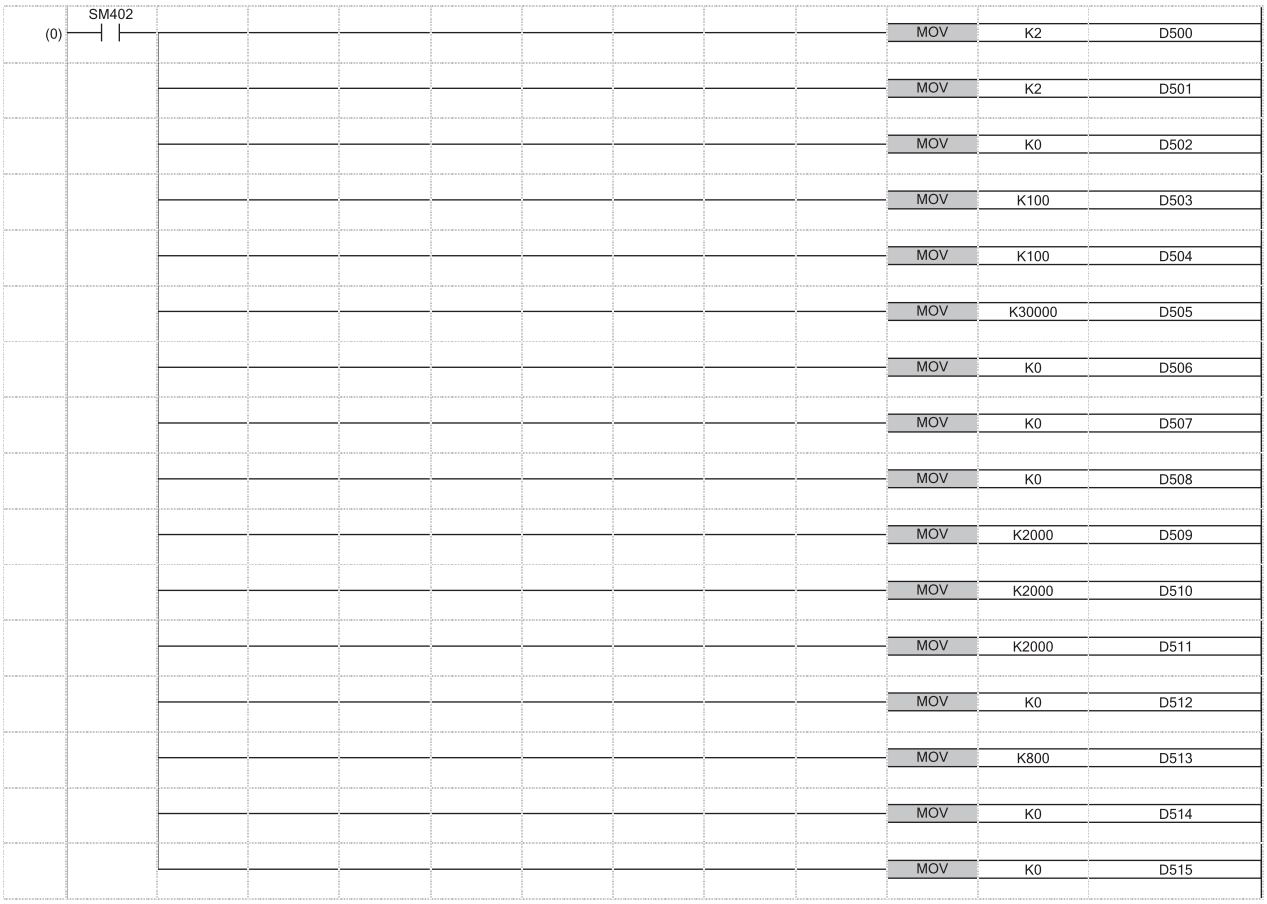
I/O number of the R60AD4: X/Y80 to X/Y8F, I/O number of the R60DA4: X/YA0 to X/YAF

Program examples for PID control in automatic mode

The program performs PID operation using the digital value input from the R60AD4 as a process value (PV), and outputs the obtained manipulated value (MV) from the R60DA4.
The program conditions are as follows:

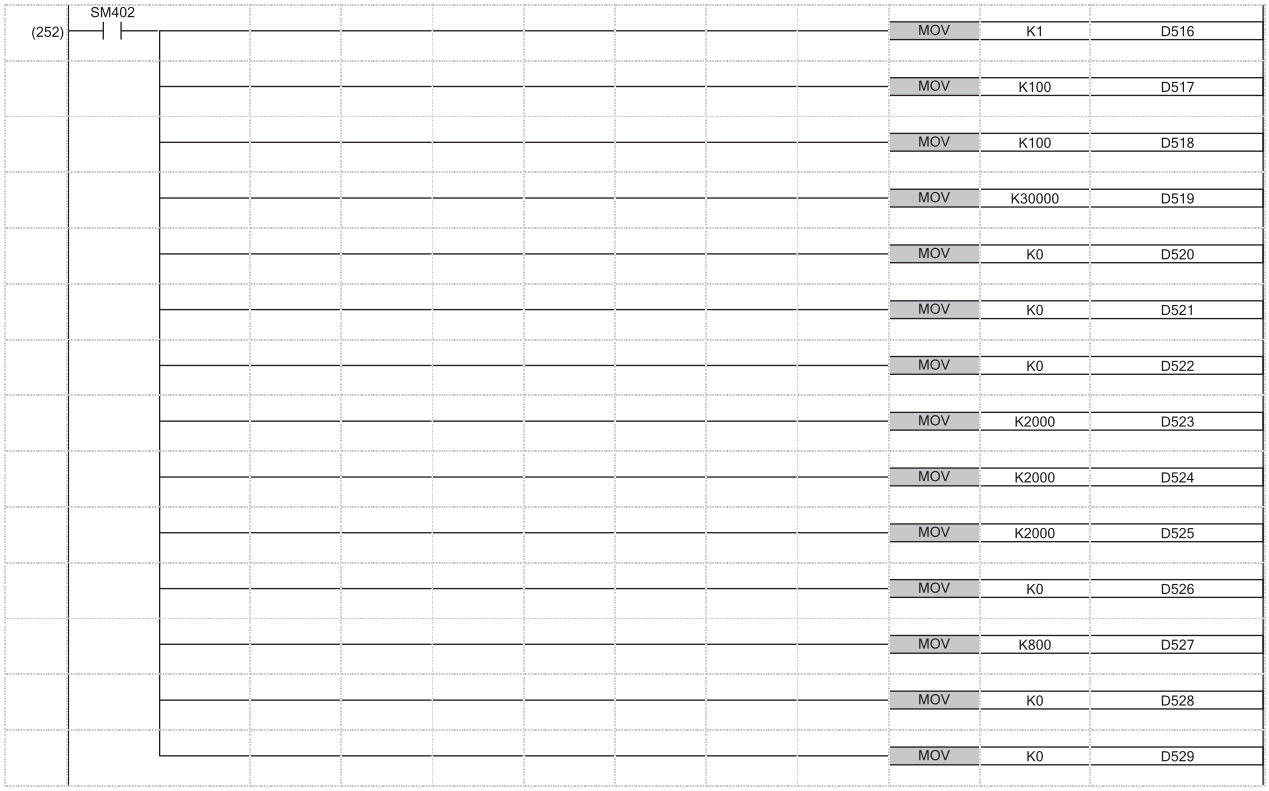
Item		Inexact differential	Exact differential
Number of loops that performs PID operation		2	
Sampling period		1 second	
Device where PID control data are set	Common data	D500, D501	
	Loop 1 data	D502 to D515	D502 to D511
	Loop 2 data	D516 to D529	D512 to D521
Device where I/O data are set	Common data	D600 to D609	
	Loop 1 data	D610 to D632	D610 to D627
	Loop 2 data	D633 to D655	D628 to D645
Set value (SV)	Loop 1	600	
	Loop 2	1000	
Device used to start/stop PID control	PID control start command	X0	
	PID control stop command	X1	
Digital value of the R60AD4 and R60DA4		0 to 2000	

Program example (inexact differential)

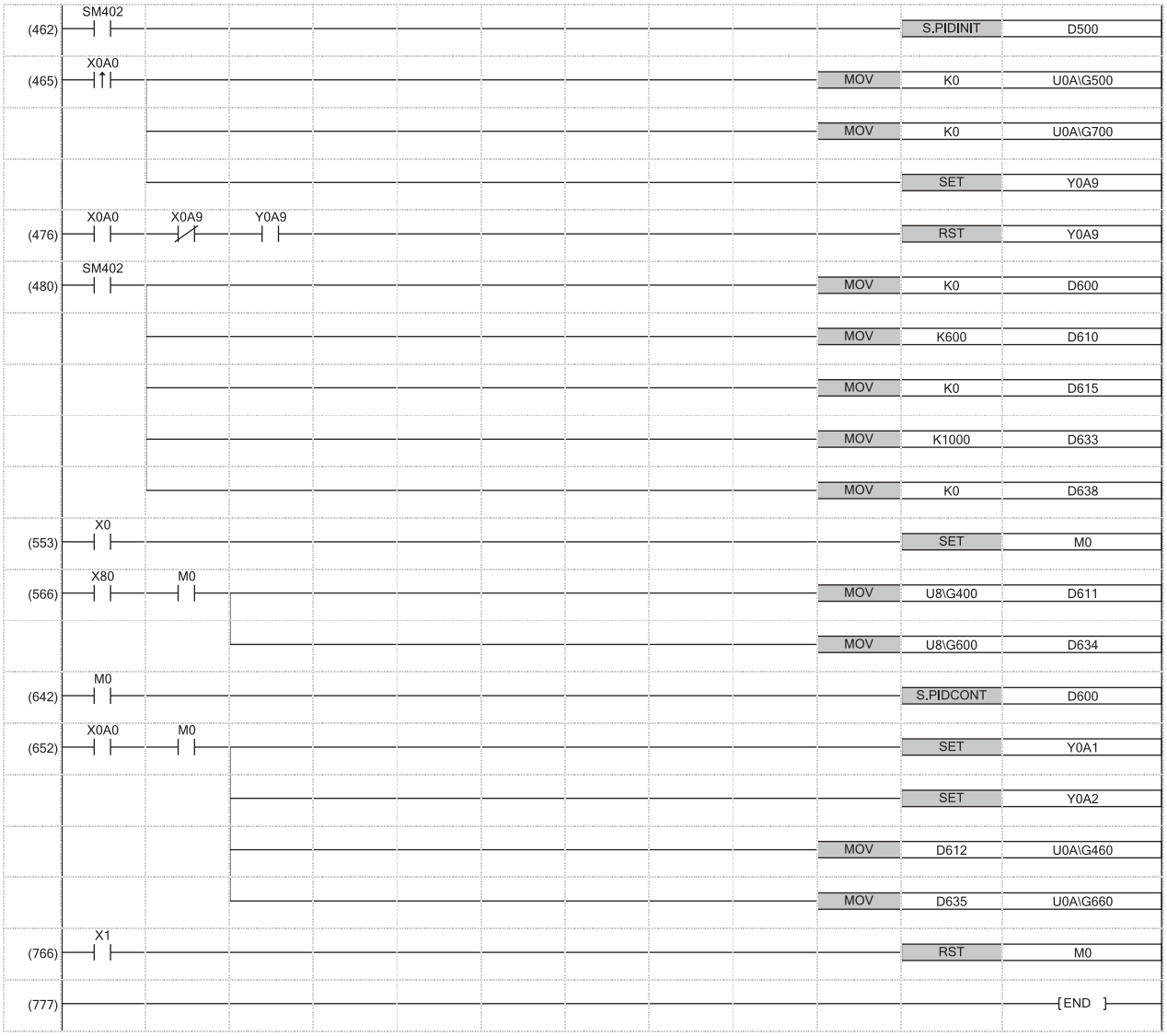


(0) Set PID control data common to all loops.
Set PID control data for loop 1.

A



(252) Set PID control data for loop 2.



(462) Set the PID control data that are set in D500 to D529.

(465) to (476) Enable output of the R60DA4.

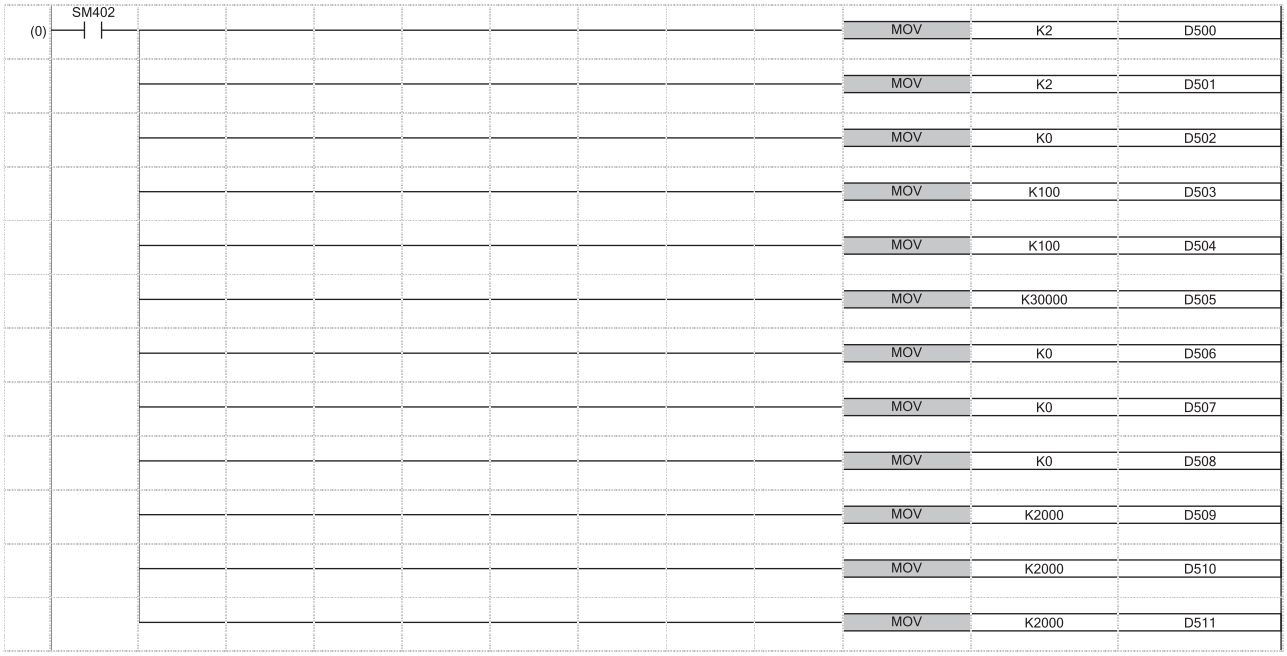
(480) Set I/O control data common to all loops.

Set I/O data for loop 1.

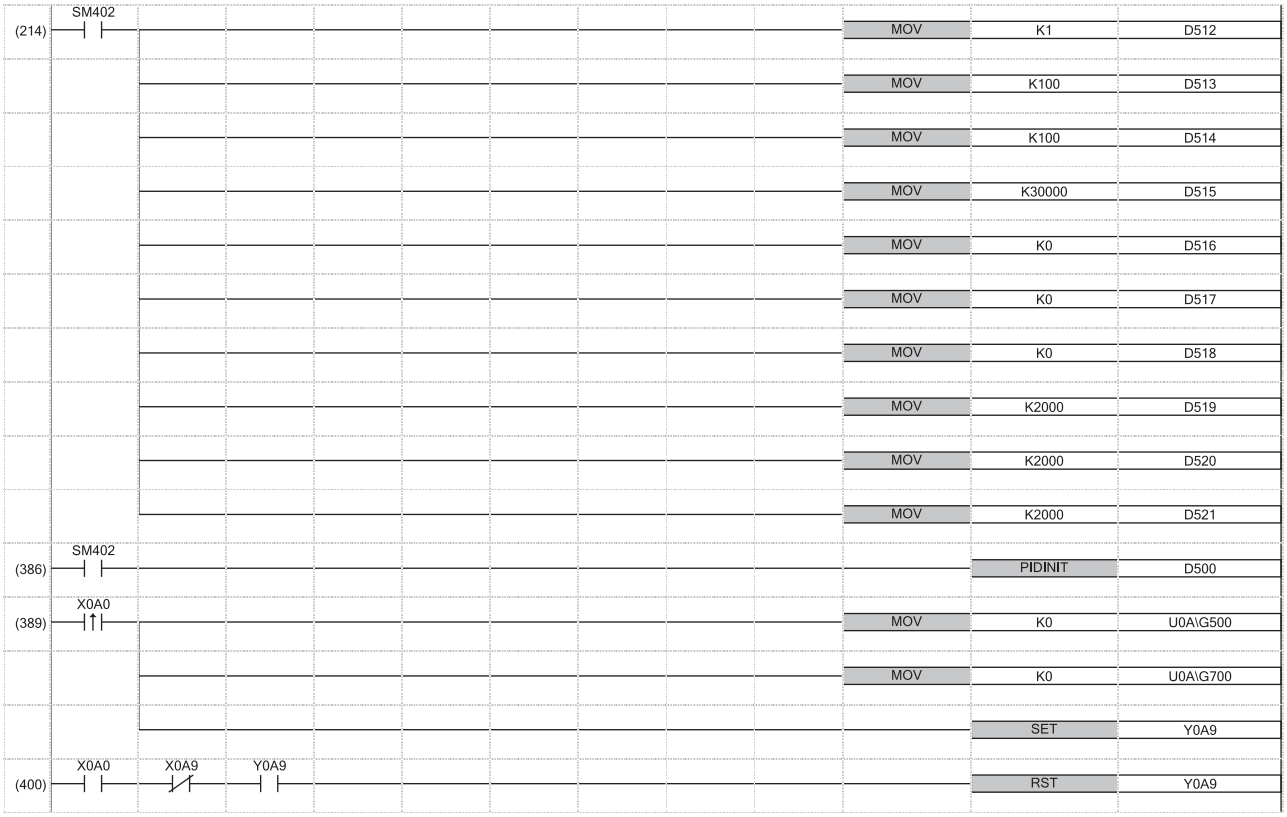
Set I/O data for loop 2.

A

Program example (exact differential)



(0) Set PID control data common to all loops.
Set PID control data for loop 1.

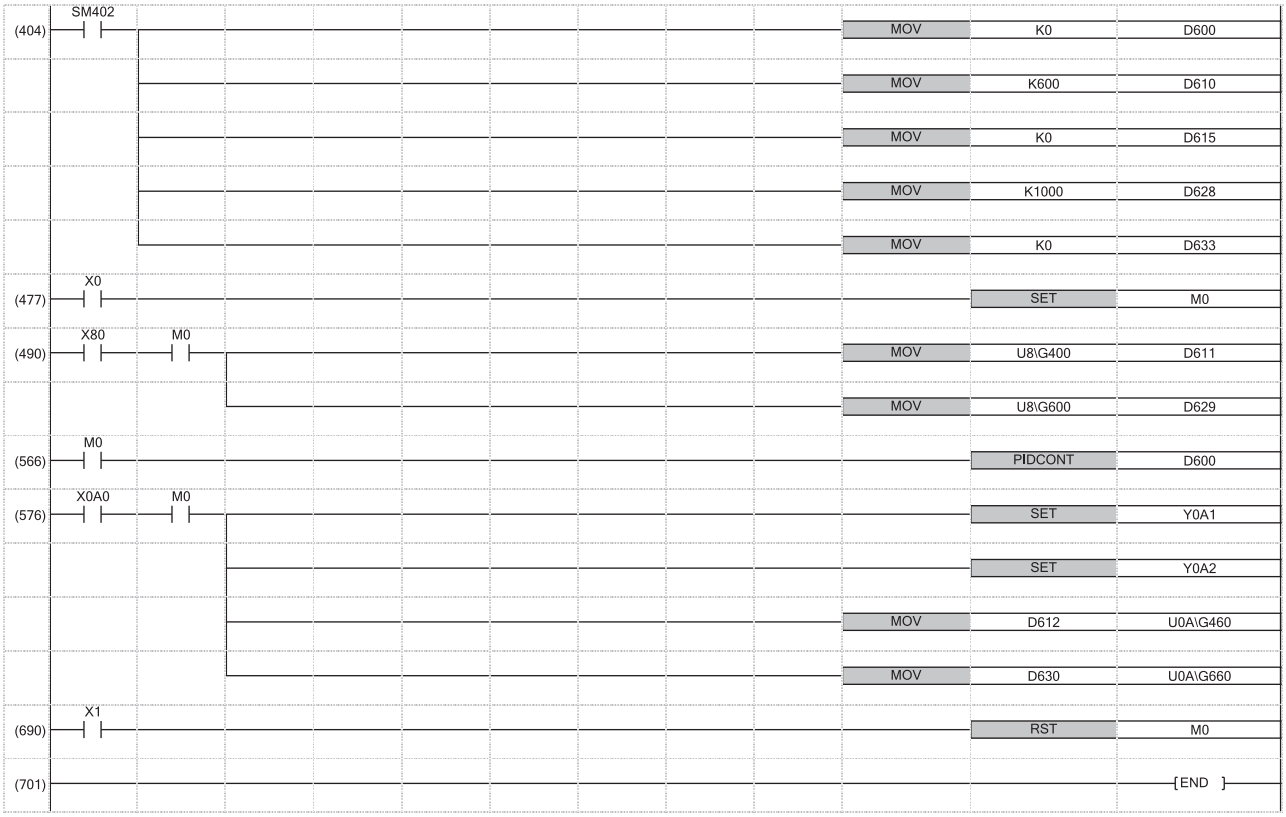


(214) Set PID control data for loop 2.

(386) Set the PID control data that are set in D500 to D521.

(389) to (400) Enable output of the R60DA4.

A



(404) Set I/O control data common to all loops.

Set I/O data for loop 1.

Set I/O data for loop 2.

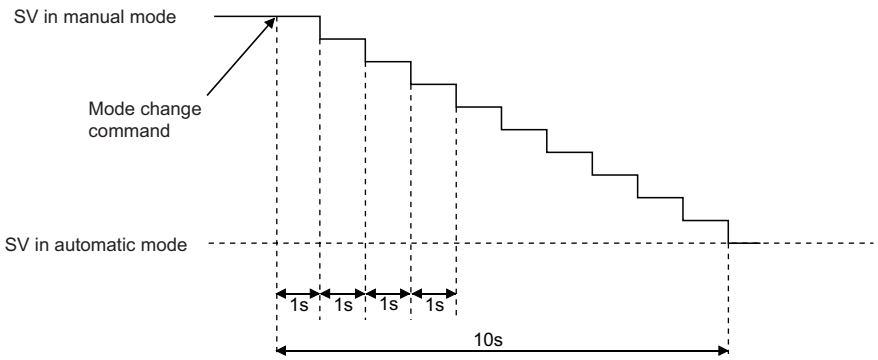
Program examples for PID control when switching modes

The program performs PID operation by switching an automatic mode and a manual mode.
The program conditions are as follows:

Item		Inexact differential	Exact differential
Number of loops that performs PID operation		1	
Sampling period		1 second	
Device where PID control data are set	Common data	D500, D501	
	Loop 1 data	D502 to D515	D502 to D511
Device where I/O data are set	Common data	D600 to D609	
	Loop 1 data	D610 to D630	D610 to D627
Set value (SV) set by external digital switches and manipulated value (MV) in manual mode	SV	X30 to X3F	
	MV in manual mode	X20 to X2F	
Device used to start/stop PID control	PID control start command	X0	
	PID control stop command	X1	
Device used to switch the mode	SV set command	X3	
	MV (in manual mode) set command	X4	
	Automatic/manual switch command	X6 (Off: Automatic mode, On: Manual mode)	
Digital value of the R60AD4 and R60DA4		0 to 2000	
SM792, SM794		Off	

Point

In manual mode, the set value (SV) is automatically rewritten to the process value (PV) when PID operation is performed. Therefore, when the manual mode is returned to the automatic mode, the set value (SV) must be rewritten to the one used before the mode was switched.
The set value (SV) is rewritten in 10 steps as shown below:



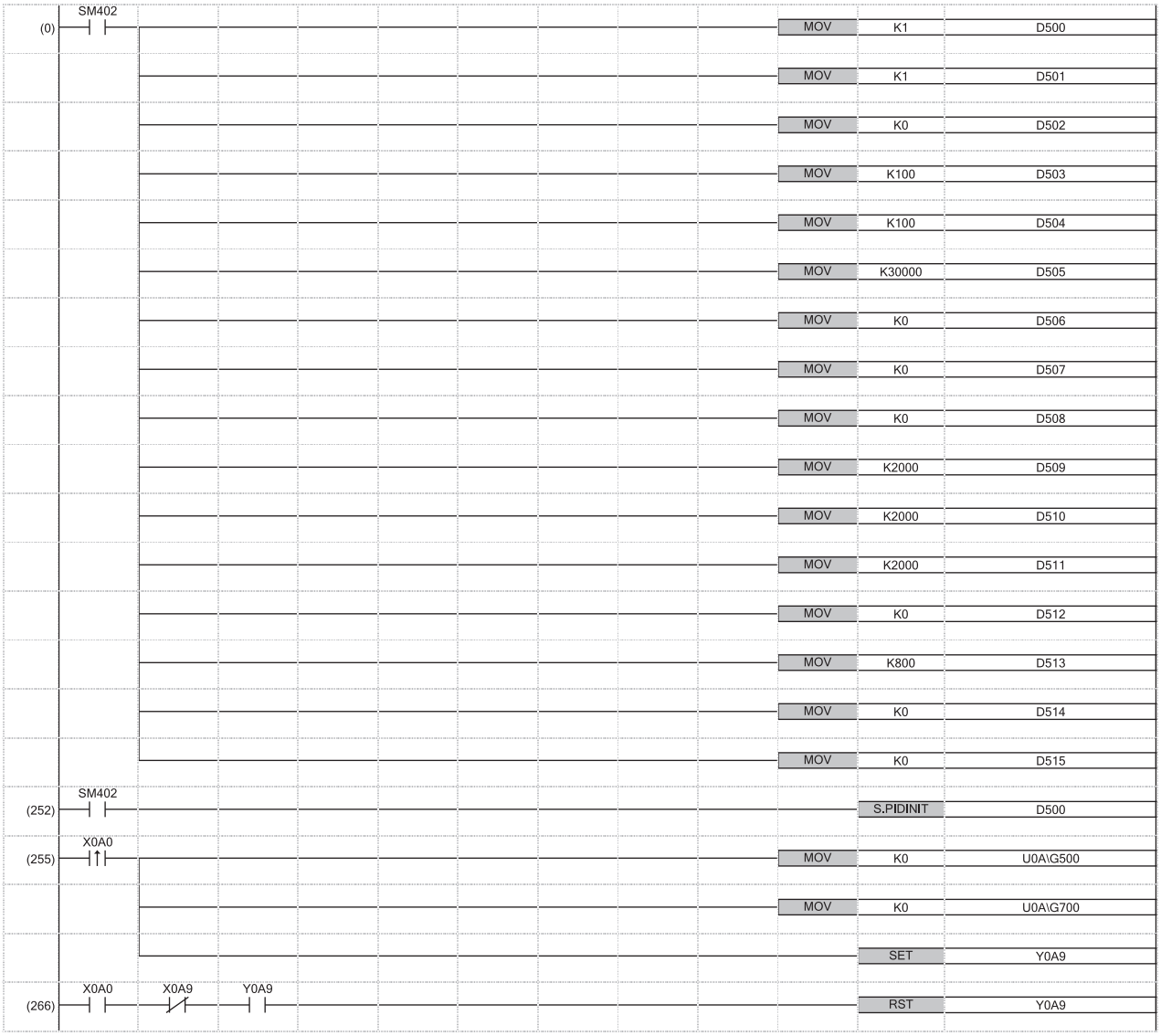
The following formula is used to rewritten the set value (SV).

$$\frac{(SV_m) - (SV_a)}{10} = \boxed{\text{Subtraction value}} \cdots\cdots \boxed{\text{Remainder}}$$

SV_m: SV in manual mode
SV_a: SV in automatic mode
The value obtained with the above operational expression is subtracted from the set value (SV) every second.
The remainder is subtracted in the first subtraction.

A

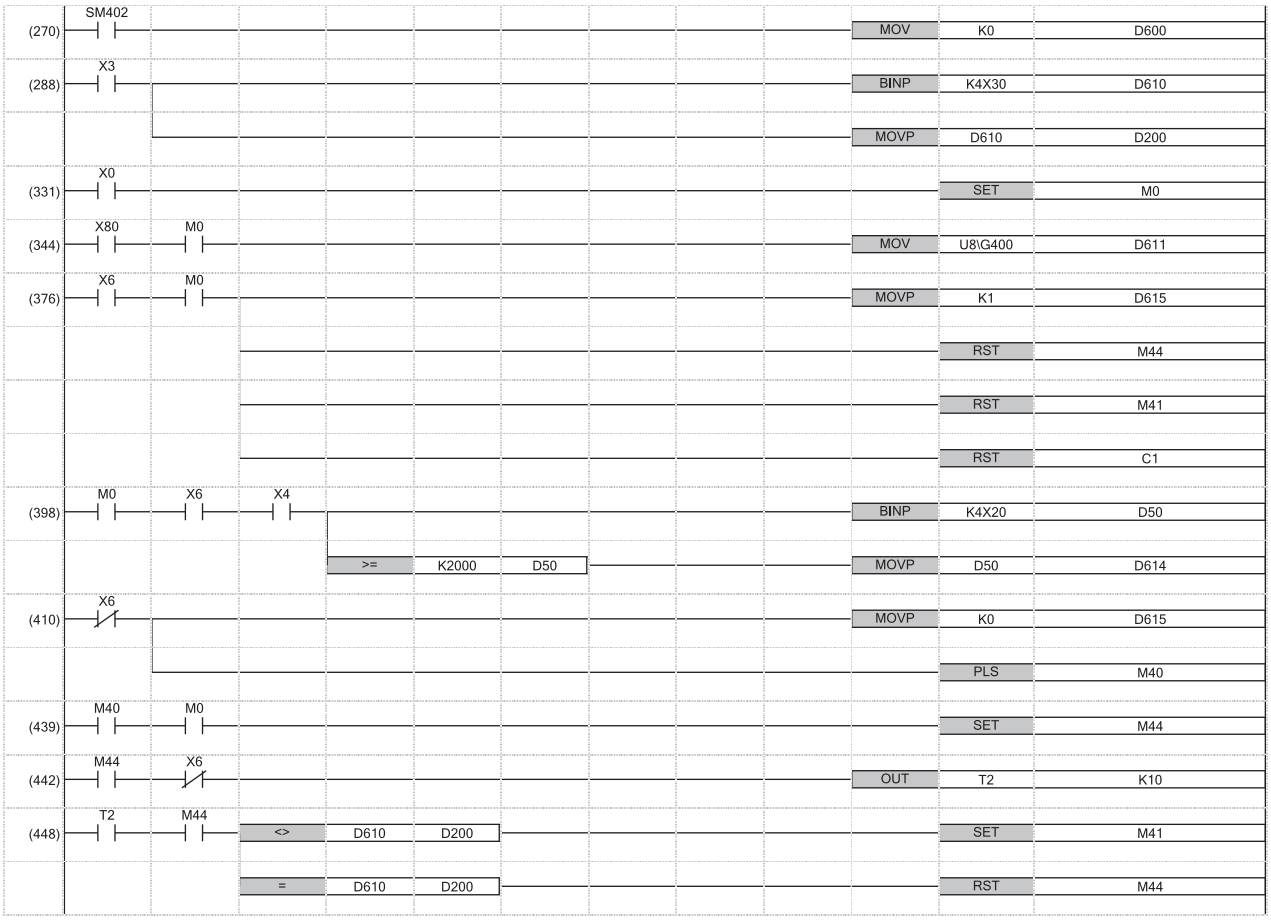
Program example (inexact differential)



(0) Set PID control data common to all loops.
Set PID control data for loop 1.

(252) Set the PID control data that are set in D500 to D515.

(255) Enable output of the R60DA4.



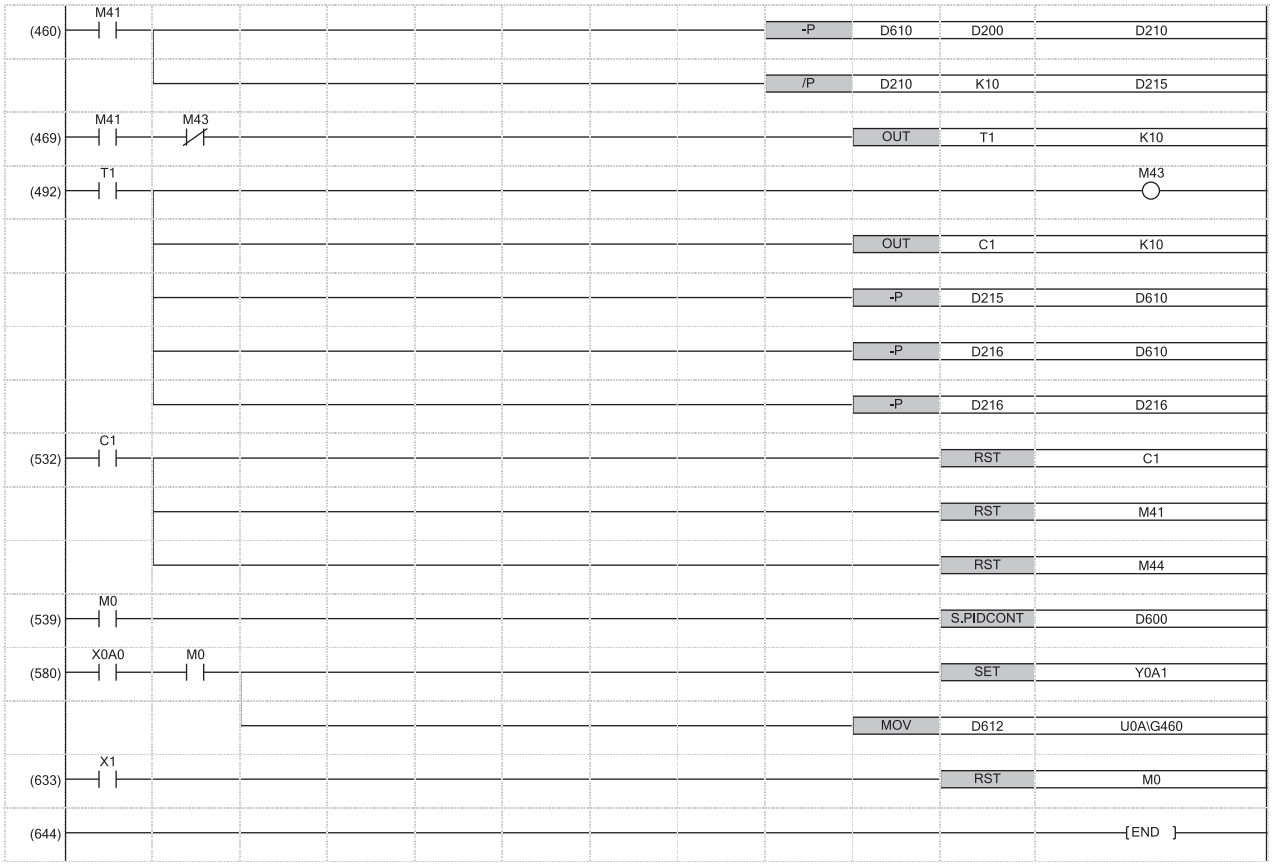
(270) Set I/O data.

(376) Reset the device used to switch manual mode to automatic mode.

(398) Externally set the manipulated value (MV).

(439) to (448) Perform the mode switch processing considering a delay in switching time. The mode switches to the automatic mode by turning on the automatic/manual switch command and performing PID operation.

A



(460) Store the quotient of the following operational expression in D215, and the remainder in D216.

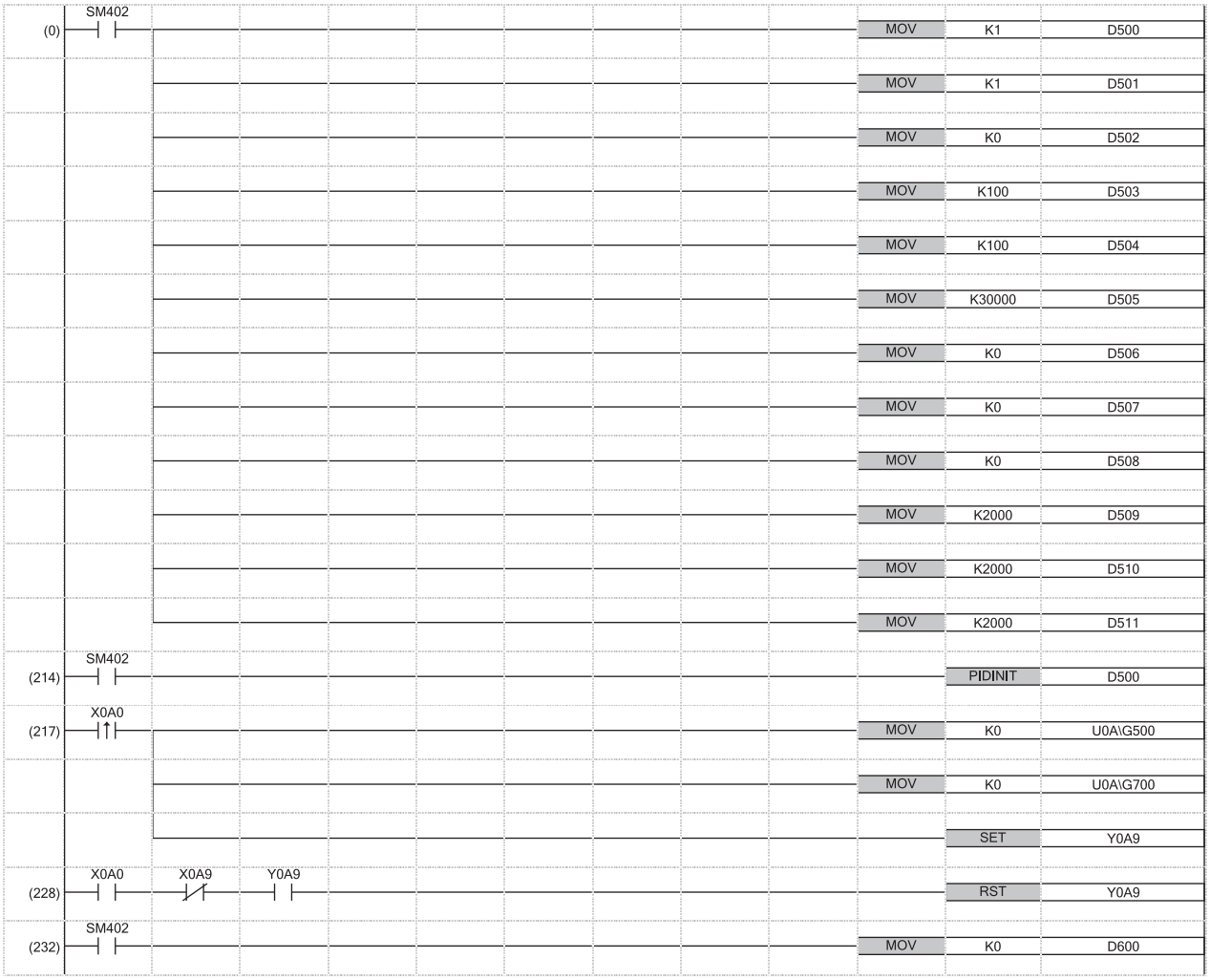
$$\frac{\left(\ SV_m\ \right) - \left(\ SV_a\ \right)}{10}$$

SV_m: SV in manual mode
SV_a: SV in automatic mode

(469) to (492) Rewrite the set value (SV) to the value used in automatic mode.
The remainder is subtracted from the set value (SV) in the first subtraction.

(532) End the mode switch processing.

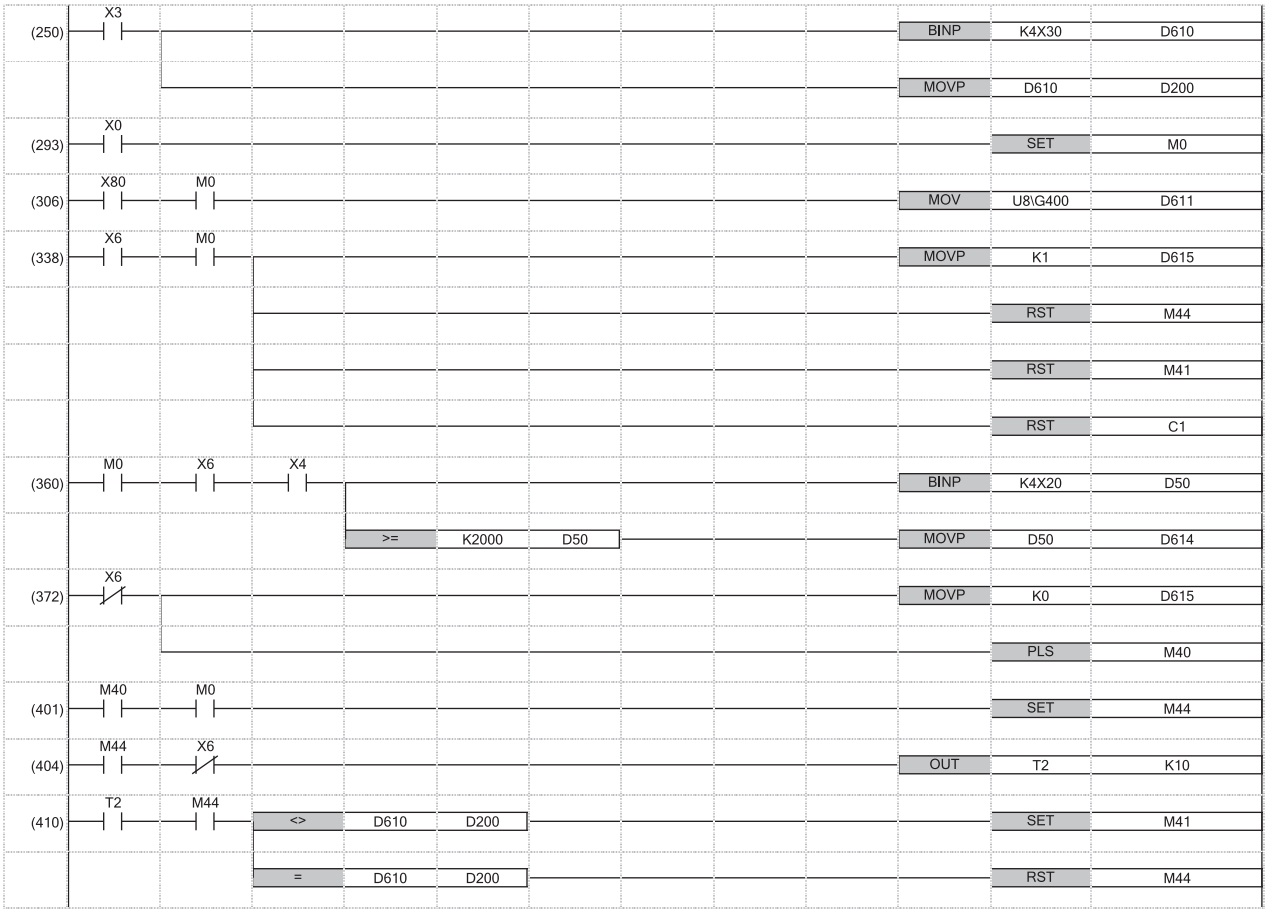
Program example (exact differential)



(0) to (214) Set PID control data common to all loops.
Set PID control data for loop 1.

(217) to (232) Set the PID control data that are set in D500 to D511.
Enable output of the R60DA4.

A

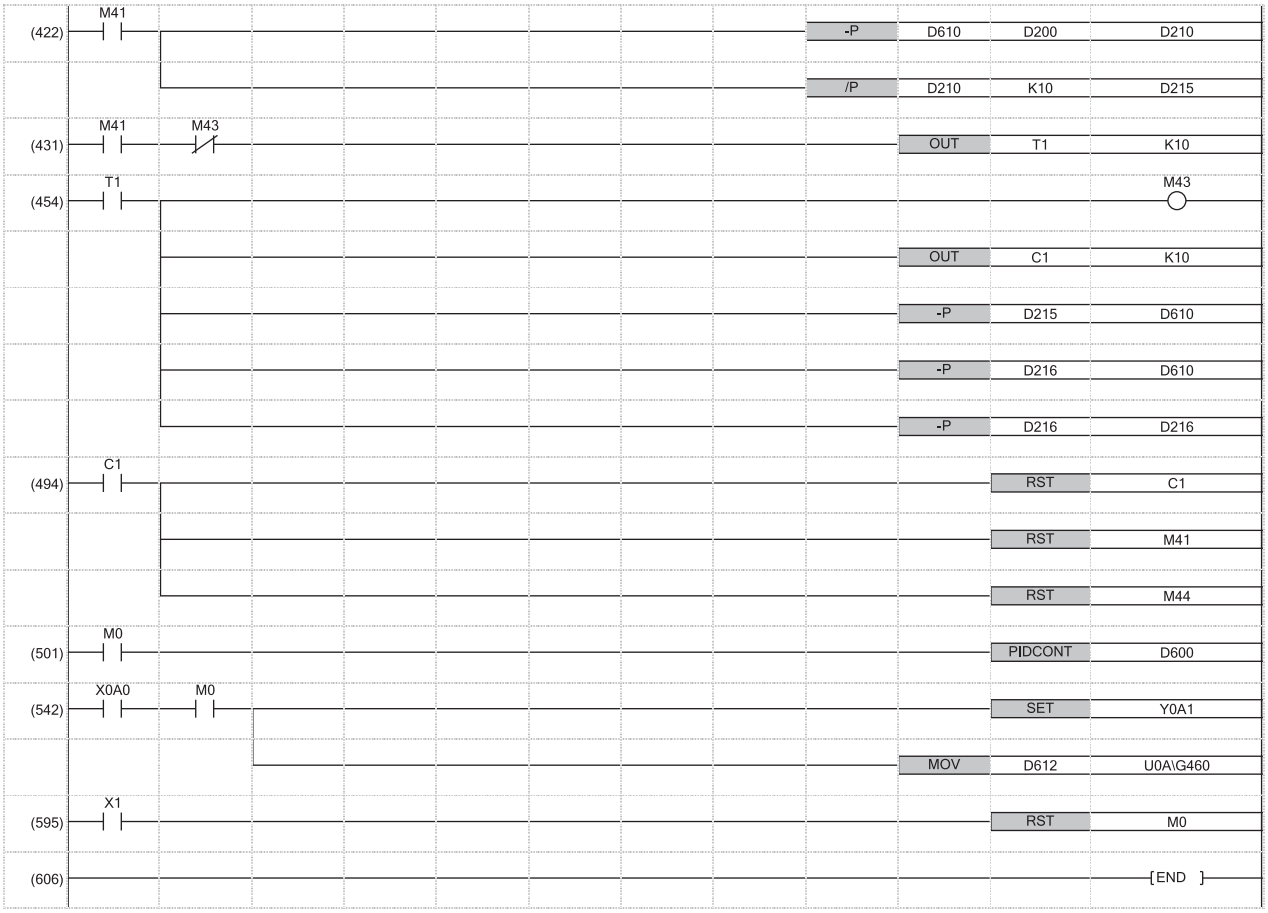


(250) Set I/O data.

(338) Reset the device used to switch manual mode to automatic mode.

(360) Externally set the manipulated value (MV).

(401) to (410) Perform the mode switch processing considering a delay in switching time. The mode switches to the automatic mode by turning on the automatic/manual switch command and performing PID operation.



(422) Store the quotient of the following operational expression in D215, and the remainder in D216.

$$\frac{\left(\ SV_m\ \right) - \left(\ SV_a\ \right)}{10}$$

SV_m: SV in manual mode
SV_a: SV in automatic mode

(431) to (454) Rewrite the set value (SV) to the value used in automatic mode.
The remainder is subtracted from the set value (SV) in the first subtraction.

(494) End the mode switch processing.

A

Appendix 6 Replacement of Other Format Projects

Replacement of a GX Works2 format project

When a GX Works2 format project is used in GX Works3, the following instructions are automatically replaced by the compatible instructions/functions/function blocks (FBs).*1

GX Works3 with version "1.047Z" or later supports the replacement by the compatible instructions/functions.

GX Works3 with version "1.055H" or later supports the replacement by the compatible function blocks.

*1 Data types or arguments and orders are the same as GX Works2.

Compatible instructions (basic instructions/application instructions)

Basic instructions and application instructions used in programs using ST (including the inline ST) or structured ladder/FBD are replaced by the compatible instructions.

Instruction name in GX Works2	Compatible instruction name	Program type		Reference
		ST	FBD	
*	MULTI_M2	×	○	Page 247 *(P)(_U)
*P	MULTIP_M2	×	○	
+	PLUS_M2	×	○	Page 233 +(P)(_U) [when three operands are set]
+P	PLUSP_M2	×	○	
-	MINUS_M2	×	○	Page 237 -(P)(_U) [when three operands are set]
-P	MINUSP_M2	×	○	
/	DIVISION_M2	×	○	Page 249 /(P)(_U)
/P	DIVISIONP_M2	×	○	
ABRST1_M	Z_ABRST1_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
ABRST2_M	Z_ABRST2_M2	○	○	
ABRST3_M	Z_ABRST3_M2	○	○	
ABRST4_M	Z_ABRST4_M2	○	○	
ACOS	ACOS_M2	○	○	Page 1452 ACOS(_E)
ACOS_MD		○	○	
ACOS_E_MD		○	○	
ACOSD	ACOSD_M2	○	○	
ACOSD_MD		○	○	
ACOSD_E_MD		○	○	
AND<	AND_LT_M2	×	○	Page 214 LD□(_U), AND□(_U), OR□(_U)
AND<=	AND_LE_M2	×	○	
AND<>	AND_NE_M2	×	○	
AND=	AND_EQ_M2	×	○	
AND>	AND_GT_M2	×	○	
AND>=	AND_GE_M2	×	○	
AND_EQ_M	AND_EQ_M2	○	○	
AND_GE_M	AND_GE_M2	○	○	
AND_GT_M	AND_GT_M2	○	○	
AND_LE_M	AND_LE_M2	○	○	
AND_LT_M	AND_LT_M2	○	○	
AND_NE_M	AND_NE_M2	○	○	

Instruction name in GX Works2	Compatible instruction name	Program type		Reference
		ST	FBD	
ANDD<	ANDD_LT_M2	×	○	Page 216 LDD□(_U), ANDD□(_U), ORD□(_U)
ANDD<=	ANDD_LE_M2	×	○	
ANDD<>	ANDD_NE_M2	×	○	
ANDD=	ANDD_EQ_M2	×	○	
ANDD>	ANDD_GT_M2	×	○	
ANDD>=	ANDD_GE_M2	×	○	
ANDD_EQ_M	ANDD_EQ_M2	○	○	
ANDD_GE_M	ANDD_GE_M2	○	○	
ANDD_GT_M	ANDD_GT_M2	○	○	
ANDD_LE_M	ANDD_LE_M2	○	○	
ANDD_LT_M	ANDD_LT_M2	○	○	
ANDD_NE_M	ANDD_NE_M2	○	○	
ASIN	ASIN_M2	○	○	Page 1451 ASIN(_E)
ASIN_MD		○	○	
ASIN_E_MD		○	○	
ASIND	ASIND_M2	○	○	
ASIND_MD		○	○	
ASIND_E_MD		○	○	
ATAN	ATAN_M2	○	○	Page 1453 ATAN(_E)
ATAN_MD		○	○	
ATAN_E_MD		○	○	
ATAND	ATAND_M2	○	○	
ATAND_MD		○	○	
ATAND_E_MD		○	○	
BAND	BAND_M2	○	○	Page 705 BAND(P)(_U)
BAND_MD		○	○	
BANDP	BANDP_M2	○	○	
BAND_P_MD		○	○	
BDSQR	BDSQR_M2	○	○	Page 982 BDSQRT(P)
BDSQRP	BDSQRP_M2	○	○	
BIDIN_M	G_BIDIN_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
BIDINP_M	GP_BIDIN_M2	○	○	
BIDOUT_M	G_BIDOUT_M2	○	○	
BIDOUTP_M	GP_BIDOUT_M2	○	○	
BINDA	BINDA_M2	○	○	Page 796 BINDA(P)(_U)
BINDA_MD		○	○	
BINDA_K_MD		○	○	
BINDA_S_MD		○	○	
BINDAP	BINDAP_M2	○	○	
BINDA_P_MD		○	○	
BINDA_P_S_MD		○	○	
BINDA_K_P_MD		○	○	
BK+	BKPLUS_M2	×	○	Page 277 BK+(P)(_U)
BK+P	BKPLUSP_M2	×	○	
BK-	BKMINUS_M2	×	○	Page 279 BK-(P)(_U)
BK-P	BKMINUSP_M2	×	○	

A

Instruction name in GX Works2	Compatible instruction name	Program type		Reference
		ST	FBD	
BKCMP<	BKCMP_LT_M2	×	○	Page 226 BKCMP□(P)(_U)
BKCMP<P	BKCMP_LTP_M2	×	○	
BKCMP<=	BKCMP_LE_M2	×	○	
BKCMP<=P	BKCMP_LEP_M2	×	○	
BKCMP<>	BKCMP_NE_M2	×	○	
BKCMP<>P	BKCMP_NEP_M2	×	○	
BKCMP=	BKCMP_EQ_M2	×	○	
BKCMP=P	BKCMP_EQP_M2	×	○	
BKCMP>	BKCMP_GT_M2	×	○	
BKCMP>P	BKCMP_GTP_M2	×	○	
BKCMP>=	BKCMP_GE_M2	×	○	
BKCMP>=P	BKCMP_GEP_M2	×	○	
BKCMP_EQ_M	BKCMP_EQ_M2	○	○	
BKCMP_EQP_M	BKCMP_EQP_M2	○	○	
BKCMP_GE_M	BKCMP_GE_M2	○	○	
BKCMP_GEP_M	BKCMP_GEP_M2	○	○	
BKCMP_GT_M	BKCMP_GT_M2	○	○	
BKCMP_GTP_M	BKCMP_GTP_M2	○	○	
BKCMP_LE_M	BKCMP_LE_M2	○	○	
BKCMP_LEP_M	BKCMP_LEP_M2	○	○	
BKCMP_LT_M	BKCMP_LT_M2	○	○	
BKCMP_LTP_M	BKCMP_LTP_M2	○	○	
BKCMP_NE_M	BKCMP_NE_M2	○	○	
BKCMP_NEP_M	BKCMP_NEP_M2	○	○	
BKMINUS_M	BKMINUS_M2	○	○	Page 279 BK-(P)(_U)
BKMINUSP_M	BKMINUSP_M2	○	○	
BKPLUS_M	BKPLUS_M2	○	○	Page 277 BK+(P)(_U)
BKPLUSP_M	BKPLUSP_M2	○	○	
BKRST	BKRST_M2	○	○	Page 343 BKRST(P)
BKRST_M		○	○	
BKRSTP	BKRSTP_M2	○	○	
BKRSTP_M		○	○	
BSQR	BSQR_M2	○	○	Page 980 BSQRT(P)
BSQRP	BSQRP_M2	○	○	
BUFRCV_M	ZP_BUFRCV_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
BUFSND_M	ZP_BUFSND_M2	○	○	
CLOSE_M	ZP_CLOSE_M2	○	○	
COS	COS_M2	○	○	Page 1449 COS(_E)
COS_MD		○	○	
COS_E_MD		○	○	
COSD	COSD_M2	○	○	
COSD_MD		○	○	
COSD_E_MD		○	○	
CSET_M	ZP_CSET_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
CSET_P_M		○	○	
D*	DMULTI_M2	×	○	Page 251 D*(P)(_U)
D*P	DMULTIP_M2	×	○	
D+	DPLUS_M2	×	○	Page 241 D+(P)(_U) [when three operands are set]
D+P	DPLUSP_M2	×	○	
D-	DMINUS_M2	×	○	Page 245 D-(P)(_U) [when three operands are set]
D-P	DMINUSP_M2	×	○	

Instruction name in GX Works2	Compatible instruction name	Program type		Reference
		ST	FBD	
D/	DDIVISION_M2	×	○	Page 253 D/(P)(_U)
D/P	DDIVISIONP_M2	×	○	
DABIN	DABIN_M2	○	○	Page 469 DABIN(P)(_U)
DABIN_MD		○	○	
DABIN_S_MD		○	○	
DABINP	DABINP_M2	○	○	
DABIN_P_MD		○	○	
DABIN_P_S_MD		○	○	
DB*	DBMULTI_M2	×	○	Page 273 DB*(P)
DB*P	DBMULTIP_M2	×	○	
DBAND	DBAND_M2	○	○	Page 707 DBAND(P)(_U)
DBAND_MD		○	○	
DBANDP	DBANDP_M2	○	○	
DBAND_P_MD		○	○	
DBINDA	DBINDA_M2	○	○	Page 800 DBINDA(P)(_U)
DBINDA_MD		○	○	
DBINDA_K_MD		○	○	
DBINDA_S_MD		○	○	
DBINDAP	DBINDAP_M2	○	○	
DBINDA_P_MD		○	○	
DBINDA_P_S_MD		○	○	
DBINDA_K_P_MD		○	○	
DBK+	DBKPLUS_M2	×	○	Page 281 DBK+(P)(_U)
DBK+P	DBKPLUSP_M2	×	○	
DBK-	DBKMINUS_M2	×	○	Page 284 DBK-(P)(_U)
DBK-P	DBKMINUSP_M2	×	○	
DBKCMP<	DBKCMP_LT_M2	×	○	Page 228 DBKCMP□(P)(_U)
DBKCMP<P	DBKCMP_LTP_M2	×	○	
DBKCMP<=	DBKCMP_LE_M2	×	○	
DBKCMP<=P	DBKCMP_LEP_M2	×	○	
DBKCMP<>	DBKCMP_NE_M2	×	○	
DBKCMP<>P	DBKCMP_NEP_M2	×	○	
DBKCMP=	DBKCMP_EQ_M2	×	○	
DBKCMP=P	DBKCMP_EQP_M2	×	○	
DBKCMP>	DBKCMP_GT_M2	×	○	
DBKCMP>P	DBKCMP_GTP_M2	×	○	
DBKCMP>=	DBKCMP_GE_M2	×	○	
DBKCMP>=P	DBKCMP_GEP_M2	×	○	
DBKCMP_EQ_M	DBKCMP_EQ_M2	○	○	
DBKCMP_EQP_M	DBKCMP_EQP_M2	○	○	
DBKCMP_GE_M	DBKCMP_GE_M2	○	○	
DBKCMP_GEP_M	DBKCMP_GEP_M2	○	○	
DBKCMP_GT_M	DBKCMP_GT_M2	○	○	
DBKCMP_GTP_M	DBKCMP_GTP_M2	○	○	
DBKCMP_LE_M	DBKCMP_LE_M2	○	○	
DBKCMP_LEP_M	DBKCMP_LEP_M2	○	○	
DBKCMP_LT_M	DBKCMP_LT_M2	○	○	
DBKCMP_LTP_M	DBKCMP_LTP_M2	○	○	
DBKCMP_NE_M	DBKCMP_NE_M2	○	○	
DBKCMP_NEP_M	DBKCMP_NEP_M2	○	○	
DBKMINUS_M	DBKMINUS_M2	○	○	Page 284 DBK-(P)(_U)
DBKMINUSP_M	DBKMINUSP_M2	○	○	

A

Instruction name in GX Works2	Compatible instruction name	Program type		Reference
		ST	FBD	
DBKPLUS_M	DBKPLUS_M2	○	○	Page 281 DBK+(P)(_U)
DBKPLUSP_M	DBKPLUSP_M2	○	○	
DBMULTI_M	DBMULTI_M2	○	○	Page 273 DB*(P)
DBMULTIP_M	DBMULTIP_M2	○	○	
DDABIN	DDABIN_M2	○	○	Page 472 DDABIN(P)(_U)
DDABIN_MD		○	○	
DDABIN_S_MD		○	○	
DDABINP	DDABINP_M2	○	○	
DDABIN_P_MD		○	○	
DDABIN_P_S_MD		○	○	
DDEC	DDEC_M2	○	○	Page 293 DDEC(P)(_U)
DDEC_M		○	○	
DDECP	DDECP_M2	○	○	
DDECP_M		○	○	
DDIVID_3_M	DDIVISION_M2	○	○	Page 253 D/(P)(_U)
DDIVIDP_3_M	DDIVISIONP_M2	○	○	
DEC	DEC_M2	○	○	Page 289 DEC(P)(_U)
DEC_M		○	○	
DECP	DECP_M2	○	○	
DECP_M		○	○	
DGBIN	DGBIN_M2	○	○	Page 463 DGBIN(P)(_U)
DGBIN_M		○	○	
DGBINP	DGBINP_M2	○	○	
DGBINP_M		○	○	
DGRY	DGRY_M2	○	○	Page 459 DGRY(P)(_U)
DGRY_M		○	○	
DGRYP	DGRYP_M2	○	○	
DGRYP_M		○	○	
DINC	DINC_M2	○	○	Page 291 DINC(P)(_U)
DINC_M		○	○	
DINCP	DINCP_M2	○	○	
DINCP_M		○	○	
DIVID_3_M	DIVISION_M2	○	○	Page 249 /(P)(_U)
DIVIDP_3_M	DIVISIONP_M2	○	○	
DLIMIT	DLIMIT_M2	○	○	Page 1485 LIMIT(_E)
DLIMIT_MD		○	○	
DLIMITP	DLIMITP_M2	○	○	Page 703 DLIMIT(P)(_U)
DLIMIT_P_MD		○	○	
DMAX	DMAX_M2	○	○	Page 1483 MAX(_E), MIN(_E)
DMAX_M		○	○	
DMAXP	DMAXP_M2	○	○	Page 741 DMAX(P)(_U)
DMAXP_M		○	○	
DMEAN	DMEAN_M2	○	○	Page 769 DMEAN(P)(_U)
DMEAN_M		○	○	
DMEANP	DMEANP_M2	○	○	
DMEANP_M		○	○	
DMIN	DMIN_M2	○	○	Page 1483 MAX(_E), MIN(_E)
DMIN_M		○	○	
DMINP	DMINP_M2	○	○	Page 745 DMIN(P)(_U)
DMINP_M		○	○	
DMINUS_M	DMINUS_2_M2	○	○	Page 243 D-(P)(_U) [when two operands are set]
DMINUS_3_M	DMINUS_M2	○	○	Page 245 D-(P)(_U) [when three operands are set]

Instruction name in GX Works2	Compatible instruction name	Program type		Reference
		ST	FBD	
DMINUSP_M	DMINUSP_2_M2	○	○	Page 243 D-(P)(_U) [when two operands are set]
DMINUSP_3_M	DMINUSP_M2	○	○	Page 245 D-(P)(_U) [when three operands are set]
DMULTI_3_M	DMULTI_M2	○	○	Page 251 D*(P)(_U)
DMULTIP_3_M	DMULTIP_M2	○	○	
DPLUS_M	DPLUS_2_M2	○	○	Page 239 D+(P)(_U) [when two operands are set]
DPLUS_3_M	DPLUS_M2	○	○	Page 241 D+(P)(_U) [when three operands are set]
DPLUSP_M	DPLUSP_2_M2	○	○	Page 239 D+(P)(_U) [when two operands are set]
DPLUSP_3_M	DPLUSP_M2	○	○	Page 241 D+(P)(_U) [when three operands are set]
DROL	DROL_M2	○	○	Page 623 DROL(P), DRCL(P)
DROL_M		○	○	
DROR	DROR_M2	○	○	Page 621 DROR(P), DRCR(P)
DROR_M		○	○	
DSCL	DSCL_M2	○	○	Page 716 DSCL(P)(_U)
DSCLP	DSCLP_M2	○	○	
DSCL2	DSCL2_M2	○	○	Page 721 DSCL2(P)(_U)
DSCL2P	DSCL2P_M2	○	○	
DSORT	DSORT_M2	○	○	Page 749 DSORTD(_U)
DSORT_M		○	○	
DSTR	DSTR_M2	○	○	Page 816 DSTR(P)(_U)
DSTR_MD		○	○	
DSTR_K_MD		○	○	
DSTR_S_MD		○	○	
DSTRP	DSTRP_M2	○	○	
DSTR_P_MD		○	○	
DSTR_P_S_MD		○	○	
DSTR_K_P_MD		○	○	
DTO	DTO_M2	○	○	Page 1141 TO(P), DTO(P)
DTO_M		○	○	
DTOP	DTOP_M2	○	○	
DTOP_M		○	○	
DVAL	DVAL_M2	○	○	Page 491 DVAL(P)(_U)
DVAL_MD		○	○	
DVAL_S_MD		○	○	
DVALP	DVALP_M2	○	○	
DVAL_P_MD		○	○	
DVAL_P_S_MD		○	○	
DWSUM	DWSUM_M2	○	○	Page 765 DWSUM(P)(_U)
DWSUM_M		○	○	
DWSUMP	DWSUMP_M2	○	○	
DWSUMP_M		○	○	
DZONE	DZONE_M2	○	○	Page 711 DZONE(P)(_U)
DZONE_MD		○	○	
DZONEP	DZONEP_M2	○	○	
DZONE_P_MD		○	○	
EREXP	EREXP_M2	○	○	Page 918 EREXP(P)
EREXP_M		○	○	
EREXPP	EREXPP_M2	○	○	
EREXPP_M		○	○	
ERRCLR_M	ZP_ERRCLR_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
ERRRD_M	ZP_ERRRD_M2	○	○	

A

Instruction name in GX Works2	Compatible instruction name	Program type		Reference
		ST	FBD	
EXP	EXP_M2	○	○	Page 1447 EXP(_E)
EXP_MD		○	○	
EXP_E_MD		○	○	
EXPD	EXPD_M2	○	○	
EXPD_MD		○	○	
EXPD_E_MD		○	○	
G_BIDIN	G_BIDIN_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
G_BIDOUT	G_BIDOUT_M2	○	○	
G_CCPASET	G_CCPASET_M2	○	○	
G_CPRTCL	G_CPRTCL_M2	○	○	
G_GETE	G_GETE_M2	○	○	
G_INPUT	G_INPUT_M2	○	○	
G_OGLOAD	G_OGLOAD_M2	○	○	
G_OGSTOR	G_OGSTOR_M2	○	○	
G_ONDEMAND	G_ONDEMAND_M2	○	○	
G_OUTPUT	G_OUTPUT_M2	○	○	
G_PRR	G_PRR_M2	○	○	
G_PUTE	G_PUTE_M2	○	○	
G_RDMSG	G_RDMSG_M2	○	○	
G_REQ	G_REQ_M2	○	○	
G_RIRCV	G_RIRCV_M2	○	○	
G_RIRD	G_RIRD_M2	○	○	
G_RISEND	G_RISEND_M2	○	○	
G_RITO	G_RITO_M2	○	○	
G_RIWT	G_RIWT_M2	○	○	
G_RLPASET	G_RLPASET_M2	○	○	
G_SPBUSY	G_SPBUSY_M2	○	○	
GBIN	GBIN_M2	○	○	Page 461 GBIN(P)(_U)
GBIN_M		○	○	
GBINP	GBINP_M2	○	○	
GBINP_M		○	○	

Instruction name in GX Works2	Compatible instruction name	Program type		Reference
		ST	FBD	
GETE_M	G_GETE_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
GETEP_M	GP_GETE_M2	○	○	
GP_BIDIN	GP_BIDIN_M2	○	○	
GP_BIDOUT	GP_BIDOUT_M2	○	○	
GP_CCPASET	GP_CCPASET_M2	○	○	
GP_CPRTCL	GP_CPRTCL_M2	○	○	
GP_ECPRTCL	GP_ECPRTCL_M2	○	○	
GP_GETE	GP_GETE_M2	○	○	
GP_OGLOAD	GP_OGLOAD_M2	○	○	
GP_OGSTOR	GP_OGSTOR_M2	○	○	
GP_ONDEMAND	GP_ONDEMAND_M2	○	○	
GP_OUTPUT	GP_OUTPUT_M2	○	○	
GP_PRR	GP_PRR_M2	○	○	
GP_PUTE	GP_PUTE_M2	○	○	
GP_RDMSG	GP_RDMSG_M2	○	○	
GP_RECV	GP_RECV_M2	○	○	
GP_REQ	GP_REQ_M2	○	○	
GP_RIRCV	GP_RIRCV_M2	○	○	
GP_RIRD	GP_RIRD_M2	○	○	
GP_RISEND	GP_RISEND_M2	○	○	
GP_RITO	GP_RITO_M2	○	○	
GP_RIWT	GP_RIWT_M2	○	○	
GP_RLPASET	GP_RLPASET_M2	○	○	
GP_SEND	GP_SEND_M2	○	○	
GP_SPBUSY	GP_SPBUSY_M2	○	○	
GRY	GRY_M2	○	○	Page 457 GRY(P)(_U)
GRY_M		○	○	
GRYP	GRYP_M2	○	○	
GRYP_M		○	○	
INC	INC_M2	○	○	Page 287 INC(P)(_U)
INC_M		○	○	
INCP	INCP_M2	○	○	
INCP_M		○	○	
INPUT_M	G_INPUT_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
LD<	LD_LT_M2	×	○	Page 214 LD□(_U), AND□(_U), OR□(_U)
LD<=	LD_LE_M2	×	○	
LD<>	LD_NE_M2	×	○	
LD=	LD_EQ_M2	×	○	
LD>	LD_GT_M2	×	○	
LD>=	LD_GE_M2	×	○	
LD_EQ_M	LD_EQ_M2	○	○	
LD_GE_M	LD_GE_M2	○	○	
LD_GT_M	LD_GT_M2	○	○	
LD_LE_M	LD_LE_M2	○	○	
LD_LT_M	LD_LT_M2	○	○	
LD_NE_M	LD_NE_M2	○	○	

A

Instruction name in GX Works2	Compatible instruction name	Program type		Reference
		ST	FBD	
LDD<	LDD_LT_M2	×	○	Page 216 LDD□(_U), ANDD□(_U), ORD□(_U)
LDD<=	LDD_LE_M2	×	○	
LDD<>	LDD_NE_M2	×	○	
LDD=	LDD_EQ_M2	×	○	
LDD>	LDD_GT_M2	×	○	
LDD>=	LDD_GE_M2	×	○	
LDD_EQ_M	LDD_EQ_M2	○	○	
LDD_GE_M	LDD_GE_M2	○	○	
LDD_GT_M	LDD_GT_M2	○	○	
LDD_LE_M	LDD_LE_M2	○	○	
LDD_LT_M	LDD_LT_M2	○	○	
LDD_NE_M	LDD_NE_M2	○	○	
LEFT	LEFT_M2	○	○	Page 1496 LEFT(_E), RIGHT(_E)
LEFT_M		○	○	
LEN	LEN_M2	○	○	Page 1494 LEN(_E)
LEN_MD		○	○	
LEN_S		○	○	
LEN_S_MD		○	○	
LIMIT	LIMIT_M2	○	○	Page 701 LIMIT(P)(_U)
LIMIT_MD		○	○	
LIMITP	LIMITP_M2	○	○	
LIMIT_P_MD		○	○	
LOG	LOG_M2	○	○	Page 1445 LOG(_E)
LOG_MD		○	○	
LOG_E_MD		○	○	
LOGD	LOGD_M2	○	○	
LOGD_MD		○	○	
LOGD_E_MD		○	○	
MAX	MAX_M2	○	○	Page 1483 MAX(_E), MIN(_E)
MAX_M		○	○	
MAXP	MAXP_M2	○	○	Page 739 MAX(P)(_U)
MAXP_M		○	○	Page 741 DMAX(P)(_U)
MEAN	MEAN_M2	○	○	Page 767 MEAN(P)(_U)
MEAN_M		○	○	
MEANP	MEANP_M2	○	○	
MEANP_M		○	○	
MIN	MIN_M2	○	○	Page 1483 MAX(_E), MIN(_E)
MIN_M		○	○	
MINP	MINP_M2	○	○	Page 743 MIN(P)(_U)
MINP_M		○	○	
MINUS_M	MINUS_2_M2	○	○	Page 235 -(P)(_U) [when two operands are set]
MINUS_3_M	MINUS_M2	○	○	Page 237 -(P)(_U) [when three operands are set]
MINUSP_M	MINUSP_2_M2	○	○	Page 235 -(P)(_U) [when two operands are set]
MINUSP_3_M	MINUSP_M2	○	○	Page 237 -(P)(_U) [when three operands are set]
MULTI_3_M	MULTI_M2	○	○	Page 247 *(P)(_U)
MULTIP_3_M	MULTIP_M2	○	○	

Instruction name in GX Works2	Compatible instruction name	Program type		Reference
		ST	FBD	
OGLOAD_U_M	G_OGLOAD_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
OGLOAD_UP_M	GP_OGLOAD_M2	○	○	
OGSTOR_U_M	G_OGSTOR_M2	○	○	
OGSTOR_UP_M	GP_OGSTOR_M2	○	○	
ONDEMAND_M	G_ONDEMAND_M2	○	○	
ONDEMANDP_M	GP_ONDEMAND_M2	○	○	
OPEN_M	ZP_OPEN_M2	○	○	
OR<	OR_LT_M2	×	○	Page 214 LD□(_U), AND□(_U), OR□(_U)
OR<=	OR_LE_M2	×	○	
OR<>	OR_NE_M2	×	○	
OR=	OR_EQ_M2	×	○	
OR>	OR_GT_M2	×	○	
OR>=	OR_GE_M2	×	○	
OR_EQ_M	OR_EQ_M2	○	○	
OR_GE_M	OR_GE_M2	○	○	
OR_GT_M	OR_GT_M2	○	○	
OR_LE_M	OR_LE_M2	○	○	
OR_LT_M	OR_LT_M2	○	○	
OR_NE_M	OR_NE_M2	○	○	
ORD<	ORD_LT_M2	×	○	Page 216 LDD□(_U), ANDD□(_U), ORD□(_U)
ORD<=	ORD_LE_M2	×	○	
ORD<>	ORD_NE_M2	×	○	
ORD=	ORD_EQ_M2	×	○	
ORD>	ORD_GT_M2	×	○	
ORD>=	ORD_GE_M2	×	○	
ORD_EQ_M	ORD_EQ_M2	○	○	
ORD_GE_M	ORD_GE_M2	○	○	
ORD_GT_M	ORD_GT_M2	○	○	
ORD_LE_M	ORD_LE_M2	○	○	
ORD_LT_M	ORD_LT_M2	○	○	
ORD_NE_M	ORD_NE_M2	○	○	
OUTPUT_M	G_OUTPUT_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
OUTPUTP_M	GP_OUTPUT_M2	○	○	
PFWRT_P_M	ZP_PFWRT_M2	○	○	
PINIT_M	ZP_PINIT_M2	○	○	
PINIT_P_M		○	○	
PLUS_M	PLUS_2_M2	○	○	Page 231 +(P)(_U) [when two operands are set]
PLUS_3_M	PLUS_M2	○	○	Page 233 +(P)(_U) [when three operands are set]
PLUSP_M	PLUSP_2_M2	○	○	Page 231 +(P)(_U) [when two operands are set]
PLUSP_3_M	PLUSP_M2	○	○	Page 233 +(P)(_U) [when three operands are set]

A

Instruction name in GX Works2	Compatible instruction name	Program type		Reference
		ST	FBD	
PRR_M	G_PRR_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
PRRP_M	GP_PRR_M2	○	○	
PSTRT1_P_M	ZP_PSTRT1_M2	○	○	
PSTRT2_P_M	ZP_PSTRT2_M2	○	○	
PSTRT3_P_M	ZP_PSTRT3_M2	○	○	
PSTRT4_P_M	ZP_PSTRT4_M2	○	○	
PUTE_M	G_PUTE_M2	○	○	
PUTEP_M	GP_PUTE_M2	○	○	
RDMSG_MD	G_RDMSG_M2	○	○	
RDMSG_P_MD	GP_RDMSG_M2	○	○	
RECV_UP_M	GP_RECV_M2	○	○	
RECVP_M		○	○	
RECVS_U_M	Z_RECVS_M2	○	○	
REMTO_P_MD	ZP_REMTO_M2	○	○	
REQ_M	G_REQ_M2	○	○	
REQ_U_M		○	○	
REQ_UP_M	GP_REQ_M2	○	○	
REQP_M		○	○	
RIGHT	RIGHT_M2	○	○	Page 1496 LEFT(_E), RIGHT(_E)
RIGHT_M		○	○	
RIRCV_MD	G_RIRCV_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
RIRCV_P_MD	GP_RIRCV_M2	○	○	
RIRD_MD	G_RIRD_M2	○	○	
RIRD_P_MD	GP_RIRD_M2	○	○	
RISEND_MD	G_RISEND_M2	○	○	
RISEND_P_MD	GP_RISEND_M2	○	○	
RITO_MD	G_RITO_M2	○	○	
RITO_P_MD	GP_RITO_M2	○	○	
RIWT_MD	G_RIWT_M2	○	○	
RIWT_P_MD	GP_RIWT_M2	○	○	
RLPASET_MD	G_RLPASET_M2	○	○	
RLPASET_P_MD	GP_RLPASET_M2	○	○	
ROL	ROL_M2	○	○	Page 618 ROL(P), RCL(P)
ROL_M		○	○	
ROR	ROR_M2	○	○	Page 615 ROR(P), RCR(P)
ROR_M		○	○	
SCL	SCL_M2	○	○	Page 713 SCL(P)(_U)
SCLP	SCLP_M2	○	○	
SCL2	SCL2_M2	○	○	Page 719 SCL2(P)(_U)
SCL2P	SCL2P_M2	○	○	
SEND_4_P_M	GP_SEND_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
SEND_UP_M		○	○	
SIN	SIN_M2	○	○	Page 1448 SIN(_E)
SIN_MD		○	○	
SIN_E_MD		○	○	
SIND	SIND_M2	○	○	
SIND_MD		○	○	
SIND_E_MD		○	○	
SORT	SORT_M2	○	○	Page 747 SORTD(_U)
SORT_M		○	○	
SPBUSY_MD	G_SPBUSY_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
SPBUSY_P_MD	GP_SPBUSY_M2	○	○	

Instruction name in GX Works2	Compatible instruction name	Program type		Reference
		ST	FBD	
STR	STR_M2	○	○	Page 813 STR(P)(_U)
STR_MD		○	○	
STR_K_MD		○	○	
STR_S_MD		○	○	
STRP	STRP_M2	○	○	
STR_P_MD		○	○	
STR_P_S_MD		○	○	
STR_K_P_MD		○	○	
SWAP	SWAP_M2	×	○	Page 553 SWAP(P)
SWAP_MD		×	○	
SWAP_P_MD	SWAPP_M2	×	○	
SWAPP		×	○	
TAN	TAN_M2	○	○	Page 1450 TAN(_E)
TAN_MD		○	○	
TAN_E_MD		○	○	
TAND	TAND_M2	○	○	
TAND_MD		○	○	
TAND_E_MD		○	○	
TEACH1_P_M	ZP_TEACH1_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
TEACH2_P_M	ZP_TEACH2_M2	○	○	
TEACH3_P_M	ZP_TEACH3_M2	○	○	
TEACH4_P_M	ZP_TEACH4_M2	○	○	
TO	TO_M2	○	○	Page 1141 TO(P), DTO(P)
TO_M		○	○	
TOP	TOP_M2	○	○	
TOP_M		○	○	
UINI_M	ZP_UINI_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
UINI_U_M	Z_UINI_M2	○	○	
UINI_UP_M	ZP_UINI_M2	○	○	
VAL	VAL_M2	○	○	Page 488 VAL(P)(_U)
VAL_MD		○	○	
VAL_S_MD		○	○	
VALP	VALP_M2	○	○	
VAL_P_MD		○	○	
VAL_P_S_MD		○	○	
WSUM	WSUM_M2	○	○	Page 763 WSUM(P)(_U)
WSUM_M		○	○	
WSUMP	WSUMP_M2	○	○	
WSUMP_M		○	○	
Z_ABRST1	Z_ABRST1_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
Z_ABRST2	Z_ABRST2_M2	○	○	
Z_ABRST3	Z_ABRST3_M2	○	○	
Z_ABRST4	Z_ABRST4_M2	○	○	
Z_RECVS	Z_RECVS_M2	○	○	
Z_UINI	Z_UINI_M2	○	○	
ZONE	ZONE_M2	○	○	Page 709 ZONE(P)(_U)
ZONE_MD		○	○	
ZONEP	ZONEP_M2	○	○	
ZONE_P_MD		○	○	

A

Instruction name in GX Works2	Compatible instruction name	Program type		Reference
		ST	FBD	
ZP_BUFRCV	ZP_BUFRCV_M2	○	○	MELSEC iQ-R Programming Manual (Module Dedicated Instructions)
ZP_BUFSND	ZP_BUFSND_M2	○	○	
ZP_CLOSE	ZP_CLOSE_M2	○	○	
ZP_CSET	ZP_CSET_M2	○	○	
ZP_ERRCLR	ZP_ERRCLR_M2	○	○	
ZP_ERRRD	ZP_ERRRD_M2	○	○	
ZP_OPEN	ZP_OPEN_M2	○	○	
ZP_PFWRT	ZP_PFWRT_M2	○	○	
ZP_PINIT	ZP_PINIT_M2	○	○	
ZP_PSTRT1	ZP_PSTRT1_M2	○	○	
ZP_PSTRT2	ZP_PSTRT2_M2	○	○	
ZP_PSTRT3	ZP_PSTRT3_M2	○	○	
ZP_PSTRT4	ZP_PSTRT4_M2	○	○	
ZP_REMTO	ZP_REMTO_M2	○	○	
ZP_TEACH1	ZP_TEACH1_M2	○	○	
ZP_TEACH2	ZP_TEACH2_M2	○	○	
ZP_TEACH3	ZP_TEACH3_M2	○	○	
ZP_TEACH4	ZP_TEACH4_M2	○	○	
ZP_UINI	ZP_UINI_M2	○	○	

Compatible functions (standard functions)

Standard functions used in programs using ST (including the inline ST) or structured ladder/FBD are replaced by the compatible instructions.

Function name in GX Works2	Compatible function name	Program type		Reference
		ST	FBD	
ABS_E	ABS_E_M2	○	×	Page 1441 ABS(_E)
ACOS_E	ACOS_E_M2	○	×	Page 1452 ACOS(_E)
ADD_E	ADD_E_M2	○	×	Page 1454 ADD(_E)
ADD_TIME_E	ADD_TIME_E_M2	○	×	Page 1511 ADD_TIME(_E)
AND_E	AND_E_M2	○	×	Page 1477 AND(_E), OR(_E), XOR(_E)
ASIN_E	ASIN_E_M2	○	×	Page 1451 ASIN(_E)
ATAN_E	ATAN_E_M2	○	×	Page 1453 ATAN(_E)
BCD_TO_DINT_E	BCD_TO_DINT_E_M2	○	×	Page 1388 BCD_TO_DINT(_E)
BCD_TO_INT_E	BCD_TO_INT_E_M2	○	×	Page 1386 BCD_TO_INT(_E)
BCD_TO_STR_E	BCD_TO_STR_E_M2	○	×	Page 1391 BCD_TO_STRING(_E)
BITARR_TO_DINT_E	BITARR_TO_DINT_E_M2	○	×	Page 1430 BITARR_TO_DINT(_E)
BITARR_TO_INT_E	BITARR_TO_INT_E_M2	○	×	Page 1429 BITARR_TO_INT(_E)
BOOL_TO_DINT_E	BOOL_TO_DINT_E_M2	○	×	Page 1344 BOOL_TO_DINT(_E)
BOOL_TO_DWORD_E	BOOL_TO_DWORD_E_M2	○	×	Page 1342 BOOL_TO_DWORD(_E)
BOOL_TO_INT_E	BOOL_TO_INT_E_M2	○	×	Page 1343 BOOL_TO_INT(_E)
BOOL_TO_STR_E	BOOL_TO_STR_E_M2	○	×	Page 1346 BOOL_TO_STRING(_E)
BOOL_TO_TIME_E	BOOL_TO_TIME_E_M2	○	×	Page 1345 BOOL_TO_TIME(_E)
BOOL_TO_WORD_E	BOOL_TO_WORD_E_M2	○	×	Page 1340 BOOL_TO_WORD(_E)
CONCAT_E	CONCAT_E_M2	○	×	Page 1500 CONCAT(_E)
COS_E	COS_E_M2	○	×	Page 1449 COS(_E)
CPY_BITARR_E	CPY_BITARR_E_M2	○	×	Page 1433 CPY_BITARR(_E)
CPY_BIT_OF_INT_E	CPY_BIT_OF_INT_E_M2	○	×	Page 1438 CPY_BIT_OF_INT(_E)
DELETE_E	DELETE_E_M2	○	×	Page 1504 DELETE(_E)
DINT_TO_BCD_E	DINT_TO_BCD_E_M2	○	×	Page 1379 DINT_TO_BCD(_E)
DINT_TO_BITARR_E	DINT_TO_BITARR_E_M2	○	×	Page 1432 DINT_TO_BITARR(_E)
DINT_TO_BOOL_E	DINT_TO_BOOL_E_M2	○	×	Page 1374 DINT_TO_BOOL(_E)
DINT_TO_DWORD_E	DINT_TO_DWORD_E_M2	○	×	Page 1377 DINT_TO_DWORD(_E)
DINT_TO_INT_E	DINT_TO_INT_E_M2	○	×	Page 1378 DINT_TO_INT(_E)
DINT_TO_LREAL_E	DINT_TO_LREAL_E_M2	○	×	Page 1382 DINT_TO_LREAL(_E)
DINT_TO_REAL_E	DINT_TO_REAL_E_M2	○	×	Page 1381 DINT_TO_REAL(_E)
DINT_TO_STR_E	DINT_TO_STR_E_M2	○	×	Page 1384 DINT_TO_STRING(_E)
DINT_TO_TIME_E	DINT_TO_TIME_E_M2	○	×	Page 1383 DINT_TO_TIME(_E)
DINT_TO_WORD_E	DINT_TO_WORD_E_M2	○	×	Page 1375 DINT_TO_WORD(_E)
DIV_E	DIV_E_M2	○	×	Page 1462 DIV(_E)
DIV_TIME_E	DIV_TIME_E_M2	○	×	Page 1517 DIV_TIME(_E)
DWORD_TO_BOOL_E	DWORD_TO_BOOL_E_M2	○	×	Page 1354 DWORD_TO_BOOL(_E)
DWORD_TO_DINT_E	DWORD_TO_DINT_E_M2	○	×	Page 1359 DWORD_TO_DINT(_E)
DWORD_TO_INT_E	DWORD_TO_INT_E_M2	○	×	Page 1357 DWORD_TO_INT(_E)
DWORD_TO_STR_E	DWORD_TO_STR_E_M2	○	×	Page 1361 DWORD_TO_STRING(_E)
DWORD_TO_TIME_E	DWORD_TO_TIME_E_M2	○	×	Page 1360 DWORD_TO_TIME(_E)
DWORD_TO_WORD_E	DWORD_TO_WORD_E_M2	○	×	Page 1355 DWORD_TO_WORD(_E)
EQ_E	EQ_E_M2	○	×	Page 1490 GT(_E), GE(_E), EQ(_E), LE(_E), LT(_E)
EXP_E	EXP_E_M2	○	×	Page 1447 EXP(_E)
EXPT_E	EXPT_E_M2	○	×	Page 1466 EXPT(_E)
FIND_E	FIND_E_M2	○	×	Page 1509 FIND(_E)
GE_E	GE_E_M2	○	×	Page 1490 GT(_E), GE(_E), EQ(_E), LE(_E), LT(_E)
GET_BIT_OF_INT_E	GET_BIT_OF_INT_E_M2	○	×	Page 1434 GET_BIT_OF_INT(_E)
GT_E	GT_E_M2	○	×	Page 1490 GT(_E), GE(_E), EQ(_E), LE(_E), LT(_E)

A

Function name in GX Works2	Compatible function name	Program type		Reference
		ST	FBD	
INSERT_E	INSERT_E_M2	○	×	Page 1502 INSERT(_E)
INT_TO_BCD_E	INT_TO_BCD_E_M2	○	×	Page 1367 INT_TO_BCD(_E)
INT_TO_BITARR_E	INT_TO_BITARR_E_M2	○	×	Page 1431 INT_TO_BITARR(_E)
INT_TO_BOOL_E	INT_TO_BOOL_E_M2	○	×	Page 1362 INT_TO_BOOL(_E)
INT_TO_DINT_E	INT_TO_DINT_E_M2	○	×	Page 1366 INT_TO_DINT(_E)
INT_TO_DWORD_E	INT_TO_DWORD_E_M2	○	×	Page 1364 INT_TO_DWORD(_E)
INT_TO_LREAL_E	INT_TO_LREAL_E_M2	○	×	Page 1370 INT_TO_LREAL(_E)
INT_TO_REAL_E	INT_TO_REAL_E_M2	○	×	Page 1369 INT_TO_REAL(_E)
INT_TO_STR_E	INT_TO_STR_E_M2	○	×	Page 1372 INT_TO_STRING(_E)
INT_TO_TIME_E	INT_TO_TIME_E_M2	○	×	Page 1371 INT_TO_TIME(_E)
INT_TO_WORD_E	INT_TO_WORD_E_M2	○	×	Page 1363 INT_TO_WORD(_E)
LE_E	LE_E_M2	○	×	Page 1490 GT(_E), GE(_E), EQ(_E), LE(_E), LT(_E)
LEFT_E	LEFT_E_M2	○	×	Page 1496 LEFT(_E), RIGHT(_E)
LEN_E	LEN_E_M2	○	×	Page 1494 LEN(_E)
LIMITATION_E	LIMITATION_E_M2	○	×	Page 1485 LIMIT(_E)
LN_E	LN_E_M2	○	×	Page 1444 LN(_E)
LOG_E	LOG_E_M2	○	×	Page 1445 LOG(_E)
LREAL_TO_DINT_E	LREAL_TO_DINT_E_M2	○	×	Page 1404 LREAL_TO_DINT(_E)
LREAL_TO_INT_E	LREAL_TO_INT_E_M2	○	×	Page 1402 LREAL_TO_INT(_E)
LREAL_TO_REAL_E	LREAL_TO_REAL_E_M2	○	×	Page 1406 LREAL_TO_REAL(_E)
LT_E	LT_E_M2	○	×	Page 1490 GT(_E), GE(_E), EQ(_E), LE(_E), LT(_E)
M_REAL_TO_REAL_E	M_REAL_TO_REAL_E_M2	○	×	—
MAXIMUM_E	MAXIMUM_E_M2	○	×	Page 1483 MAX(_E), MIN(_E)
MID_E	MID_E_M2	○	×	Page 1498 MID(_E)
MINIMUM_E	MINIMUM_E_M2	○	×	Page 1483 MAX(_E), MIN(_E)
MOD_E	MOD_E_M2	○	×	Page 1464 MOD(_E)
MOVE_E	MOVE_E_M2	○	×	Page 1467 MOVE(_E)
MUL_E	MUL_E_M2	○	×	Page 1457 MUL(_E)
MUL_TIME_E	MUL_TIME_E_M2	○	×	Page 1515 MUL_TIME(_E)
MUX_E	MUX_E_M2	○	×	Page 1488 MUX(_E)
NE_E	NE_E_M2	○	×	Page 1492 NE(_E)
NOT_E	NOT_E_M2	○	×	Page 1480 NOT(_E)
OR_E	OR_E_M2	○	×	Page 1477 AND(_E), OR(_E), XOR(_E)
REAL_TO_DINT_E	REAL_TO_DINT_E_M2	○	×	Page 1395 REAL_TO_DINT(_E)
REAL_TO_INT_E	REAL_TO_INT_E_M2	○	×	Page 1393 REAL_TO_INT(_E)
REAL_TO_LREAL_E	REAL_TO_LREAL_E_M2	○	×	Page 1397 REAL_TO_LREAL(_E)
REAL_TO_M_REAL_E	REAL_TO_M_REAL_E_M2	○	×	—
REAL_TO_STR_E	REAL_TO_STR_E_M2	○	×	Page 1399 REAL_TO_STRING(_E)
REPLACE_E	REPLACE_E_M2	○	×	Page 1506 REPLACE(_E)
RIGHT_E	RIGHT_E_M2	○	×	Page 1496 LEFT(_E), RIGHT(_E)
ROL_E	ROL_E_M2	○	×	Page 1473 ROL(_E)
ROR_E	ROR_E_M2	○	×	Page 1475 ROR(_E)
SEL_E	SEL_E_M2	○	×	Page 1481 SEL(_E)
SET_BIT_OF_INT_E	SET_BIT_OF_INT_E_M2	○	×	Page 1436 SET_BIT_OF_INT(_E)
SHL_E	SHL_E_M2	○	×	Page 1469 SHL(_E)
SHR_E	SHR_E_M2	○	×	Page 1471 SHR(_E)
SIN_E	SIN_E_M2	○	×	Page 1448 SIN(_E)
SQRT_E	SQRT_E_M2	○	×	Page 1443 SQRT(_E)
STR_TO_BCD_E	STR_TO_BCD_E_M2	○	×	Page 1422 STRING_TO_BCD(_E)
STR_TO_BOOL_E	STR_TO_BOOL_E_M2	○	×	Page 1415 STRING_TO_BOOL(_E)
STR_TO_DINT_E	STR_TO_DINT_E_M2	○	×	Page 1420 STRING_TO_DINT(_E)
STR_TO_DWORD_E	STR_TO_DWORD_E_M2	○	×	Page 1417 STRING_TO_DWORD(_E)

Function name in GX Works2	Compatible function name	Program type		Reference
		ST	FBD	
STR_TO_INT_E	STR_TO_INT_E_M2	○	×	Page 1418 STRING_TO_INT(_E)
STR_TO_REAL_E	STR_TO_REAL_E_M2	○	×	Page 1424 STRING_TO_REAL(_E)
STR_TO_TIME_E	STR_TO_TIME_E_M2	○	×	Page 1427 STRING_TO_TIME(_E)
STR_TO_WORD_E	STR_TO_WORD_E_M2	○	×	Page 1416 STRING_TO_WORD(_E)
SUB_E	SUB_E_M2	○	×	Page 1459 SUB(_E)
SUB_TIME_E	SUB_TIME_E_M2	○	×	Page 1513 SUB_TIME(_E)
TAN_E	TAN_E_M2	○	×	Page 1450 TAN(_E)
TIME_TO_BOOL_E	TIME_TO_BOOL_E_M2	○	×	Page 1408 TIME_TO_BOOL(_E)
TIME_TO_DINT_E	TIME_TO_DINT_E_M2	○	×	Page 1412 TIME_TO_DINT(_E)
TIME_TO_DWORD_E	TIME_TO_DWORD_E_M2	○	×	Page 1410 TIME_TO_DWORD(_E)
TIME_TO_INT_E	TIME_TO_INT_E_M2	○	×	Page 1411 TIME_TO_INT(_E)
TIME_TO_STR_E	TIME_TO_STR_E_M2	○	×	Page 1413 TIME_TO_STRING(_E)
TIME_TO_WORD_E	TIME_TO_WORD_E_M2	○	×	Page 1409 TIME_TO_WORD(_E)
WORD_TO_BOOL_E	WORD_TO_BOOL_E_M2	○	×	Page 1347 WORD_TO_BOOL(_E)
WORD_TO_DINT_E	WORD_TO_DINT_E_M2	○	×	Page 1350 WORD_TO_DINT(_E)
WORD_TO_DWORD_E	WORD_TO_DWORD_E_M2	○	×	Page 1348 WORD_TO_DWORD(_E)
WORD_TO_INT_E	WORD_TO_INT_E_M2	○	×	Page 1349 WORD_TO_INT(_E)
WORD_TO_STR_E	WORD_TO_STR_E_M2	○	×	Page 1353 WORD_TO_STRING(_E)
WORD_TO_TIME_E	WORD_TO_TIME_E_M2	○	×	Page 1352 WORD_TO_TIME(_E)
XOR_E	XOR_E_M2	○	×	Page 1477 AND(_E), OR(_E), XOR(_E)

Compatible function blocks (standard function blocks)

Standard function blocks used in programs using ST (including the inline ST) or structured ladder/FBD are replaced by the compatible function blocks.

Function block name in GX Works2	Compatible function block name	Program type		Reference
		ST	FBD	
TOF_HIGH	TOF_HIGH_M2*1	○	○	Page 1543 TOF(_E)
TOF_HIGH_E	TOF_HIGH_E_M2*1	○	○	
TON_HIGH	TON_HIGH_M2*1	○	○	Page 1540 TON(_E)
TON_HIGH_E	TON_HIGH_E_M2*1	○	○	
TP_HIGH	TP_HIGH_M2*1	○	○	Page 1537 TP(_E)
TP_HIGH_E	TP_HIGH_E_M2*1	○	○	
CTD	CTD_M2*2	○	○	Page 1530 CTD(_E)
CTD_E	CTD_E_M2*2	○	○	
CTU	CTU_M2*2	○	○	Page 1528 CTU(_E)
CTU_E	CTU_E_M2*2	○	○	
CTUD	CTUD_M2*2	○	○	Page 1532 CTUD(_E)
CTUD_E	CTUD_E_M2*2	○	○	
F_TRIG	F_TRIG_M2*2	○	○	Page 1526 F_TRIG(_E)
F_TRIG_E	F_TRIG_E_M2*2	○	○	
RS	RS_M2*2	○	○	Page 1522 RS(_E)
RS_E	RS_E_M2*2	○	○	
R_TRIG	R_TRIG_M2*2	○	○	Page 1524 R_TRIG(_E)
R_TRIG_E	R_TRIG_E_M2*2	○	○	
SR	SR_M2*2	○	○	Page 1520 SR(_E)
SR_E	SR_E_M2*2	○	○	

*1 A part of operation of the compatible function block is different from that of the function block in GX Works2.
*2 When the function block is used in a GX Works3 project, only the input label is automatically replaced.

■Operation difference

Compatible function block	Description
TOF_HIGH_M2 TOF_LOW_M2	The valid range of the OFF delay timer time*1 varies depending on the long timer setting. Since the data type of the OFF delay timer time is TIME data type (32-bit value), the maximum value will fall within the range of TIME data type. ■Minimum value • Identical to the long timer setting value (ms). Note that if the long timer setting value is smaller than 1ms, the minimum value will be 1ms. ■Maximum value • OFF delay timer time (ms) ≤ 2147483647 (ms) × Long timer setting value (ms) If the OFF delay timer time exceeds the valid range, an operation error occurs. (Error code: 3401H) If an operation error occurs, correct the long timer setting value.
TON_HIGH_M2 TON_LOW_M2	The valid range of the ON delay timer time*1 varies depending on the long timer setting. Since the data type of the ON delay timer time is TIME data type (32-bit value), the maximum value will fall within the range of TIME data type. ■Minimum value • Identical to the long timer setting value (ms). Note that if the long timer setting value is smaller than 1ms, the minimum value will be 1ms. ■Maximum value • ON delay timer time (ms) ≤ 2147483647 (ms) × Long timer setting value (ms) If the ON delay timer time exceeds the valid range, an operation error occurs. (Error code: 3401H) If an operation error occurs, correct the long timer setting value.
TP_HIGH_M2 TP_LOW_M2	The valid range of the pulse width time*1 varies depending on the long timer setting. Since the data type of the pulse width time is TIME data type (32-bit value), the maximum value will fall within the range of TIME data type. ■Minimum value • Identical to the long timer setting value (ms). Note that if the long timer setting value is smaller than 1ms, the minimum value will be 1ms. ■Maximum value • Pulse width time (ms) ≤ 2147483647 (ms) × Long timer setting value (ms) If the pulse width time exceeds the valid range, an operation error occurs. (Error code: 3401H) If an operation error occurs, correct the long timer setting value.

*1 For the OFF delay timer time, ON delay timer time, and pulse width time, refer to the following.
📖 PX Developer Version 1 Programming Manual

Replacement of a PX Developer format project

When a PX Developer format project is used in GX Works3, some of the instructions are automatically replaced by the compatible function blocks.
For details, refer to the following.
📖 MELSEC iQ-R Programming Manual (Process Control Function Blocks/Instructions)

MEMO

INDEX

0 to 9

16-bit data (word data) 40
32-bit data (double word data) 43

A

Application Instructions 561

B

Backup mode 22
Basic instruction 214
Bit data 38
Buffer memory 22
Built-in Ethernet function instruction 1174

C

CC-Link IE Controller Network-equipped module . 24
CC-Link IE Field Network-equipped master/local module 24
Control system 22

D

Destination (d) 32
Double-precision real number data 47

E

Ethernet interface module with built-in CC-Link IE 24
Execution condition 51
External device 24

I

Instruction configuration 32
Instruction processing time 1549

L

Label 22

M

Manual page organization 26
Multiple CPU dedicated instruction 1281

N

Numerical value (n) 33

O

Operand 24

P

PID control instruction 1249
Process CPU 24
Process CPU (process mode) 22
Process CPU (redundant mode) 22
Programmable controller CPU 24

R

Real number data (floating-point data) 46
Recording function 1172
Redundant system 22
Redundant system instruction 1320
Redundant system with redundant extension base unit 1162
Remote head module 24
Request message 22
Response message 22
RnCPU 24
RnENCPU 24
RnENCPU (network part) 24
RnPCPU 24

S

Safety communications 22
Safety control 22
Safety CPU 22
Safety cycle processing 22
Safety device 22
Safety function module 22
Safety label 22,25
Safety program 22
Separate mode 22
Sequence instruction 150
SFC program instruction 1291
SIL2 function module 22
SIL2 Process CPU 22
Single-precision real number data 46
SLMP 24
SLMP-compatible device 24
Source (s) 32
Standard communications 23
Standard CPU 25
Standard device 23
Standard program 23
Standby system 22
String data 49
Subset processing 52
System A 22
System B 22

T

Types of PID instructions 1234

I

MEMO

INSTRUCTION INDEX

Symbols

-(P)(_U) 235,237
*(P)(_U) 247
/(P)(_U) 249
+(P)(_U). 231,233
$+(P) 788,790
$MOV(P) 792
$MOV(P)_WS. 794

A

ABS(_E) 1441
ABS_E_M2 1645
ACOS(_E) 1452
ACOS(P) 932
ACOSD(P) 944
ACOSD_M2 1632
ACOS_E_M2 1645
ACOS_M2 1632
ADD(_E) 1454
ADD_E_M2 1645
ADD_TIME(_E). 1511
ADD_TIME_E_M2 1645
ADRSET(P) 1018
ANB 159
AND 150,1291,1294
AND(_E) 1477
AND<(_U) 214
AND<=(_U) 214
AND<>(_U) 214
AND=(_U) 214
AND>(_U) 214
AND>=(_U) 214
AND$<. 785
AND$<=. 785
AND$<>. 785
AND$=. 785
AND$>. 785
AND$>=. 785
ANDD<(_U) 216
ANDD<=(_U) 216
ANDD<>(_U) 216
ANDD=(_U) 216
ANDD>(_U) 216
ANDD>=(_U) 216
ANDD_EQ(_U) 216
ANDD_EQ_M2 1633
ANDD_GE(_U) 216
ANDD_GE_M2 1633
ANDD_GT(_U) 216
ANDD_GT_M2 1633
ANDD_LE(_U) 216
ANDD_LE_M2 1633
ANDD_LT(_U) 216
ANDD_LT_M2 1633
ANDD_NE(_U) 216
ANDD_NE_M2 1633
ANDDT< 1106
ANDDT<= 1106
ANDDT<> 1106
ANDDT= 1106
ANDDT> 1106
ANDDT>= 1106
ANDDT_EQ 1106
ANDDT_GE 1106
ANDDT_GT 1106
ANDDT_LE 1106
ANDDT_LT 1106
ANDDT_NE 1106
ANDE< 857
ANDE<= 857
ANDE<> 857
ANDE= 857
ANDE> 857
ANDE>= 857
ANDED< 859
ANDED<= 859
ANDED<> 859
ANDED= 859
ANDED> 859
ANDED>= 859
ANDED_EQ. 859
ANDED_GE. 859
ANDED_GT 859
ANDED_LE 859
ANDED_LT 859
ANDED_NE 859
ANDE_EQ 857
ANDE_GE 857
ANDE_GT 857
ANDE_LE 857
ANDE_LT 857
AND_E_M2 1645
ANDE_NE 857
AND_EQ(_U) 214
AND_EQ_M2 1632
ANDF 153
ANDFI. 156
AND_GE(_U) 214
AND_GE_M2 1632
AND_GT(_U) 214
AND_GT_M2 1632
AND_LE(_U) 214
AND_LE_M2 1632
AND_LT(_U) 214
AND_LT_M2 1632
AND_NE(_U) 214
AND_NE_M2 1632
ANDP 153
ANDPI. 156
ANDSTRING_EQ 785
ANDSTRING_GE 785
ANDSTRING_GT 785
ANDSTRING_LE 785
ANDSTRING_LT 785
ANDSTRING_NE 785
ANDTM< 1110
ANDTM<= 1110
ANDTM<> 1110
ANDTM= 1110
ANDTM> 1110
ANDTM>= 1110
ANDTM_EQ. 1110

I

ANDTM_GE 1110
ANDTM_GT 1110
ANDTM_LE 1110
ANDTM_LT 1110
ANDTM_NE 1110
ANI 150,1291,1294
ASC2INT(P) 494
ASIN(_E) 1451
ASIN(P) 930
ASIND(P) 942
ASIND_M2 1633
ASIN_E_M2 1645
ASIN_M2 1633
ATAN(_E) 1453
ATAN(P) 934
ATAND(P) 946
ATAND_M2 1633
ATAN_E_M2 1645
ATAN_M2 1633

B

B-(P) 258,259
B*(P) 269
B/(P) 271
B+(P) 255,256
BACOS(P) 956
BAND(P)(_U) 705
BAND_M2 1633
BANDP_M2 1633
BASIN(P) 954
BATAN(P) 958
BCD(P) 409
BCDDA(P) 819
BCD_TO_DINT(_E) 1388
BCD_TO_DINT_E_M2 1645
BCD_TO_INT(_E) 1386
BCD_TO_INT_E_M2 1645
BCD_TO_STR_E_M2 1645
BCD_TO_STRING(_E) 1391
BCOS(P) 950
BDIVISION(P) 271
BDSQR_M2 1633
BDSQRP_M2 1633
BDSQRT(P) 982
BIN(P) 413
BINDA(P)(_U) 796
BINDA_M2 1633
BINDAP_M2 1633
BINHA(P) 805
BITARR_TO_DINT(_E) 1430
BITARR_TO_DINT_E_M2 1645
BITARR_TO_INT(_E) 1429
BITARR_TO_INT_E_M2 1645
BK-(P)(_U) 279
BK+(P)(_U) 277
BKAND(P) 303
BKBCD(P) 465
BKBIN(P) 467
BKCMP<(P)(_U) 226
BKCMP<=(P)(_U) 226
BKCMP<>(P)(_U) 226
BKCMP=(P)(_U) 226
BKCMP>(P)(_U) 226
BKCMP>=(P)(_U) 226
BKCMP_EQ(P)(_U) 226
BKCMP_EQ_M2 1634
BKCMP_EQP_M2 1634
BKCMP_GE(P)(_U) 226
BKCMP_GE_M2 1634
BKCMP_GEP_M2 1634
BKCMP_GT(P)(_U) 226
BKCMP_GT_M2 1634
BKCMP_GTP_M2 1634
BKCMP_LE(P)(_U) 226
BKCMP_LE_M2 1634
BKCMP_LEP_M2 1634
BKCMP_LT(P)(_U) 226
BKCMP_LT_M2 1634
BKCMP_LTP_M2 1634
BKCMP_NE(P)(_U) 226
BKCMP_NE_M2 1634
BKCMP_NEP_M2 1634
BKMINUS(P)(_U) 279
BKMINUS_M2 1633,1634
BKMINUSP_M2 1633,1634
BKOR(P) 313
BKPLUS(P)(_U) 277
BKPLUS_M2 1633,1634
BKPLUSP_M2 1633,1634
BKRST(P) 343
BKRST_M2 1634
BKRSTP_M2 1634
BKXNR(P) 333
BKXOR(P) 323
BLKMOVB(P) 558
BMINUS(P) 259
BMOV(P) 534,1302
BMOVL(P) 537
BMULTI(P) 269
BON(P) 735
BOOL_TO_DINT(_E) 1344
BOOL_TO_DINT_E_M2 1645
BOOL_TO_DWORD(_E) 1342
BOOL_TO_DWORD_E_M2 1645
BOOL_TO_INT(_E) 1343
BOOL_TO_INT_E_M2 1645
BOOL_TO_STR_E_M2 1645
BOOL_TO_STRING(_E) 1346
BOOL_TO_TIME(_E) 1345
BOOL_TO_TIME_E_M2 1645
BOOL_TO_WORD(_E) 1340
BOOL_TO_WORD_E_M2 1645
BPLUS(P) 256
BREAK(P) 583
BRSET 1317
BRST(P) 337
BSET(P) 335
BSFL(P) 351
BSFR(P) 349
BSIN(P) 948
BSQR_M2 1634
BSQRP_M2 1634
BSQRT(P) 980
BTAN(P) 952
BTOW(P) 519
BXCH(P) 551

C

CALL(P) 585
CCD(P) 776

CJ 563
CML(P) 525
CMLB(P) 532
CMP(P)(_U) 218
COM(P) 1133
CONCAT(_E) 1500
CONCAT_E_M2 1645
CONTWR(P) 1326
COS(_E) 1449
COS(P) 926
COSD(P) 938
COSD_M2 1634
COS_E_M2 1645
COS_M2 1634
COUNTER_FB_M. 1535
CPY_BITARR(_E). 1433
CPY_BITARR_E_M2. 1645
CPY_BIT_OF_INT(_E) 1438
CPY_BIT_OF_INT_E_M2 1645
CRC(P) 774
CTD(_E) 1530
CTD_E_M2 1648
CTD_M2 1648
CTU(_E) 1528
CTUD(_E) 1532
CTUD_E_M2 1648
CTUD_M2 1648
CTU_E_M2 1648
CTU_M2 1648

D

D(P).DDRD 1285
D(P).DDWR 1288
D-(P)(_U). 243,245
D(P)_DDRD 1285
D(P)_DDWR. 1288
D*(P)(_U). 251
D/(P)(_U) 253
D+(P)(_U) 239,241
DABCD(P) 482
DABIN(P)(_U). 469
DABIN_M2. 1635
DABINP_M2. 1635
DAND(P) 299,301
DATATRG 1172
DATE-(P). 1096
DATE+(P) 1094
DATE2SEC(P)(_U) 1102
DATEMINUS(P) 1096
DATEPLUS(P) 1094
DATERD(P) 1090
DATEWR(P). 1092
DB-(P) 265,267
DB*(P) 273
DB/(P) 275
DB+(P) 261,263
DBAND(P)(_U) 707
DBAND_M2 1635
DBANDP_M2 1635
DBCD(P) 411
DBCDDA(P). 823
DBCLOSE(P) 1053
DBCOMMIT(P). 1082
DBDELETE(P) 1076
DBDIVISION(P) 275
DBEXPORT(P). 1048
DBIMPORT(P) 1045
DBIN(P). 415
DBINDA(P)(_U) 800
DBINDA_M2 1635
DBINDAP_M2 1635
DBINHA(P) 809
DBINSERT(P) 1055
DBK-(P)(_U) 284
DBK+(P)(_U) 281
DBKCMP<(P)(_U) 228
DBKCMP<=(P)(_U) 228
DBKCMP<>(P)(_U) 228
DBKCMP=(P)(_U) 228
DBKCMP>(P)(_U) 228
DBKCMP>=(P)(_U) 228
DBKCMP_EQ(P)(_U) 228
DBKCMP_EQ_M2 1635
DBKCMP_EQP_M2 1635
DBKCMP_GE(P)(_U) 228
DBKCMP_GE_M2 1635
DBKCMP_GEP_M2 1635
DBKCMP_GT(P)(_U) 228
DBKCMP_GT_M2 1635
DBKCMP_GTP_M2 1635
DBKCMP_LE(P)(_U). 228
DBKCMP_LE_M2. 1635
DBKCMP_LEP_M2. 1635
DBKCMP_LT(P)(_U). 228
DBKCMP_LT_M2. 1635
DBKCMP_LTP_M2. 1635
DBKCMP_NE(P)(_U) 228
DBKCMP_NE_M2 1635
DBKCMP_NEP_M2 1635
DBKMINUS(P)(_U). 284
DBKMINUS_M2 1635
DBKMINUSP_M2 1635
DBKPLUS(P)(_U). 281
DBKPLUS_M2 1635,1636
DBKPLUSP_M2 1635,1636
DBL2DINT(P). 429
DBL2FLT(P). 902
DBL2INT(P) 425
DBL2UDINT(P) 431
DBL2UINT(P). 427
DBMINUS(P) 267
DBMULTI(P) 273
DBMULTI_M2 1635,1636
DBMULTIP_M2 1635,1636
DBON(P). 737
DBOPEN(P). 1051
DBPLUS(P) 263
DBROLBAK(P). 1084
DBSELECT(P) 1068
DBTRANS(P). 1080
DBUPDATE(P). 1062
DCML(P) 527
DCMP(P)(_U). 220
DCONTSW 1324
DDABCD(P). 485
DDABIN(P)(_U) 472
DDABIN_M2 1636
DDABINP_M2 1636
DDEC(P)(_U). 293
DDEC_M2 1636
DDECP_M2 1636

I

DDIVISION(P)(_U) 253
DDIVISION_M2 1635,1636
DDIVISIONP_M2 1635,1636
DDSFL(P) 359
DDSFR(P) 357
DEC(P)(_U) 289
DEC_M2 1636
DECO(P) 502
DECP_M2 1636
DEG(P) 962
DEGD(P) 966
DELETE(_E) 1504
DELETE_E_M2 1645
DELTA(P) 193
DFMOV(P) 543
DFMOVL(P) 545
DFROM(P) 1137
DFROMD(P) 1146
DGBIN(P)(_U) 463
DGBIN_M2 1636
DGBINP_M2 1636
DGRY(P)(_U) 459
DGRY_M2 1636
DGRYP_M2 1636
DHABIN(P) 479
DHOURM 1129
DI 567,570
DI_1 570
DINC(P)(_U) 291
DINC_M2 1636
DINCP_M2 1636
DINT2DBL(P) 908
DINT2FLT(P) 898
DINT2INT(P) 445
DINT2UDINT(P) 449
DINT2UINT(P) 447
DINT_TO_BCD(_E) 1379
DINT_TO_BCD_E_M2 1645
DINT_TO_BITARR(_E) 1432
DINT_TO_BITARR_E_M2 1645
DINT_TO_BOOL(_E) 1374
DINT_TO_BOOL_E_M2 1645
DINT_TO_DWORD(_E) 1377
DINT_TO_DWORD_E_M2 1645
DINT_TO_INT(_E) 1378
DINT_TO_INT_E_M2 1645
DINT_TO_LREAL(_E) 1382
DINT_TO_LREAL_E_M2 1645
DINT_TO_REAL(_E) 1381
DINT_TO_REAL_E_M2 1645
DINT_TO_STR_E_M2 1645
DINT_TO_STRING(_E) 1384
DINT_TO_TIME(_E) 1383
DINT_TO_TIME_E_M2 1645
DINT_TO_WORD(_E) 1375
DINT_TO_WORD_E_M2 1645
DIS(P) 509
DIV(_E) 1462
DIV_E_M2 1645
DIVISION(P)(_U) 249
DIVISION_M2 1632,1636
DIVISIONP_M2 1632,1636
DIV_TIME(_E) 1517
DIV_TIME_E_M2 1645
DLIMIT(P)(_U) 703
DLIMIT_M2 1636
DLIMITP_M2 1636
DMAX(P)(_U) 741
DMAX_M2 1636
DMAXP_M2 1636
DMEAN(P)(_U) 769
DMEAN_M2 1636
DMEANP_M2 1636
DMIN(P)(_U) 745
DMIN_M2 1636
DMINP_M2 1636
DMINUS(P)(_U) 245
DMINUS_2_M2 1636
DMINUS_M2 1634,1636
DMINUSP_2_M2 1637
DMINUSP_M2 1634,1637
DMOV(P) 523,1299
DMULTI(P)(_U) 251
DMULTI_M2 1634,1637
DMULTIP_M2 1634,1637
DNEG(P) 500
DOR(P) 309,311
DPLUS(P)(_U) 241
DPLUS_2_M2 1637
DPLUS_M2 1634,1637
DPLUSP_2_M2 1637
DPLUSP_M2 1634,1637
DRCL(P) 623
DRCR(P) 621
DROL(P) 623
DROL_M2 1637
DROR(P) 621
DROR_M2 1637
DSCL(P)(_U) 716
DSCL2(P)(_U) 721
DSCL2_M2 1637
DSCL2P_M2 1637
DSCL_M2 1637
DSCLP_M2 1637
DSERDATA(P) 725
DSERMM(P) 729
DSFL(P) 355
DSFR(P) 353
DSORTD(_U) 749
DSORT_M2 1637
DSORTTBL2(_U) 759
DSQRT(P) 773
DSTR(P)(_U) 816
DSTR_M2 1637
DSTRP_M2 1637
DSUM(P) 733
DSWAP(P) 554
DTEST(P) 341
DTO(P) 1141
DTOD(P) 1150
DTO_M2 1637
DTOP_M2 1637
DUTY 1123
DVAL(P)(_U) 491
DVAL_M2 1637
DVALP_M2 1637
DWORD_TO_BOOL(_E) 1354
DWORD_TO_BOOL_E_M2 1645
DWORD_TO_DINT(_E) 1359
DWORD_TO_DINT_E_M2 1645
DWORD_TO_INT(_E) 1357
DWORD_TO_INT_E_M2 1645

DWORD_TO_STR_E_M2 1645
DWORD_TO_STRING(_E) 1361
DWORD_TO_TIME(_E). 1360
DWORD_TO_TIME_E_M2 1645
DWORD_TO_WORD(_E) 1355
DWORD_TO_WORD_E_M2 1645
DWSFTL(P) 391
DWSFTR(P). 387
DWSUM(P)(_U) 765
DWSUM_M2 1637
DWSUMP_M2 1637
DXCH(P) 549
DXNR(P) 329,331
DXOR(P) 319,321
DZCP(P)(_U) 224
DZONE(P)(_U). 711
DZONE_M2 1637
DZONEP_M2 1637

E

E-(P) 874,876
E*(P) 886
E/(P) 888
E+(P). 870,872
ECALL(P). 594
ECMP(P) 862
ECONTSW. 1324
ED-(P) 882,884
ED*(P) 890
ED/(P) 892
ED+(P) 878,880
EDCMP(P) 864
EDDIVISION(P) 892
EDIVISION(P) 888
EDMAX(P) 994
EDMIN(P) 998
EDMINUS(P) 884
EDMOV(P). 923
EDMULTI(P). 890
EDNEG(P) 921
EDPLUS(P) 880
EDSFL(P) 367
EDSFR(P) 365
EDSFTL(P) 407
EDSFTR(P) 403
EDSQRT(P) 970
EDZCP(P) 868
EFCALL(P). 599
EGF 164
EGP 164
EI 567
EMAX(P) 992
EMIN(P). 996
EMINUS(P) 876
EMOD(P). 496
EMOV(P) 922
EMULTI(P). 886
ENCO(P) 504
END 208
ENEG(P) 920
EPLUS(P) 872
EQ(_E) 1490
EQ_E_M2 1645
EREXP(P) 918
EREXP_M2 1637
EREXPP_M2 1637
ESFL(P) 363
ESFR(P) 361
ESFTL(P) 399
ESFTR(P) 395
ESQRT(P). 968
ESTR(P) 828
EVAL(P) 914
EXP(_E) 1447
EXP(P) 972
EXPD(P) 974
EXPD_M2 1638
EXP_E_M2 1645
EXP_M2 1638
EXPT(_E) 1466
EXPT_E_M2 1645
EZCP(P) 866

F

FCALL(P) 590
FDEL(P) 633
FEND 207
FF. 191
FIFR(P). 625
FIFW(P) 629
FIND(_E). 1509
FIND_E_M2. 1645
FINS(P). 631
FLT2DBL(P). 912
FLT2DINT(P) 421
FLT2INT(P) 417
FLT2UDINT(P). 423
FLT2UINT(P) 419
FMOV(P). 539
FMOVL(P) 541
FOR 581
FPOP(P) 627
FROM(P). 1137
FROMD(P). 1146
F_TRIG(_E). 1526
F_TRIG_E_M2. 1648
F_TRIG_M2. 1648

G

G_BIDIN_M2 1633,1638
G_BIDOUT_M2 1633,1638
GBIN(P)(_U) 461
GBIN_M2. 1638
GBINP_M2 1638
G_CCPASET_M2. 1638
G_CPRTCL_M2 1638
GE(_E) 1490
GE_E_M2 1645
GET_BIT_OF_INT(_E) 1434
GET_BIT_OF_INT_E_M2 1645
GET_BOOL_ADDR 1440
GET_INT_ADDR 1440
GET_WORD_ADDR. 1440
G_GETE_M2 1638,1639
G_INPUT_M2 1638,1639
GOEND. 566
G_OGLOAD_M2 1638,1641
G_OGSTOR_M2 1638,1641
G_ONDEMAND_M2 1638,1641

I

G_OUTPUT_M2 1638,1641
GP_BIDIN_M2 1633,1639
GP_BIDOUT_M2 1633,1639
GP_CCPASET_M2 1639
GP_CPRTCL_M2 1639
GP_ECPRTCL_M2 1639
GP_GETE_M2 1639
GP_OGLOAD_M2 1639,1641
GP_OGSTOR_M2 1639,1641
GP_ONDEMAND_M2 1639,1641
GP_OUTPUT_M2 1639,1641
GP_PRR_M2 1639,1642
GP_PUTE_M2 1639,1642
GP_RDMSG_M2 1639,1642
GP_RECV_M2 1639,1642
GP_REQ_M2 1639,1642
GP_RIRCV_M2 1639,1642
GP_RIRD_M2 1639,1642
GP_RISEND_M2 1639,1642
GP_RITO_M2 1639,1642
GP_RIWT_M2 1639,1642
GP_RLPASET_M2 1639,1642
G_PRR_M2 1638,1642
GP_SEND_M2 1639,1642
GP_SPBUSY_M2 1639,1642
G_PUTE_M2 1638,1642
G_RDMSG_M2 1638,1642
G_REQ_M2 1638,1642
G_RIRCV_M2 1638,1642
G_RIRD_M2 1638,1642
G_RISEND_M2 1638,1642
G_RITO_M2 1638,1642
G_RIWT_M2 1638,1642
G_RLPASET_M2 1638,1642
GRY(P)(_U) 457
GRY_M2 1639
GRYP_M2 1639
G_SPBUSY_M2 1638,1642
GT(_E) 1490
GT_E_M2 1645

H

HABIN(P) 476
HOURM 1127

I

IMASK 575
INC(P)(_U) 287
INC_M2 1639
INCP_M2 1639
INSERT(_E) 1502
INSERT_E_M2 1646
INSTR(P) 851
INT2ASC(P) 832
INT2DBL(P) 904
INT2DINT(P) 435
INT2FLT(P) 894
INT2UDINT(P) 437
INT2UINT(P) 433
INT_TO_BCD(_E) 1367
INT_TO_BCD_E_M2 1646
INT_TO_BITARR(_E) 1431
INT_TO_BITARR_E_M2 1646
INT_TO_BOOL(_E) 1362
INT_TO_BOOL_E_M2 1646
INT_TO_DINT(_E) 1366
INT_TO_DINT_E_M2 1646
INT_TO_DWORD(_E) 1364
INT_TO_DWORD_E_M2 1646
INT_TO_LREAL(_E) 1370
INT_TO_LREAL_E_M2 1646
INT_TO_REAL(_E) 1369
INT_TO_REAL_E_M2 1646
INT_TO_STR_E_M2 1646
INT_TO_STRING(_E) 1372
INT_TO_TIME(_E) 1371
INT_TO_TIME_E_M2 1646
INT_TO_WORD(_E) 1363
INT_TO_WORD_E_M2 1646
INV 162
IRET 579

J

JMP 563

L

LD 150,1291,1294
LD<(_U) 214
LD<=(_U) 214
LD<>(_U) 214
LD=(_U) 214
LD>(_U) 214
LD>=(_U) 214
LD$< 785
LD$<= 785
LD$<> 785
LD$= 785
LD$> 785
LD$>= 785
LDD<(_U) 216
LDD<=(_U) 216
LDD<>(_U) 216
LDD=(_U) 216
LDD>(_U) 216
LDD>=(_U) 216
LDD_EQ(_U) 216
LDD_EQ_M2 1640
LDD_GE(_U) 216
LDD_GE_M2 1640
LDD_GT(_U) 216
LDD_GT_M2 1640
LDD_LE(_U) 216
LDD_LE_M2 1640
LDD_LT(_U) 216
LDD_LT_M2 1640
LDD_NE(_U) 216
LDD_NE_M2 1640
LDDT< 1106
LDDT<= 1106
LDDT<> 1106
LDDT= 1106
LDDT> 1106
LDDT>= 1106
LDDT_EQ 1106
LDDT_GE 1106
LDDT_GT 1106
LDDT_LE 1106
LDDT_LT 1106

LDDT_NE 1106
LDE< 857
LDE<= 857
LDE<> 857
LDE= 857
LDE> 857
LDE>= 857
LDED< 859
LDED<= 859
LDED<> 859
LDED= 859
LDED> 859
LDED>= 859
LDED_EQ 859
LDED_GE 859
LDED_GT 859
LDED_LE 859
LDED_LT 859
LDED_NE 859
LDE_EQ 857
LDE_GE 857
LDE_GT 857
LDE_LE 857
LDE_LT 857
LDE_NE 857
LD_EQ(_U) 214
LD_EQ_M2 1639
LDF 153
LDFI 156
LD_GE(_U) 214
LD_GE_M2 1639
LD_GT(_U) 214
LD_GT_M2 1639
LDI 150,1291,1294
LD_LE(_U) 214
LD_LE_M2 1639
LD_LT(_U) 214
LD_LT_M2 1639
LD_NE(_U) 214
LD_NE_M2 1639
LDP 153
LDPI 156
LDSTRING_EQ 785
LDSTRING_GE 785
LDSTRING_GT 785
LDSTRING_LE 785
LDSTRING_LT 785
LDSTRING_NE 785
LDTM< 1110
LDTM<= 1110
LDTM<> 1110
LDTM= 1110
LDTM> 1110
LDTM>= 1110
LDTM_EQ 1110
LDTM_GE 1110
LDTM_GT 1110
LDTM_LE 1110
LDTM_LT 1110
LDTM_NE 1110
LE(_E) 1490
LEDR 779
LE_E_M2 1646
LEFT(_E) 1496
LEFT(P) 844
LEFT_E_M2 1646
LEFT_M2 1640
LEN(_E) 1494
LEN(P) 840
LEN_E_M2 1646
LEN_M2 1640
LIMIT(_E) 1485
LIMIT(P)(_U) 701
LIMITATION_E_M2 1646
LIMIT_M2 1640
LIMITP_M2 1640
LN(_E) 1444
LN_E_M2 1646
LOG(_E) 1445
LOG(P) 976
LOG10(P) 988
LOG10D(P) 990
LOGD(P) 978
LOGD_M2 1640
LOG_E_M2 1646
LOG_M2 1640
LOGTRG 1168
LOGTRGR 1170
LREAL_TO_DINT(_E) 1404
LREAL_TO_DINT_E_M2 1646
LREAL_TO_INT(_E) 1402
LREAL_TO_INT_E_M2 1646
LREAL_TO_REAL(_E) 1406
LREAL_TO_REAL_E_M2 1646
LT(_E) 1490
LT_E_M2 1646

M

M(P).DDRD 1285
M(P).DDWR 1288
M(P)_DDRD 1285
M(P)_DDWR 1288
MAX(_E) 1483
MAX(P)(_U) 739
MAXIMUM_E_M2 1646
MAX_M2 1640
MAXP_M2 1640
MC 197
MCR 197
MEAN(P)(_U) 767
MEAN_M2 1640
MEANP_M2 1640
MEF 163
MEP 163
MID(_E) 1498
MID_E_M2 1646
MIDR(P) 846
MIDW(P) 848
MIN(_E) 1483
MIN(P)(_U) 743
MINIMUM_E_M2 1646
MIN_M2 1640
MINP_M2 1640
MINUS(P)(_U) 237
MINUS_2_M2 1640
MINUS_M2 1632,1640
MINUSP_2_M2 1640
MINUSP_M2 1632,1640
MOD(_E) 1464
MOD_E_M2 1646
MOV(P) 521,1296

I

MOVB(P) 556
MOVE(_E) 1467
MOVE_E_M2 1646
MPP 160
MPS 160
MRD 160
M_REAL_TO_REAL_E_M2 1646
MTR 1042
MUL(_E) 1457
MUL_E_M2 1646
MULTI(P)(_U). 247
MULTI_M2 1632,1640
MUL_TIME(_E). 1515
MUL_TIME_E_M2 1646
MULTIP_M2 1632,1640
MUX(_E) 1488
MUX_E_M2 1646

N

NDIS(P) 513
NE(_E) 1492
NE_E_M2 1646
NEG(P) 498
NEXT 581
NOP 211
NOPLF 212
NOT(_E) 1480
NOT_E_M2 1646
NUNI(P). 515

O

OR 150,1291,1294
OR(_E) 1477
OR<(_U) 214
OR<=(_U) 214
OR<>(_U) 214
OR=(_U) 214
OR>(_U) 214
OR>=(_U) 214
OR$<. 785
OR$<=. 785
OR$<>. 785
OR$=. 785
OR$>. 785
OR$>=. 785
ORB 159
ORD<(_U) 216
ORD<=(_U) 216
ORD<>(_U) 216
ORD=(_U) 216
ORD>(_U) 216
ORD>=(_U) 216
ORD_EQ(_U) 216
ORD_EQ_M2 1641
ORD_GE(_U) 216
ORD_GE_M2 1641
ORD_GT(_U) 216
ORD_GT_M2 1641
ORD_LE(_U) 216
ORD_LE_M2 1641
ORD_LT(_U) 216
ORD_LT_M2 1641
ORD_NE(_U) 216
ORD_NE_M2 1641
ORDT< 1106
ORDT<= 1106
ORDT<> 1106
ORDT= 1106
ORDT> 1106
ORDT>= 1106
ORDT_EQ 1106
ORDT_GE 1106
ORDT_GT 1106
ORDT_LE 1106
ORDT_LT 1106
ORDT_NE 1106
ORE< 857
ORE<= 857
ORE<> 857
ORE= 857
ORE> 857
ORE>= 857
ORED< 859
ORED<= 859
ORED<> 859
ORED= 859
ORED> 859
ORED>= 859
ORED_EQ. 859
ORED_GE. 859
ORED_GT. 859
ORED_LE 859
ORED_LT 859
ORED_NE 859
ORE_EQ 857
ORE_GE 857
ORE_GT 857
ORE_LE 857
ORE_LT 857
OR_E_M2 1646
ORE_NE 857
OR_EQ(_U). 214
OR_EQ_M2 1641
ORF 153
ORFI. 156
OR_GE(_U). 214
OR_GE_M2 1641
OR_GT(_U). 214
OR_GT_M2 1641
ORI. 150,1291,1294
OR_LE(_U) 214
OR_LE_M2 1641
OR_LT(_U) 214
OR_LT_M2 1641
OR_NE(_U). 214
OR_NE_M2 1641
ORP 153
ORPI. 156
ORSTRING_EQ 785
ORSTRING_GE 785
ORSTRING_GT 785
ORSTRING_LE 785
ORSTRING_LT 785
ORSTRING_NE 785
ORTM< 1110
ORTM<= 1110
ORTM<> 1110
ORTM= 1110
ORTM> 1110
ORTM>= 1110

ORTM_EQ 1110
ORTM_GE 1110
ORTM_GT 1110
ORTM_LE 1110
ORTM_LT 1110
ORTM_NE 1110
OUT 166
OUT C 174
OUT F 178
OUT LC 176
OUT LST 171
OUT LT 171
OUT ST 168
OUT T 168
OUT_C 174
OUTH 168
OUTH ST 168
OUTH T 168
OUT_T 168

P

PABORT 783
PALERT(P) 781
PAUSE 1309
PHASE 202
PHASECHG 204
PHASEEND 206
PID 1245
PIDCONT(P) 1274
PIDINIT(P) 1272
PIDPRMW(P) 1279
PIDRUN(P) 1278
PIDSTOP(P) 1277
PLF 189
PLS 187
PLSY 1032
PLUS(P)(_U) 233
PLUS_2_M2 1641
PLUS_M2 1632,1641
PLUSP_2_M2 1641
PLUSP_M2 1632,1641
POFF(P) 611
POW(P) 984
POWD(P) 986
PSCAN(P) 613
PSTOP(P) 609
PWM 1034

Q

QDRSET(P) 1012

R

RAD(P) 960
RADD(P) 964
RAMPQ 1039
RCL(P) 618
RCR(P) 615
REAL_TO_DINT(_E) 1395
REAL_TO_DINT_E_M2 1646
REAL_TO_INT(_E) 1393
REAL_TO_INT_E_M2 1646
REAL_TO_LREAL(_E) 1397
REAL_TO_LREAL_E_M2 1646
REAL_TO_M_REAL_E_M2 1646
REAL_TO_STR_E_M2 1646
REAL_TO_STRING(_E) 1399
REPLACE(_E) 1506
REPLACE_E_M2 1646
RET 589
RFS(P) 1131
RIGHT(_E) 1496
RIGHT(P) 842
RIGHT_E_M2 1646
RIGHT_M2 1642
RND(P) 1000
ROL(_E) 1473
ROL(P) 618
ROL_E_M2 1646
ROL_M2 1642
ROR(_E) 1475
ROR(P) 615
ROR_E_M2 1646
ROR_M2 1642
ROTC 1036
RS(_E) 1522
RS_E_M2 1648
RSET(P) 1010
RS_M2 1648
RST 181,1307,1315
RST F 185
RSTART 1311
R_TRIG(_E) 1524
R_TRIG_E_M2 1648
R_TRIG_M2 1648

S

S.SOCRCVS 1183
S(P).DATE- 1121
S(P).DATE+ 1119
S(P).DATERD 1117
S(P).DEVLD 636
S(P).PIDCONT 1263
S(P).PIDINIT 1260
S(P).PIDPRMW 1268
S(P).PIDRUN 1267
S(P).PIDSTOP 1266
S(P).RTREAD 1164
S(P).RTWRITE 1166
S(P).SOCRDATA 1196
S(P).ZCOM 1135
S(P)_DATEMINUS 1121
S(P)_DATEPLUS 1119
S(P)_DATERD 1117
S(P)_DEVLD 636
S(P)_PIDCONT 1263
S(P)_PIDINIT 1260
S(P)_PIDPRMW 1268
S(P)_PIDRUN 1267
S(P)_PIDSTOP 1266
S(P)_RTREAD 1164
S(P)_RTWRITE 1166
S(P)_SOCRDATA 1196
S(P)_ZCOM 1135
SCJ 563
SCL(P)(_U) 713
SCL2(P)(_U) 719
SCL2_M2 1642
SCL2P_M2 1642

I

SCL_M2 1642
SCLP_M2 1642
SEC2DATE(P)(_U) 1104
SEC2TIME(P). 1100
SEG(P) 506
SEL(_E). 1481
SEL_E_M2. 1646
SERDATA(P) 723
SERMM(P). 727
SET. 179,1305,1313
SET F 183
SET_BIT_OF_INT(_E). 1436
SET_BIT_OF_INT_E_M2 1646
SFL(P). 347
SFR(P) 345
SFT(P). 195
SFTBL(P). 373
SFTBR(P) 369
SFTDWL(P) 389
SFTDWR(P). 385
SFTEDL(P) 405
SFTEDR(P) 401
SFTEL(P). 397
SFTER(P) 393
SFTL(P). 375
SFTR(P) 371
SFTWL(P) 381
SFTWR(P) 377
SHL(_E). 1469
SHL_E_M2. 1646
SHR(_E) 1471
SHR_E_M2 1646
SIMASK. 577
SIN(_E) 1448
SIN(P) 924
SIND(P). 936
SIND_M2. 1642
SIN_E_M2 1646
SIN_M2 1642
SJIS2WS(P). 836
SJIS2WSB(P). 838
SMOV(P) 529
SORTD(_U) 747
SORT_M2 1642
SORTTBL(_U) 751
SORTTBL2(_U) 755
SP.CONTSW 1320
SP.DEVST 638
SP.ECPRTCL. 1198
SP.FCOPY. 682
SP.FDELETE 678
SP.FMOVE 687
SP.FREAD. 641
SP.FRENAME 692
SP.FSTATUS 696
SP.FTPGET 1223
SP.FTPPUT 1213
SP.FWRITE 660
SP.SIDRD 1331
SP.SLMPSND 1206
SP.SOCCINF 1188
SP.SOCCLOSE 1177
SP.SOCCSET 1190
SP.SOCOPEN 1174
SP.SOCRCV 1179
SP.SOCRMODE. 1192
SP.SOCSND 1185
SP_CONTSW 1320
SPD 1030
SP_DEVST 638
SP_ECPRTCL 1198
SP_FCOPY 682
SP_FDELETE 678
SP_FMOVE 687
SP_FREAD 641
SP_FRENAME. 692
SP_FSTATUS 696
SP_FTPGET 1223
SP_FTPPUT 1213
SP_FWRITE 660
SP_SIDRD. 1331
SP_SLMPSND. 1206
SP_SOCCINF 1188
SP_SOCCLOSE. 1177
SP_SOCCSET. 1190
SP_SOCOPEN 1174
SP_SOCRCV. 1179
SP_SOCRMODE 1192
SP_SOCSND. 1185
SQRT(_E) 1443
SQRT(P) 771
SQRT_E_M2 1646
SR(_E) 1520
SR_E_M2 1648
SR_M2 1648
SRND(P) 1001
S_SOCRCVS. 1183
STMR 1027
STOP 210
STR(P)(_U) 813
STRDEL(P) 855
STRINGMOV(P). 792
STRINGMOV(P)_WS 794
STRINGPLUS(P) 790
STRING_TO_BCD(_E) 1422
STRING_TO_BOOL(_E) 1415
STRING_TO_DINT(_E). 1420
STRING_TO_DWORD(_E) 1417
STRING_TO_INT(_E). 1418
STRING_TO_REAL(_E) 1424
STRING_TO_TIME(_E). 1427
STRING_TO_WORD(_E) 1416
STRINS(P). 853
STR_M2 1643
STRP_M2 1643
STR_TO_BCD_E_M2 1646
STR_TO_BOOL_E_M2. 1646
STR_TO_DINT_E_M2 1646
STR_TO_DWORD_E_M2 1646
STR_TO_INT_E_M2. 1647
STR_TO_REAL_E_M2 1647
STR_TO_TIME_E_M2 1647
STR_TO_WORD_E_M2 1647
SUB(_E) 1459
SUB_E_M2 1647
SUB_TIME(_E) 1513
SUB_TIME_E_M2 1647
SUM(P) 731
SWAP(P). 553
SWAP_M2. 1643
SWAPP_M2. 1643

T

TAN(_E) 1450
TAN(P) 928
TAND(P) 940
TAND_M2 1643
TAN_E_M2 1647
TAN_M2 1643
TCMP(P) 1113
TEST(P) 339
TIMCHK. 1125
TIME2SEC(P). 1098
TIMER_100_FB_M 1545
TIMER_10_FB_M 1545
TIMER_CONT_FB_M 1545
TIMER_CONTHFB_M 1545
TIMER_HIGH_FB_M. 1545
TIMER_LOW_FB_M 1545
TIME_TO_BOOL(_E) 1408
TIME_TO_BOOL_E_M2 1647
TIME_TO_DINT(_E) 1412
TIME_TO_DINT_E_M2 1647
TIME_TO_DWORD(_E). 1410
TIME_TO_DWORD_E_M2 1647
TIME_TO_INT(_E) 1411
TIME_TO_INT_E_M2 1647
TIME_TO_STR_E_M2. 1647
TIME_TO_STRING(_E). 1413
TIME_TO_WORD(_E). 1409
TIME_TO_WORD_E_M2. 1647
TO(P) 1141
TOD(P) 1150
TOF(_E) 1543
TOF_HIGH_E_M2 1648
TOF_HIGH_M2 1648
TO_M2 1643
TON(_E) 1540
TON_HIGH_E_M2 1648
TON_HIGH_M2 1648
TOP_M2 1643
TP(_E). 1537
TP_HIGH_E_M2. 1648
TP_HIGH_M2. 1648
TRAN 1319
TTMR 1025
TYPERD(P) 1155
TZCP(P) 1115

U

UDCNT1 1020
UDCNT2 1023
UDINT2DBL(P). 910
UDINT2DINT(P) 455
UDINT2FLT(P) 900
UDINT2INT(P) 451
UDINT2UINT(P) 453
UINT2DBL(P). 906
UINT2DINT(P) 441
UINT2FLT(P) 896
UINT2INT(P) 439
UINT2UDINT(P) 443
UNI(P). 511
UNIINFRD(P) 1159

V

VAL(P)(_U) 488
VAL_M2 1643
VALP_M2 1643

W

WAND(P). 295,297
WDT(P). 580
WOR(P) 305,307
WORD_TO_BOOL(_E) 1347
WORD_TO_BOOL_E_M2 1647
WORD_TO_DINT(_E). 1350
WORD_TO_DINT_E_M2. 1647
WORD_TO_DWORD(_E) 1348
WORD_TO_DWORD_E_M2 1647
WORD_TO_INT(_E). 1349
WORD_TO_INT_E_M2. 1647
WORD_TO_STR_E_M2 1647
WORD_TO_STRING(_E) 1353
WORD_TO_TIME(_E). 1352
WORD_TO_TIME_E_M2 1647
WS2SJIS(P). 834
WSFL(P) 383
WSFR(P). 379
WSUM(P)(_U) 763
WSUM_M2 1643
WSUMP_M2 1643
WTOB(P). 517
WXNR(P). 325,327
WXOR(P) 315,317

X

XCALL 604
XCH(P) 547
XOR(_E) 1477
XOR_E_M2 1647

Z

Z_ABRST1_M2 1632,1643
Z_ABRST2_M2 1632,1643
Z_ABRST3_M2 1632,1643
Z_ABRST4_M2 1632,1643
ZCP(P)(_U) 222
ZONE(P)(_U) 709
ZONE_M2 1643
ZONEP_M2 1643
ZP_BUFRCV_M2 1634,1644
ZP_BUFSND_M2 1634,1644
ZP_CLOSE_M2 1634,1644
ZP_CSET_M2 1634,1644
ZP_ERRCLR_M2 1637,1644
ZP_ERRRD_M2 1637,1644
ZPOP(P) 1004,1008
ZPOP(P)_2 1008
ZP_OPEN_M2 1641,1644
ZP_PFWRT_M2. 1641,1644
ZP_PINIT_M2 1641,1644
ZP_PSTRT1_M2 1642,1644
ZP_PSTRT2_M2 1642,1644
ZP_PSTRT3_M2 1642,1644
ZP_PSTRT4_M2 1642,1644
ZP_REMTO_M2. 1642,1644

ZP_TEACH1_M2 1643,1644
ZP_TEACH2_M2 1643,1644
ZP_TEACH3_M2 1643,1644
ZP_TEACH4_M2 1643,1644
ZP_UINI_M2 1643,1644
ZPUSH(P) 1002,1005
ZPUSH(P)_2 1005
Z_RECVS_M2 1642,1643
ZRRDB(P) 1014
ZRWRB(P). 1016
Z_UINI_M2. 1643

REVISIONS

*The manual number is given on the bottom left of the back cover.

Revision date	*Manual number	Description
June 2014	SH(NA)-081266ENG-A	First edition
July 2014	SH(NA)-081266ENG-B	Error correction
October 2014	SH(NA)-081266ENG-C	■Added or modified parts Section 1.3, Chapter 11, 13, "Execution condition" (for each instruction)
January 2015	SH(NA)-081266ENG-D	■Added or modified parts Chapter 10, "FBD/LD" (for each instruction)
April 2015	SH(NA)-081266ENG-E	■Added or modified parts Section 7.26, 7.27, 13.5, 13.6, 16.8, 17.2
August 2015	SH(NA)-081266ENG-F	■Added or modified parts CONDITIONS OF USE FOR THE PRODUCT, TERMS, Section 1.2, 1.5, Chapter 2, 3, 4, Section 5.3, 5.5, 7.4, 7.26, 7.29, Chapter 12, WARRANTY, "SFC devices and safety devices" (for each instruction)
August 2015	SH(NA)-081266ENG-G	■Added or modified part Section 12.1
January 2016	SH(NA)-081266ENG-H	■Added or modified parts TERMS, Section 2.2, 2.3, 2.4, 2.5, 3.9, 3.10, 6.1, 6.5, 6.7, 7.9, 7.19, 7.20, 7.25, 7.26, 8.4, Chapter 9, 13, 20, Section 21.4, 21.5, Chapter 22, 23, Appendix 1, 2, 3, 4
May 2016	SH(NA)-081266ENG-I	■Added or modified parts TERMS, Section 2.3, 2.9, 3.4, 7.7, 8.5, Chapter 13, 14, 15, Section 15.2, 17.1, 17.2, 17.3, 17.4, 18.1, 18.2, 18.3, 18.4, 18.5, 18.6
June 2016	SH(NA)-081266ENG-J	■Added or modified part Section 8.5
October 2016	SH(NA)-081266ENG-K	■Added or modified parts Section 6.7, 7.22, 7.27
January 2017	SH(NA)-081266ENG-L	■Added or modified parts TERMS, Chapter 12, 21
May 2017	SH(NA)-081266ENG-M	■Added or modified parts Section 3.4, 6.1, 6.2, 6.6, 7.6, 7.8, 7.9, 7.21, 7.25, 18.9, 31.1, 31.2, Appendix 1
October 2017	SH(NA)-081266ENG-N	■Added or modified parts CONDITIONS OF USE FOR THE PRODUCT, INTRODUCTION, TERMS, Section 1.2, 1.5, 2.2, 2.3, 5.1, 6.5, 7.25, 27.2, Appendix 1, "SIL2 Process CPU" (for icons of each instruction)
April 2018	SH(NA)-081266ENG-O	• The chapters of the process control instructions are moved to MELSEC iQ-R Programming Manual (Process Control Function Blocks/Instructions). • The chapters of the module dedicated instructions are moved to MELSEC iQ-R Programming Manual (Module Dedicated Instructions). ■Added or modified part Appendix 1
July 2018	SH(NA)-081266ENG-P	■Added or modified parts Section 1.1, 6.5, 19.1, Appendix 1, 6
October 2018	SH(NA)-081266ENG-Q	■Added or modified parts Section 2.3, 8.3, 8.4, 8.6, Chapter 27, Section 28.3, Chapter 29, Appendix 1, 2
May 2019	SH(NA)-081266ENG-R	■Added or modified parts GENERIC TERMS AND ABBREVIATIONS, Section 20.1, 21.1, Appendix 6
October 2019	SH(NA)-081266ENG-S	■Added or modified part Section 19.1
February 2020	SH(NA)-081266ENG-T	■Added or modified part Appendix 1
May 2020	SH(NA)-081266ENG-U	■Added or modified parts Section 8.4, Chapter 23, Appendix 1, 2
October 2020	SH(NA)-081266ENG-V	■Added or modified parts Section 8.4, 20.1, Chapter 23
April 2021	SH(NA)-081266ENG-W	■Added or modified parts Section 2.1, 5.6, 8.4, Appendix 1, 2
October 2021	SH(NA)-081266ENG-X	■Added or modified parts Section 5.6, 8.4, 20.1, 28.1, 28.2, Appendix 1, 6

Revision date	*Manual number	Description
January 2022	SH(NA)-081266ENG-Y	■Added or modified parts GENERIC TERMS AND ABBREVIATIONS, Section 20.1
April 2022	SH(NA)-081266ENG-Z	■Added or modified part Section 27.1
November 2022	SH(NA)-081266ENG-AA	■Added or modified parts CONDITIONS OF USE FOR THE PRODUCT, TRADEMARKS
December 2023	SH(NA)-081266ENG-AB	■Added or modified parts Section 5.2, 6.7, 7.3, 16.1, 27.1, Appendix 1
October 2024	SH(NA)-081266ENG-AC	■Added or modified parts RELEVANT MANUALS, Section 1.5, 5.2, 5.3, 6.2, 7.4, 22.1, 24.5, 39.4, 43.8, Appendix 1

Japanese manual number: SH-081226-AD

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.

© 2014 MITSUBISHI ELECTRIC CORPORATION

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.

TRADEMARKS

Microsoft, Microsoft Access, Excel, SQL Server, Visual Basic, Visual C++, Visual Studio, Windows, Windows NT, Windows Server, Windows Vista, and Windows XP are trademarks of the Microsoft group of companies.
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.

SH(NA)-081266ENG-AC(2410)
MODEL: R-P-MF-E

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE : TOKYO BUILDING, 2-7-3 MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS : 1-14 , YADA-MINAMI 5-CHOME , HIGASHI-KU, NAGOYA , 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.