## MITSUBISHI

Mitsubishi Programmable Controller


## MELSEC-Q/L Structured Programming Manual

Common Instructions

## - SAFETY PRECAUTIONS

(Read these precautions before using this product.)

Before using MELSEC-Q or -L series programmable controllers, please read the manuals included with each 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 the manuals included with each product, and keep the manuals in a safe place for future reference.

## -CONDITIONS OF USE FOR THE PRODUCT•

(1) Mitsubishi 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 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'S USER, 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 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 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 representative in your region.

| Print date | Manual number | Revision |
| :---: | :---: | :---: |
| Jul., 2008 | SH(NA)-080783ENG-A | First edition |
| Jan., 2009 | SH(NA)-080783ENG-B | Model Addition <br> Q00UJCPU, Q00UCPU, Q01UCPU, Q10UDHCPU, Q10UDEHCPU, Q20UDHCPU, <br> Q20UDEHCPU <br> Addition <br> Section 2.4, Section 2.5, Section 6.1.7, Section 6.2.14, Section 6.4.8, Section 7.3.3, Section 7.3.5, Section 7.5.15, Section 7.11.18, Section 7.11.19, Section 7.12.17, Section 7.12.18, Section 7.12.25, Section 7.13.4, Section 7.13.5, Section 7.15.7, Section 7.15.8, Section 8.14, Section 8.15.1, Section 8.15.2, Section 8.15.3 Correction <br> MANUALS, Generic Terms and Abbreviations in This Manual, Section 3.4, <br> Section 3.5.1, Section 3.7, Section 3.8, Section 3.11, Chapter 4, Section 5.3, <br> Section 6.1.6, Section 6.2.13, Section 6.3.1, Section 6.4.6, Section 6.4.7, <br> Section 7.4, Section 7.8.1, Section 7.8.2, Section 7.11.8, Section 7.12.7, <br> Section 7.12.8, Section 7.12.9, Section 7.12.10, Section 7.12.11, Section 7.12.12, <br> Section 7.12.15, Section 7.12.16, Section 7.12.19, Section 7.15.1, Section 8.1, <br> Section 8.2, Section 8.11, Section 8.11.2, Section 8.13 |
| Jul., 2009 | SH(NA)-080783ENG-C | Model Addition <br> Q00JCPU, Q00CPU, Q01CPU <br> Addition <br> Section 7.18.10 <br> Correction <br> Section 1.2, Section 2.1, Section 2.2, Section 2.5.6, Section, 2.5.13, Section 2.5.16, Section 2.5.18, Section 2.5.19, Section 3.1, Section 3.5, Section 3.10, Chapter 4, Section 5.1 to 5.7 , Section 6.1 to 6.8 , Section 7.1 to 7.17 , Section 8.1 to 8.15 , <br> Section 9.2, Section 9.3 <br> Section 2.5.16 to 2.5.17 are changed to Section 2.5.17 to 2.5.18, <br> Section 2.5.19 is changed to Section 2.5.20, <br> Section 3.8 to 3.11 are changed to Section 3.2 to 3.5, <br> Section 7.16 is changed to Section 7.17, <br> Section 7.17.1 to 7.17.8 are changed to Section 7.18.1 to 7.18.8, <br> Section 8.1 is changed to Section 7.18.9, <br> Section 8.2 to 8.10 are changed to Section 7.18 .11 to 7.18 .19 , <br> Section 8.11 to 8.12 are changed to Section 8.1 to 8.2, <br> Section 8.13 to 8.14 are changed to Section 7.6 .6 to 7.6.7, <br> Section 8.15 is changed to Section 7.16 <br> Deletion <br> Section 3.2 to 3.7, Appendix |
| Oct., 2009 | SH(NA)-080783ENG-D | Addition <br> Section 5.1.3, Appendix 1 <br> Correction <br> Section 1.2, Section 2.1, Section 2.3.1, Chapter 4, Section 5.1.2 |
| Jan., 2010 | SH(NA)-080783ENG-E | Model Addition <br> L02CPU, L26CPU-BT <br> Addition <br> CONDITIONS OF USE FOR THE PRODUCT, Section 2.6, Section 7.18.20, <br> Chapter 8 <br> Correction <br> MANUALS, Section 1.1, Section 1.2, Section 2.1, Section 2.5.18, Section 3.2, <br> Section 3.5, Chapter 4, Chapter 5, Chapter 6, Chapter 7, Chapter 9, Chapter 10, Appendix 1, Section 2.5.19 to 2.5.20 are changed to Section 2.7 to 2.8 Chapter 8 is changed to Chapter 9, Chapter 9 is changed to Chapter 10 |


| Print date | Manual number | Revision |
| :---: | :---: | :---: |
| Apr., 2010 | SH(NA)-080783ENG-F | Model Addition <br> Q50UDEHCPU, Q100UDEHCPU <br> Correction <br> Section 1.2, Section 3.5, Chapter 4, Section 7.6.6, Section 7.18.19, Section 8.1.1, Section 8.2.1, Section 8.2.2, Section 9.1 |
| Sep., 2010 | SH(NA)-080783ENG-G | Addition <br> Section 3.2 <br> Correction <br> Chapter 4, Section 5.1.2, Section 5.1.3, Section 5.3.2, Section 5.3.3, Section 5.6.1, Section 6.1.1, Section 6.1.2, Section 6.3.4, Section 6.5.1, Section 7.5.15, Section 7.6.3, Section 7.6.6, Section 7.6.7, Section 7.8.1, Section 7.8.2, Section 7.11.7, Section 7.14.3, Section 7.15.7, Section 7.16.1, Section 7.16.2, Section 7.16.3, Section 7.18.12, Section 7.18.13, Section 8.2.2 |
| Jan., 2011 | SH(NA)-080783ENG-H | Correction <br> Section 1.1, Section 2.1, Section 2.5.9, Section 2.5.18, Section 3.2, Section 5.3.2, Section 5.3.3, Section 5.3.4, Section 5.3.7, Section 6.1.7, Section 7.6.5, Section 7.6.6, Section 7.8.2, Section 7.9.1, Section 7.9.3, Section 7.17.1, Section 7.17.2, Section 7.17.5, Section 7.18.7, Section 7.18.9, Section 7.18.10, Section 7.18.16, Section 7.18.17, Section 7.18.18, Section 8.1.1 |
| Mar., 2011 | SH(NA)-080783ENG-I | Correction <br> Section 1.1, Section 2.1, Section 2.2, Section 2.3.3, Section 2.3.4, <br> Section 2.4.1 to Section 2.4.8, Section 2.5.1 to Section 2.5.13, <br> Section 2.5.15 to Section 2.5.18, Section 2.6.1, Section 2.6.2, Section 2.7.1, <br> Section 2.7.2, Section 2.8.1, Chapter 4, <br> Section 5.1.2 to Section 5.1.4, Section 5.2.1 to Section 5.2.3, <br> Section 5.5.1, Section 6.1.1 to Section 6.1.5, Section 6.2.3, Section 6.2.4, <br> Section 6.2.8, Section 6.2.11, Section 6.2.17, Section 6.3.3, Section 6.3.4, <br> Section 6.3.10, Section 6.3.13, Section 6.3.16, Section 6.3.17, Section 6.4.4, <br> Section 6.4.7, Section 6.6.1, Section 6.6.2, Section 6.7.1, Section 6.8.4, <br> Section 6.8.6, Section 6.8.7, Section 6.8.9, Section 7.2.2, Section 7.2.4, <br> Section 7.3.3, Section 7.5.3, Section 7.5.4, Section 7.5.9, Section 7.5.10, <br> Section 7.5.12, Section 7.6.1, Section 7.8.1, Section 7.8.2, Section 7.9.2, <br> Section 7.9.3, Section 7.10.2, Section 7.11.20, Section 7.11.21, <br> Section 7.12.1 to Section 7.12.26, Section 7.14.1 to Section 7.14.3, <br> Section 7.15.3, Section 7.15.5, Section 7.15.7, Section 7.17.3, Section 7.18.2, <br> Section 7.18.6, Section 7.18.9, Section 8.1.1, Section 9.1.1, Section 9.1.2, <br> Section 9.2.1, Section 10.1 to Section 10.3 |
| Jul., 2011 | SH(NA)-080783ENG-J | Model Addition <br> L02CPU-P, L26CPU-PBT <br> Correction <br> Section 1.2, Section 3.2, Section 3.5, Chapter 4, Section 5.1.2, Section 5.4.1, <br> Section 5.5.1, Section 5.6.1, Section 6.5.1, Section 6.6.2, Section 6.8.10, <br> Section 7.3.2, Section 7.3.3, Section 7.3.4, Section 7.3.5, Section 7.5.6, <br> Section 7.5.12, Section 7.6.3, Section 7.6.4, Section 7.6.6, Section 7.7.1, <br> Section 7.7.2, Section 7.7.3, Section 7.7.4, Section 7.10.2, Section 7.11.7, <br> Section 7.11.20 to Section 7.11.26, Section 7.12.28, Section 7.14.3, Section 7.17.5, <br> Section 7.18.7, Section 7.18.8, Section 7.18.13, Section 7.18.15, Section 7.18.16, <br> Section 7.18.18, Section 7.18.19, Section 8.1.1, Section 8.2.1, Section 8.2.2 |


| Print date | Manual number | Revision |
| :---: | :---: | :---: |
| Apr., 2012 | SH(NA)-080783ENG-K | "PLC" was changed to "programmable controller". <br> Correction <br> INTRODUCTION, Section 1.1, Section 1.2, Section 5.3.2, Section 6.1.3, <br> Section 6.1.4, Section 6.2.9 to Section 6.2.12, Section 6.3.5, Section 6.3.6, <br> Section 6.3.16, Section 6.3.17, Section 6.4.2, Section 6.4.3, Section 7.5.4, <br> Section 7.8.2, Section 7.9.1, Section 7.9.2, Section 7.10.2, Section 7.11.7, <br> Section 7.11.11, Section 7.11.15, Section 7.11.20, Section 7.12.1 to Section 7.12.26, <br> Section 7.18.9, Section 7.18.10 |
| Feb., 2013 | SH(NA)-080783ENG-L | Revised according to the model addition of Process CPU, Redundant CPU, Universal model QCPU, and LCPU <br> Model Addition <br> Q02PHCPU, Q06PHCPU, Q12PHCPU, Q25PHCPU, Q12PRHCPU, Q25PRHCPU, Q03UDVCPU, Q04UDVCPU, Q06UDVCPU, Q13UDVCPU, Q26UDVCPU, L02SCPU, L06CPU, L26CPU |
| Jun., 2013 | SH(NA)-080783ENG-M | Model Addition <br> L02SCPU-P, L06CPU-P, L26CPU-P <br> Correction <br> Section 1.1, 1.2, 3.5, 5.1.3, 6.1.7, 6.2.14, 6.4.8, 6.5.1, 6.8.1, 6.8.2, 6.8.3, 6.8.4, 6.8.5, 6.8.6, 6.8.7, 6.8.8, 6.8.9, 6.8.10, 7.3.3, 7.3.5, 7.5.15, 7.6.7, 7.11.8, 7.11.19, 7.12.17, $7.12 .18,7.12 .25,7.12 .26,7.13 .4,7.13 .5,7.15 .7,7.15 .8,7.18 .9,7.18 .12,7.18 .13$, 7.18.16, 7.18.18, 7.18.20, 8.2.2, 8.3.1, 8.3.2, 8.3.3, 8.3.4, 9.1, 9.1.2, 9.2, Appendix 1 |
| Sep., 2013 | SH(NA)-080783ENG-N | Correction <br> Section 6.8.4 |
|  |  |  |

Japanese manual version SH-080736-S
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.

## INTRODUCTION

Thank you for purchasing the Mitsubishi MELSEC-Q or -L series programmable controllers.
Before using this product, please read this manual and the relevant manuals carefully and develop familiarity with the programming specifications to handle the product correctly.
When applying the program examples introduced in this manual to the actual system, ensure the applicability and confirm that it will not cause system control problems.

## CONTENTS

SAFETY PRECAUTIONS ..... A-1
CONDITIONS OF USE FOR THE PRODUCT ..... A-2
REVISIONS ..... A-3
INTRODUCTION ..... A-6
CONTENTS ..... A - 6
MANUALS ..... A-14

1. OVERVIEW ..... 1-1 to $1-6$
1.1 Purpose of This Manual ..... 1-2
1.2 Terms ..... 1-5
2. INSTRUCTION TABLES ..... 2-1 to 2-48
2.1 Types of Instructions ..... 2-2
2.2 How to Read Instruction Tables ..... 2-4
2.3 Sequence Instructions ..... 2-5
2.3.1 Contact instructions ..... 2-5
2.3.2 Bond instructions ..... 2-6
2.3.3 Output instructions ..... 2-6
2.3.4 Shift instructions ..... 2-7
2.3.5 Master control instructions ..... 2-7
2.3.6 End instructions. ..... 2-7
2.3.7 Other instructions ..... 2-7
2.4 Basic Instructions ..... 2-8
2.4.1 Comparison operation instructions ..... 2-8
2.4.2 Arithmetic operation instructions ..... 2-13
2.4.3 Data conversion instructions ..... 2-16
2.4.4 Data transfer instructions ..... 2-18
2.4.5 Program branch instructions ..... 2-19
2.4.6 Program execution control instructions ..... 2-19
2.4.7 I/O refresh instructions ..... 2-19
2.4.8 Other convenient instructions ..... 2-20
2.5 Application Instructions ..... 2-21
2.5.1 Logical operation instructions. ..... 2-21
2.5.2 Rotation instructions. ..... 2-22
2.5.3 Shift instructions ..... 2-23
2.5.4 Bit processing instructions ..... 2-24
2.5.5 Data processing instructions ..... 2-25
2.5.6 Structured instructions ..... 2-27
2.5.7 Data table operation instructions ..... 2-27
2.5.8 Buffer memory access instructions ..... 2-28
2.5.9 Display instructions ..... 2-28
2.5.10 Debug/error diagnostics instructions ..... 2-29
2.5.11 String processing instructions ..... 2-30
2.5.12 Special function instructions ..... 2-33
2.5.13 Data control instructions ..... 2-36
2.5.14 File register switching instructions ..... 2-38
2.5.15 Clock instructions ..... 2-38
2.5.16 Extended clock instructions ..... 2-41
2.5.17 Program control instructions ..... 2-41
2.5.18 Other instructions ..... 2-42
2.6 Data Link Instructions ..... 2-44
2.6.1 Network refresh instructions ..... 2-44
2.6.2 Reading/Registering routing information ..... 2-44
2.6.3 Refresh device write/read instructions ..... 2-44
2.7 Multiple CPU Dedicated Instructions ..... 2-45
2.7.1 Writing data to host CPU shared memory. ..... 2-45
2.7.2 Reading data from other CPU shared memory ..... 2-45
2.8 Multiple CPU High Speed Transmission Dedicated Instructions ..... 2-46
2.8.1 Multiple CPU high speed transmission dedicated instructions ..... 2-46
2.9 Redundant System Instructions (For redundant CPU) ..... 2-47
2.9.1 Redundant system instructions (For redundant CPU) ..... 2-47
3. CONFIGURATION OF INSTRUCTIONS ..... 3-1 to 3-22
3.1 Configuration of Instructions ..... 3-2
3.2 Precautions on Programming ..... 3-4
3.3 Executing Conditions for Instructions ..... 3-13
3.4 Operation of OUT Instructions, SET/RST Instructions, or PLS/PLF Instructions Using Same Device3-14
3.5 Precautions on Using File Registers ..... 3-19
4. HOW TO READ INSTRUCTIONS ..... 4-1 to 4-4
5. SEQUENCE INSTRUCTIONS ..... 5-1 to 5-64
5.1 Contact Instructions ..... 5-2
5.1.1 Operation start, series connection, parallel connection ..... 5-2
5.1.2 Edge operation start, edge series connection, edge parallel connection. ..... 5-5
5.1.3 Negated edge operation start, negated edge series connection, negated edge pulse parallel connection ..... 5-8
5.2 Bond Instructions ..... 5-12
5.2.1 Ladder block series connection and parallel connection. ..... 5-12
5.2.2 Operation result push/read/pop. ..... 5-14
5.2.3 Operation result inversion ..... 5-17
5.2.4 Pulse conversion of operation result ..... 5-19
5.2.5 Pulse conversion of edge relay operation result ..... 5-21
5.3 Output Instructions ..... 5-24
5.3.1 Out (excluding timers, counters and annunciators) ..... 5-24
5.3.2 Timer ..... 5-26
5.3.3 Counter ..... 5-30
5.3.4 Annunciator output ..... 5-33
5.3.5 Setting devices (excluding annunciators) ..... 5-35
5.3.6 Resetting devices (excluding annunciators) ..... 5-38
5.3.7 Setting and resetting annunciators ..... 5-41
5.3.8 Rising edge and falling edge outputs ..... 5-44
5.3.9 Bit device output inversion ..... 5-47
5.3.10 Pulse conversion of direct output ..... 5-49
5.4 Shift Instructions ..... 5-51
5.4.1 Bit device shift ..... 5-51
5.5 Master Control Instructions ..... 5-55
5.5.1 Setting and resetting master control ..... 5-55
5.6 End Instructions ..... 5-60
5.6.1 Ending main routine program ..... 5-60
5.7 Other Instructions ..... 5-62
5.7.1 Sequence program stop ..... 5-62
6. BASIC INSTRUCTIONS ..... 6-1 to 6-196
6.1 Comparison Operation Instructions ..... 6-2
6.1.1 16-bit BIN data comparison ..... 6-2
6.1.2 32-bit BIN data comparison ..... 6-5
6.1.3 Floating-point data comparison (single precision) ..... 6-9
6.1.4 Floating-point data comparison (double precision) ..... 6-13
6.1.5 Character string data comparison ..... 6-17
6.1.6 16-bit BIN block data comparison ..... 6-21
6.1.7 32-bit BIN block data comparison ..... 6-25
6.2 Arithmetic Operation Instructions ..... 6-29
6.2.1 16-bit BIN data addition and subtraction ..... 6-29
6.2.2 32-bit BIN data addition and subtraction ..... 6-31
6.2.3 16-bit BIN data multiplication and division ..... 6-33
6.2.4 32-bit BIN data multiplication and division ..... 6-36
6.2.5 4-digit BCD data addition and subtraction ..... 6-39
6.2.6 8-digit BCD data addition and subtraction ..... 6-41
6.2.7 4-digit BCD data multiplication and division ..... 6-43
6.2.8 8-digit BCD data multiplication and division ..... 6-45
6.2.9 Floating-point data addition and subtraction (single precision) ..... 6-48
6.2.10 Floating-point data addition and subtraction (double precision). ..... 6-51
6.2.11 Floating-point data multiplication and division (single precision) ..... 6-54
6.2.12 Floating-point data multiplication and division (double precision) ..... 6-57
6.2.13 16-bit BIN block data addition and subtraction ..... 6-60
6.2.14 32-bit BIN block data addition and subtraction. ..... 6-63
6.2.15 Character string data concatenation ..... 6-67
6.2.16 16-bit BIN data increment and decrement ..... 6-69
6.2.17 32-bit BIN data increment and decrement ..... 6-72
6.3 Data Conversion Instructions ..... 6-74
6.3.1 BIN data to 4-/8-digit BCD data conversion ..... 6-74
6.3.2 4-/8-digit BCD data to BIN data conversion ..... 6-77
6.3.3 16-/32-bit BIN data to floating-point data conversion (single precision) ..... 6-80
6.3.4 16-/32-bit BIN data to floating-point data conversion (double precision) ..... 6-83
6.3.5 Floating-point data to 16-/32-bit BIN data conversion (single precision) ..... 6-86
6.3.6 Floating-point data to 16-bit/32-bit BIN data conversion (double precision) ..... 6-89
6.3.7 16-bit BIN data to 32-bit BIN data conversion ..... 6-92
6.3.8 32-bit BIN data to 16-bit BIN data conversion ..... 6-94
6.3.9 16-/32-bit BIN data to Gray code conversion ..... 6-96
6.3.10 Gray code to $16-/ 32$-bit BIN data conversion ..... 6-98
6.3.11 Two's complement of 16-/32-bit BIN data (sign inversion) ..... 6-100
6.3.12 Sign inversion of floating-point data (single precision) ..... 6-102
6.3.13 Sign inversion of floating-point data (double precision) ..... 6-104
6.3.14 16-bit BIN block data to 4-digit BCD block data conversion ..... 6-106
6.3.15 4-digit BCD block data to 16 -bit BIN block data conversion ..... 6-109
6.3.16 Single-precision to double-precision conversion ..... 6-112
6.3.17 Double-precision to single-precision conversion ..... 6-114
6.4 Data Transfer Instructions ..... 6-116
6.4.1 16-/32-bit data transfer ..... 6-116
6.4.2 Floating-point data transfer (single precision) ..... 6-119
6.4.3 Floating-point data transfer (double precision) ..... 6-121
6.4.4 Character string data transfer ..... 6-123
6.4.5 16-/32-bit data negation transfer. ..... 6-126
6.4.6 16-bit block data transfer. ..... 6-131
6.4.7 Identical 16-bit block data transfer ..... 6-135
6.4.8 Identical 32-bit block data transfer ..... 6-138
6.4.9 16-/32-bit data exchange ..... 6-141
6.4.10 16-bit block data exchange ..... 6-144
6.4.11 Upper and lower bytes exchange ..... 6-147
6.5 Program Branch Instructions ..... 6-149
6.5.1 Pointer branch ..... 6-149
6.5.2 Jump to END processing ..... 6-154
6.6 Program Execution Control Instructions ..... 6-156
6.6.1 Interrupt disable/enable, interrupt program mask ..... 6-156
6.6.2 Recovery from interrupt programs ..... 6-165
6.7 I/O Refresh Instructions ..... 6-168
6.7.1 I/O refresh ..... 6-168
6.8 Other Convenient Instructions ..... 6-170
6.8.1 Single-phase input up/down counter ..... 6-170
6.8.2 Two-phase input up/down counter ..... 6-173
6.8.3 Teaching timer ..... 6-176
6.8.4 Special function timer ..... 6-178
6.8.5 Rotary table shortest direction control ..... 6-181
6.8.6 Ramp signal ..... 6-184
6.8.7 Pulse density measurement ..... 6-187
6.8.8 Fixed cycle pulse output ..... 6-189
6.8.9 Pulse width modulation ..... 6-191
6.8.10 Matrix input. ..... 6-193
7. APPLICATION INSTRUCTIONS7-1 to 7-490
7.1 Logical Operation Instructions ..... 7-2
7.1.1 Logical AND operation on 16-/32-bit data ..... 7-3
7.1.2 Logical AND operation on block data ..... 7-6
7.1.3 Logical OR operation on 16-/32-bit data ..... 7-9
7.1.4 Logical OR operation on block data ..... 7-12
7.1.5 Exclusive OR operation on 16-/32-bit data ..... 7-15
7.1.6 Exclusive OR operation on block data ..... 7-18
7.1.7 Exclusive NOR operation on 16-/32-bit data ..... 7-21
7.1.8 Exclusive NOR operation on block data ..... 7-24
7.2 Rotation Instructions ..... 7-27
7.2.1 Right rotation of 16-bit data ..... 7-27
7.2.2 Left rotation of 16-bit data ..... 7-31
7.2.3 Right rotation of 32-bit data ..... 7-35
7.2.4 Left rotation of 32-bit data ..... 7-38
7.3 Shift Instructions ..... 7-41
7.3.1 n-bit right/left shift of 16-bit data ..... 7-41
7.3.2 $\quad$ 1-bit right/left shift of $n$-bit data ..... 7-44
7.3.3 $n$-bit right/left shift of $n$-bit data ..... 7-47
7.3.4 $\quad$ 1-word right/left shift of $n$-word data ..... 7-50
7.3.5 $n$-word right/left shift of $n$-word data ..... 7-53
7.4 Bit Processing Instructions ..... 7-56
7.4.1 Bit set and reset of word devices ..... 7-56
7.4.2 Bit test ..... 7-59
7.4.3 Batch reset of bit devices ..... 7-62
7.5 Data Processing Instructions ..... 7-64
7.5.1 16-/32-bit data search ..... 7-64
7.5.2 16-/32-bit data bit check ..... 7-68
7.5.3 Decoding from 8 to 256 bits ..... 7-71
7.5.4 Encoding from 256 to 8 bits ..... 7-75
7.5.5 $\quad$ 7-segment decode ..... 7-78
7.5.6 4-bit separation of 16-bit data ..... 7-81
7.5.7 4-bit connection of 16-bit data ..... 7-83
7.5.8 Separation and connection of random data ..... 7-85
7.5.9 Separation and connection of data in units of bytes ..... 7-90
7.5.10 Maximum value search of $16-/ 32$-bit data ..... 7-94
7.5.11 Minimum value search of 16 -/32-bit data ..... 7-97
7.5.12 Sorting 16-/32-bit data ..... 7-100
7.5.13 Total calculation of 16-bit data ..... 7-105
7.5.14 Total calculation of 32-bit data ..... 7-107
7.5.15 Average calculation of 16-/32-bit data ..... 7-109
7.6 Structured Instructions ..... 7-112
7.6.1 FOR to NEXT instruction loop ..... 7-112
7.6.2 Forced termination of FOR to NEXT instruction loop ..... 7-115
7.6.3 Subroutine program call ..... 7-118
7.6.4 Return from subroutine program ..... 7-120
7.6.5 Refresh ..... 7-122
7.6.6 Selection of refresh ..... 7-125
7.6.7 Selection of refresh ..... 7-130
7.7 Data Table Operation Instructions ..... 7-132
7.7.1 Writing data to data table ..... 7-132
7.7.2 Reading oldest data from data table ..... 7-135
7.7.3 Reading newest data from data table ..... 7-138
7.7.4 Deleting/inserting data from/to data table ..... 7-141
7.8 Buffer Memory Access Instructions ..... 7-144
7.8.1 Reading 1-/2-word data from intelligent function module ..... 7-144
7.8.2 Writing 1-/2-word data to intelligent function module ..... 7-148
7.9 Display Instructions ..... 7-152
7.9.1 Printing ASCII code ..... 7-152
7.9.2 Printing comments ..... 7-156
7.9.3 Resetting error display or annunciator ..... 7-160
7.10 Debug/Error Diagnostics Instructions ..... 7-163
7.10.1 Special format error check ..... 7-163
7.10.2 Changing check format of the CHK instruction ..... 7-167
7.11 String Processing Instructions ..... 7-171
7.11.1 16-/32-bit BIN data to decimal ASCII data conversion. ..... 7-171
7.11.2 16-/32-bit BIN data to hexadecimal ASCII data conversion ..... 7-176
7.11.3 4-/8-digit BCD data to decimal ASCII data conversion ..... 7-181
7.11.4 Decimal ASCII data to 16-/32-bit BIN data conversion ..... 7-186
7.11.5 Hexadecimal ASCII data to 16-/32-bit BIN data conversion ..... 7-190
7.11.6 Decimal ASCII data to 4-/8-digit BCD data conversion ..... 7-193
7.11.7 Reading device comment data. ..... 7-196
7.11.8 Character string length detection ..... 7-200
7.11.9 16-/32-bit BIN data to character string data conversion ..... 7-202
7.11.10 Character string data to 16 -/32-bit BIN data conversion ..... 7-209
7.11.11 Floating-point data to character string data conversion ..... 7-214
7.11.12 Character string data to floating-point data conversion. ..... 7-221
7.11.13 Hexadecimal BIN data to ASCII data conversion. ..... 7-226
7.11.14 ASCII data to hexadecimal BIN data conversion ..... 7-229
7.11.15 Extraction of character string data from right/left ..... 7-232
7.11.16 Random selection and replacement in character string data ..... 7-235
7.11.17 Character string data search ..... 7-240
7.11.18 Character string data insert. ..... 7-243
7.11.19 Character string data delete ..... 7-246
7.11.20 Floating-point data to BCD format conversion ..... 7-248
7.11.21 BCD format to floating-point data conversion ..... 7-251
7.12 Special Function Instructions ..... 7-253
7.12.1 SIN operation on floating-point data (single precision) ..... 7-253
7.12.2 SIN operation on floating-point data (double precision) ..... 7-255
7.12.3 COS operation on floating-point data (single precision) ..... 7-257
7.12.4 COS operation on floating-point data (double precision) ..... 7-259
7.12.5 TAN operation on floating-point data (single precision) ..... 7-261
7.12.6 TAN operation on floating-point data (double precision) ..... 7-263
7.12.7 $\mathrm{SIN}^{-1}$ operation on floating-point data (single precision) ..... 7-265
7.12.8 $\mathrm{SIN}^{-1}$ operation on floating-point data (double precision) ..... 7-267
7.12.9 $\mathrm{COS}^{-1}$ operation on floating-point data (single precision) ..... 7-269
7.12.10 $\mathrm{COS}^{-1}$ operation on floating-point data (double precision) ..... 7-271
7.12.11 $\mathrm{TAN}^{-1}$ operation on floating-point data (single precision) ..... 7-273
7.12.12 TAN $^{-1}$ operation on floating-point data (double precision) ..... 7-275
7.12.13 Degree to radian conversion on floating-point data (single precision) ..... 7-277
7.12.14 Degree to radian conversion on floating-point data (double precision) ..... 7-279
7.12.15 Radian to degree conversion on floating-point data (single precision) ..... 7-281
7.12.16 Radian to degree conversion on floating-point data (double precision) ..... 7-283
7.12.17 Exponentiation on floating-point data (single precision) ..... 7-285
7.12.18 Exponentiation on floating-point data (double precision) ..... 7-288
7.12.19 Square root operation on floating-point data (single precision) ..... 7-291
7.12.20 Square root operation on floating-point data (double precision) ..... 7-293
7.12.21 Exponential operation on floating-point data (single precision) ..... 7-295
7.12.22 Exponential operation on floating-point data (double precision) ..... 7-298
7.12.23 Natural logarithm operation on floating-point data (single precision) ..... 7-301
7.12.24 Natural logarithm operation on floating-point data (double precision) ..... 7-303
7.12.25 Common logarithm operation on floating-point data (single precision) ..... 7-305
7.12.26 Common logarithm operation on floating-point data (double precision) ..... 7-307
7.12.27 Random number generation and series update ..... 7-309
7.12.28 Square root operation on 4-/8-digit BCD data ..... 7-311
7.12.29 SIN operation on data in BCD format ..... 7-315
7.12.30 COS operation on data in BCD format ..... 7-318
7.12.31 TAN operation on data in BCD format ..... 7-321
7.12.32 $\mathrm{SIN}^{-1}$ operation on data in BCD format ..... 7-324
7.12.33 $\mathrm{COS}^{-1}$ operation on data in BCD format ..... 7-327
7.12.34 TAN $^{-1}$ operation on data in BCD format ..... 7-330
7.13 Data Control Instructions ..... 7-332
7.13.1 Upper and lower limit controls of 16-/32-bit BIN data ..... 7-332
7.13.2 Dead band control of 16-/32-bit BIN data ..... 7-336
7.13.3 Zone control of 16-/32-bit BIN data ..... 7-340
7.13.4 Scaling (coordinate by point data) ..... 7-344
7.13.5 Scaling (coordinate by X/Y data) ..... 7-348
7.14 File Register Switching Instructions ..... 7-351
7.14.1 Switching file register block numbers ..... 7-351
7.14.2 Setting file register files ..... 7-354
7.14.3 Setting files for comments ..... 7-357
7.15 Clock Instructions ..... 7-360
7.15.1 Reading clock data ..... 7-360
7.15.2 Writing clock data ..... 7-363
7.15.3 Clock data addition ..... 7-366
7.15.4 Clock data subtraction ..... 7-369
7.15.5 Time data conversion (hour/minute/second format to seconds) ..... 7-372
7.15.6 Time data conversion (seconds to hour/minute/second format) ..... 7-374
7.15.7 Date data comparison ..... 7-376
7.15.8 Time data comparison ..... 7-381
7.16 Extended Clock Instructions ..... 7-386
7.16.1 Read extended clock data ..... 7-386
7.16.2 Addition of extended clock data ..... 7-389
7.16.3 Subtraction of extended clock data ..... 7-392
7.17 Program Control Instructions ..... 7-395
7.17.1 Program standby ..... 7-395
7.17.2 Program output OFF standby. ..... 7-397
7.17.3 Registering program as scan execution type ..... 7-399
7.17.4 Registering program as low-speed execution type ..... 7-401
7.17.5 Checking program execution status ..... 7-403
7.18 Other Instructions ..... 7-406
7.18.1 Resetting watchdog timer ..... 7-406
7.18.2 Timing pulse generation ..... 7-408
7.18.3 Time check ..... 7-410
7.18.4 Reading 1 byte directly from file register ..... 7-412
7.18.5 Writing 1 byte directly to file register ..... 7-415
7.18.6 Reading indirect address ..... 7-418
7.18.7 Numeric input from keyboard ..... 7-419
7.18.8 Batch save and recovery of index registers ..... 7-423
7.18.9 Reading module information ..... 7-426
7.18.10 Reading module type ..... 7-432
7.18.11 Trace set/reset ..... 7-437
7.18.12 Writing data to specified file ..... 7-440
7.18.13 Reading data from specified file ..... 7-452
7.18.14 Writing data to standard ROM ..... 7-466
7.18.15 Reading data from standard ROM ..... 7-469
7.18.16 Program load from memory card ..... 7-471
7.18.17 Program unload from program memory ..... 7-475
7.18.18 Load and unload ..... 7-478
7.18.19 File register high-speed block transfer ..... 7-481
7.18.20 User message instruction ..... 7-486
8. DATA LINK INSTRUCTIONS ..... 8-1 to 8-34
8.1 Network Refresh Instructions ..... 8-2
8.1.1 Refresh instruction for specified module ..... 8-2
8.2 Reading/Registering Routing Information ..... 8-7
8.2.1 Reading routing information ..... 8-7
8.2.2 Registering routing information ..... 8-9
8.3 Refresh Device Write/Read Instruction ..... 8-12
8.3.1 Refresh device write (in 1-bit units) ..... 8-12
8.3.2 Refresh device write (in 16-bit units) ..... 8-17
8.3.3 Refresh device read (in 1-bit units) ..... 8-23
8.3.4 Refresh device read (in 16-bit units) ..... 8-28
9. MULTIPLE CPU DEDICATED INSTRUCTIONS ..... 9-1 to 9-22
9.1 Writing Data to Host CPU Shared Memory ..... 9-2
9.1.1 Writing data to host CPU shared memory. ..... 9-4
9.1.2 Writing data to host CPU shared memory. ..... 9-9
9.2 Reading Data from Other CPU Shared Memory ..... 9-14
9.2.1 Reading data from other CPU shared memory ..... 9-15
10. MULTIPLE CPU HIGH SPEED TRANSMISSION DEDICATED INSTRUCTIONS ..... 10-1 to 10-20
10.1 Overview ..... 10-2
10.2 Writing Device Data to Other CPUs ..... 10-12
10.3 Reading Device Data from Other CPUs ..... 10-16
11. REDUNDANT SYSTEM INSTRUCTIONS (For REDUNDANT CPU) ..... 11-1 to 11-6
11.1 System switching instruction ..... 11-2
APPENDIX App - 1 to App - 2
Appendix 1Added/Changed Instructions with Version Upgrade of GX Works2 ..... App - 2
INDEXIndex-1 to Index-8

## Relevant manuals

The manuals related to this product are listed below.
Order each manual as needed, referring to the following lists.

## (1) Structured programming

| Manual name | Manual number <br> (Model code) |  |
| :--- | :---: | :---: |
| MELSEC-Q/L/F Structured Programming Manual (Fundamentals) <br> Methods and languages for structured programming | (Sold separately) | SH-080782ENG |
| (13JW06) |  |  |

## (2) Operation of GX Works2

| Manual name | Manual number (Model code) |
| :---: | :---: |
| GX Works2 Version 1 Operating Manual (Common) <br> System configuration, parameter settings, and online operations of GX Works2, which are common to Simple projects and Structured projects <br> (Sold separately) | SH-080779ENG <br> (13JU63) |
| GX Works2 Version 1 Operating Manual (Structured Project) <br> Operations, such as programming and monitoring in Structured projects, of GX Works2 | SH-080781ENG <br> (13JU65) |
| GX Works2 Beginner's Manual (Structured Project) <br> Basic operations, such as programming, editing, and monitoring in Structured projects, of GX Works2. This manual is intended for first-time users of GX Works2. <br> (Sold separately) | $\begin{gathered} \text { SH-080788ENG } \\ (13 \mathrm{JZ23}) \end{gathered}$ |

## ®POINT

Operating manuals in PDF format are stored on the CD-ROM of the software package. Printed manuals are sold separately. To order manuals, please provide the manual number (model code) listed in the table above.

## OVERVIEW

1.1 Purpose of This Manual ..... 1-2
1.2 Terms ..... 1-5

### 1.1 Purpose of This Manual

This manual explains the common instructions used for creating structured programs. Manuals for reference are listed in the following table according to their purpose.
For information such as the contents and number of each manual, refer to the list of 'Related manuals'.
(1) Operation of GX Works2

| Purpose |  | GX Works2 Installation Instructions | GX Works2 <br> Beginner's Manual |  | GX Works2 Version 1 Operating Manual |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | Simple <br> Project | Structured Project | Common | Simple <br> Project | Structured Project | Intelligent Function Module |
| Installation | Learning the operating environment and installation method |  | Details |  |  |  |  |  |  |
|  | Learning a USB driver installation method |  |  |  | Details |  |  |  |
| Operation of GX Works2 | Learning all functions of GX Works2 |  |  |  |  |  |  |  |
|  | Learning the project types and available languages in GX Works2 |  |  |  |  |  |  |  |
|  | Learning the basic operations and operating procedures when creating a simple project for the first time |  | Details |  |  |  |  |  |
|  | Learning the basic operations and operating procedures when creating a structured project for the first time |  |  | Details |  |  |  |  |
|  | Learning the operations of available functions regardless of project type. |  |  |  |  |  |  |  |
|  | Learning the functions and operation methods for programming |  |  |  |  | Details |  |  |
|  | Learning data setting methods for intelligent function module |  |  |  |  |  |  | Details |

(2) Operations in each programming language

For details of instructions used in each programming language, refer to the section 3 on the next page.

| Purpose |  | GX Works2 <br> Beginner's Manual |  | GX Works2 Version 1 Operating Manual |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | Simple Project | Structured Project | Simple Project | Structured Project |
| Simple Project | Ladder | Outline |  | Details |  |
|  | SFC | $\overbrace{\text { Outline }}^{*_{1}}$ |  | Details |  |
|  | ST |  | Outline |  | Details |
| Structured Project | Ladder | Outline |  | Details |  |
|  | SFC | ${ }^{*} 1$ |  | Details |  |
|  | Structured ladder /FBD |  | Outline |  | Details |
|  | ST |  |  |  | Details |

*1: MELSAP3 and FX series SFC only
(3) Details of instructions in each programming language

| Purpose |  | MELSECQ/L/F Structured Programming Manual | MELSEC-Q/L Structured Programming Manual |  |  | MELSEC- <br> Q/L <br> Programming Manual | MELSEC- <br> Q/L/QnA <br> Programming Manual |  | MELSEC-Q <br> Programming/ <br> Structured <br> Programming <br> Manual | Manualfor module to be used |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | Fundamentals | Common Instructions | Special Instructions | Application Functions | Common Instruction | PID Control Instructions | SFC | Process control instructions |  |
| All languages | Learning details of programmable controller CPU error codes, special relays, and special registers |  |  |  |  |  |  |  |  |  |
| Using ladder language | Learning the types and details of common instructions |  |  |  |  | Details |  |  |  |  |
|  | Learning the types and details of instructions for intelligent function modules |  |  |  |  |  |  |  |  | Details |
|  | Learning the types and details of instructions for network modules |  |  |  |  |  |  |  |  | Details |
|  | Learning the types and details of instructions for the PID control function |  |  |  |  |  | Details |  |  |  |
|  | Learning the types and details of the process control instructions |  |  |  |  |  |  |  | Details |  |
| Using SFC language | Learning details of specifications, functions, and instructions of SFC (MELSAP3) |  |  |  |  |  |  | Details |  |  |
| Using structured ladder/FBD/ ST language | Learning the fundamentals for creating a structured program | Details |  |  |  |  |  |  |  |  |
|  | Learning the types and details of common instructions |  | Details |  |  |  |  |  |  |  |
|  | Learning the types and details of instructions for intelligent function modules |  |  |  |  |  |  |  |  | Details |
|  | Learning the types and details of instructions for network modules |  |  |  |  |  |  |  |  | Details |
|  | Learning the types and details of instructions for the PID control function |  |  |  |  |  | Details |  |  |  |
|  | Learning the types and details of application functions |  |  |  | Details |  |  |  |  |  |
|  | Learning the types and details of the process control instructions |  |  |  |  |  |  |  | Details |  |

[^0]This manual uses the generic terms and abbreviations listed in the following table to discuss the software packages and programmable controller CPUs. Corresponding module models are also listed if needed.

| Term | Description |
| :---: | :---: |
| GX Works2 | Product name for the MELSEC programmable controller software package |
| Basic model QCPU | A generic term for Q00JCPU, Q00CPU, and Q01CPU |
| High Performance model QCPU | A generic term for Q02CPU, Q02HCPU, Q06HCPU, Q12HCPU, and Q25HCPU |
| Process CPU | A generic term for the Q02PHCPU, Q06PHCPU, Q12PHCPU, and Q25PHCPU |
| Redundant CPU | A generic term for the Q12PRHCPU and Q25PRHCPU |
| Universal model QCPU | A generic term for Q00UJCPU, Q00UCPU, Q01UCPU, Q02UCPU, Q03UDCPU, Q03UDVCPU, Q03UDECPU, Q04UDHCPU, Q04UDVCPU, Q04UDEHCPU, Q06UDHCPU, Q06UDVCPU, Q06UDEHCPU, Q10UDHCPU, Q10UDEHCPU, Q13UDHCPU, Q13UDVCPU, Q13UDEHCPU, Q20UDHCPU, Q20UDEHCPU, Q26UDHCPU, Q26UDVCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU |
| Built-in Ethernet port QCPU | A generic term for Q03UDVCPU, Q03UDECPU, Q04UDVCPU, Q04UDEHCPU, Q06UDVCPU, Q06UDEHCPU, Q10UDEHCPU, Q13UDVCPU, Q13UDEHCPU, Q20UDEHCPU, Q26UDVCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU |
| High-speed Universal model QCPU | A generic term for the Q03UDVCPU, Q04UDVCPU, Q06UDVCPU, Q13UDVCPU, and Q26UDVCPU |
| Built-in Ethernet port LCPU | A generic term for the L02CPU, L02CPU-P, L06CPU, L06CPU-P, L26CPU, L26CPU-P, L26CPU-BT, and L26CPU-PBT |
| QCPU (Q mode) | A generic term for the Basic model QCPU, High Performance model QCPU, Process CPU, Redundant CPU, and Universal model QCPU |
| LCPU | A generic term for the L02SCPU, L02SCPU-P, L02CPU, L02CPU-P, L06CPU, L06CPU-P, L26CPU, L26CPU-P, L26CPU-BT, and L26CPU-PBT |
| CPU module | A generic term for QCPU (Q mode) and LCPU |
| QnCPU | A generic term for Q00JCPU, Q00CPU, Q01CPU, and Q02CPU |
| QnHCPU | A generic term for Q02HCPU, Q06HCPU, Q12HCPU, and Q25HCPU |
| QnPHCPU | A generic term for the Q02PHCPU, Q06PHCPU, Q12PHCPU, and Q25PHCPU |
| QnPRHCPU | A generic term for the Q12PRHCPU and Q25PRHCPU |
| QnUCPU | A generic term for Q00UJCPU, Q00UCPU, Q01UCPU, Q02UCPU, Q03UDCPU, Q03UDVCPU, Q03UDECPU, Q04UDHCPU, Q04UDVCPU, Q04UDEHCPU, Q06UDHCPU, Q06UDVCPU, Q06UDEHCPU, Q10UDHCPU, Q10UDEHCPU, Q13UDHCPU, Q13UDVCPU, Q13UDEHCPU, Q20UDHCPU, Q20UDEHCPU, Q26UDHCPU, Q26UDVCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU |
| QnU(D)(H)CPU | A generic term for Q02UCPU, Q03UDCPU, Q04UDHCPU, Q06UDHCPU, Q10UDHCPU, Q13UDHCPU, Q20UDHCPU, and Q26UDHCPU |
| QnUD(H)CPU | A generic term for Q03UDCPU, Q04UDHCPU, Q06UDHCPU, Q10UDHCPU, Q13UDHCPU, Q20UDHCPU, and Q26UDHCPU |
| QnUDVCPU | A generic term for the Q03UDVCPU, Q04UDVCPU, Q06UDVCPU, Q13UDVCPU, and Q26UDVCPU |
| QnUDE(H)CPU | A generic term for Q03UDECPU, Q04UDEHCPU, Q06UDEHCPU, Q10UDEHCPU, Q13UDEHCPU, Q20UDEHCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU |
| CC-Link IE | A generic term for CC-Link IE Controller Network and CC-Link IE Field Network |
| MELSECNET/H | The abbreviation for the MELSECNET/H network system |
| Personal computer | A generic term for personal computer on which Windows ${ }^{\circledR}$ operates |
| Common instruction | A generic term for the sequence instructions, basic instructions, application instructions, data link instructions, multiple CPU dedicated instructions, multiple CPU high-speed transmission dedicated instructions, and redundant system instructions |
| Special instruction | A generic term for module dedicated instructions, PID control instructions, socket communication function instructions, built-in I/O function instructions, and data logging function instructions |
| Application function | A generic term for functions defined in IEC61131-3 (functions and function blocks) <br> (It is executed by combinations of multiple common instructions in a programmable controller.) |

MEMO
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
$\qquad$
2.1 Types of Instructions ..... 2-2
2.2 How to Read Instruction Tables ..... 2-4
2.3 Sequence Instructions ..... 2-5
2.4 Basic Instructions ..... 2-8
2.5 Application Instructions ..... 2-21
2.6 Data Link Instructions ..... 2-44
2.7 Multiple CPU Dedicated Instructions ..... 2-45
2.8 Multiple CPU High Speed Transmission Dedicated Instructions ..... 2-46

### 2.1 Types of Instructions

Instructions for CPU modules are classified into sequence instructions, basic instructions, application instructions, data link instruction, multiple CPU dedicated instructions, multiple CPU high speed transmission dedicated instructions, and redundant system instructions. Table 2.1 shows the types of instructions.

Table 2.1 Types of instructions

| Type of instruction |  | Description | Reference |
| :---: | :---: | :---: | :---: |
| Sequence instruction | Contact instruction | Operation start, series connection, parallel connection | Chapter 5 |
|  | Bond instruction | Ladder block connection, operation result pulse, operation result store/read |  |
|  | Output instruction | Bit device output, pulse output, output inversion |  |
|  | Shift instruction | Bit device shift |  |
|  | Master control instruction | Master control |  |
|  | End instruction | Program conclusion |  |
|  | Other instructions | Instructions not classified into the above types, such as program stop and no operation |  |
| Basic instruction | Comparison operation instruction | Comparison such as $=,>$, and $<$ | Chapter 6 |
|  | Arithmetic operation instruction | Addition, subtraction, multiplication, and division of BIN, BCD, floating-point data or string |  |
|  | Data conversion instruction | Conversion from BIN to BCD data and vice versa, conversion from BIN data to floating-point data and vice versa |  |
|  | Data transfer instruction | Specified data transfer |  |
|  | Program branch instruction | Program jump |  |
|  | Program execution control instruction | Enable/disable interrupt programs |  |
|  | I/O refresh instruction | Partial refresh execution |  |
|  | Other convenient instructions | Instructions such as up/down counter, teaching timer, special function timer, and rotary table shortest direction control |  |
| Application instruction | Logical operation instruction | Logical operations such as logical OR and logical AND | Chapter 7 |
|  | Rotation instruction | Specified data rotation |  |
|  | Shift instruction | Specified data shift |  |
|  | Bit processing instruction | Bit set/reset, bit test, bit device batch reset |  |
|  | Data processing instruction | Data processes such as 16-bit data search, decode, and encode |  |
|  | Structured instruction | Repeated operation, subroutine program call, selection of refresh, fixed index setting |  |
|  | Data table operation instruction | Data read/write from/to data tables |  |
|  | Buffer memory access instruction | Data read/write from/to intelligent function module |  |
|  | Display instruction | ASCII code print, reset the annunciator |  |
|  | Debug/error diagnostics instruction | Special format error check, changing check format of the CHK instruction |  |
|  | String processing instruction | Conversion from BIN/BCD data to ASCII data and vice versa, conversion from BIN data to character string data and vice versa, conversion from floating-point data to character string data and vice versa, character string process |  |
|  | Special function instruction | Trigonometric functions, conversion from degree to radian and vice versa, exponential operation, natural logarithm, common logarithm, square root |  |
|  | Data control instruction | Upper/lower limit control, dead band control, zone control, scaling |  |
|  | File register switching instruction | File register block numbers switch, file register/comment file specification |  |
|  | Clock instruction | Clock data (year, month, day, hour, minute, second, and day of week) read/write, clock data (hour, minute, and second) addition/subtraction, time data conversion from hour/minute/second format to seconds and vice versa, date (year, month, and day) comparison, clock data (hour, minute, and second) comparison |  |
|  | Extended Clock Instruction | Clock data (year, month, day, hour, minute, second, day of week, and millisecond) read/write, clock data (hour, minute, second, and millisecond) addition/subtraction |  |
|  | Program control instruction | Switch instruction of program executing conditions |  |
|  | Other instructions | Instructions not classified into the above types, such as WDT reset and timing clock |  |

Table 2.1 Types of instructions (continued)

| Type of instruction |  | Description | Reference |
| :--- | :--- | :--- | :--- |
| Data Link <br> instruction | Network link refresh instruction | Specified network module refresh | Chapter 8 |
|  | Routing information read/ <br> register instruction | Routing data read/register | Refresh device write/read <br> instruction |
|  | Chapter 9 |  |  |
| MultipleCPU <br> dedicated <br> instruction | Multiple CPU dedicated <br> instruction | Data write to host CPU shared memory, data read from other CPU shared memory | Chapter 10 |
| Multiple CPU <br> high speed <br> transmission <br> dedicated <br> instruction | Multiple CPU device data write/ <br> read instruction | Device data write to other CPU, device data read from other CPU |  |
| Redundant <br> system <br> instruction | Redundant CPU dedicated <br> instruction | System switching | Chapter 11 |

### 2.2 How to Read Instruction Tables

Instruction tables in Sections 2.3 to 2.8 are summarized in the following form.
Table 2.2 How to read Instruction tables

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Jump | CJ | (D) | - Jumps to (iD) when the input condition is satisfied. | $\sqrt{\square}$ | 6-148 |
|  | SCJ | (b) | - Jumps to ([0) from the next scan after the input condition is satisfied. |  |  |
|  | JMP | (D) | - Unconditionally jumps to (11). |  |  |
| 4 | ¢ | 4 | ¢ | ¢ | ¢ |

## Description

1) Classifies instructions by application.
2) Indicates the instructions used in a programs.
3) Indicates the arguments of the instruction.
(®, (sl) Source.............. Stores data before operation.
(d), (11) Destination ....... Indicates the destination of data after operation.
$\mathrm{n}, \mathrm{n}$ $\qquad$ Specifies the number of devices and the number of transfers.
p $\qquad$ Specifies the pointer number.
4) Indicates the processing details of each instruction.
5) Details of executing conditions of each instruction are as follows:

Symbol \begin{tabular}{l}
\multicolumn{1}{c}{ Executing condition } <br>
No symbol <br>

| Indicates a non-conditional executing instruction that is always executed regardless of the ON/OFF status of |
| :--- |
| the precondition of the instruction. |
| When the precondition is OFF, the instruction performs 'OFF' processing. | <br>

\hline

 

Indicates an 'executed while ON' type instruction that is executed only while the precondition is ON. <br>
When the precondition is OFF, the instruction is not executed and does not perform processing.
\end{tabular}

6) Indicates the pages on which the instructions are explained.

### 2.3 Sequence Instructions

### 2.3.1 Contact instructions

Table 2.3 Contact instructions


### 2.3.2 Bond instructions

Table 2.4 Bond instructions

| Category | Instruction name | Symbol used in structured ladder/FBD | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Bond | ANB |  | - Performs a logical AND operation between logical blocks <br> (Series connection between logical blocks) |  | 5-12 |
|  | ORB |  | - Performs a logical OR operation between logical blocks <br> (Parallel connection between logical blocks) |  |  |
|  | MPS |  | - Stores operation results |  | 5-14 |
|  | MRD |  | - Reads operation results stored in the MPS instruction |  |  |
|  | MPP |  | - Reads and resets operation results stored in the MPS instruction |  |  |
|  | INV | - | - Inverts operation results |  | 5-17 |
|  | MEP | - | - Outputs a pulse at a rising edge of operation result |  | 5-19 |
|  | MEF | - | - Outputs a pulse at a falling edge of operation result |  |  |
|  | EGP | - | - Outputs a pulse at a rising edge of operation result |  | 5-21 |
|  | EGF | - | - Outputs a pulse at a falling edge of operation result |  |  |

### 2.3.3 Output instructions

Table 2.5 Output instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Output | OUT | (d) | - Outputs devices <br> (excluding timers, counters and annunciators) |  | 5-24 |
|  | OUT_T | (51), (32), (d) | - Outputs devices (Timer) |  | 5-26 |
|  | OUTH_T |  |  |  |  |
|  | OUT_C | (31), (32), (d) | - Outputs devices (Counter) |  | 5-30 |
|  | OUT | (d) | - Outputs devices (Annunciator output) |  | 5-33 |
|  | SET | (d) | - Sets devices |  | $\begin{aligned} & 5-35 \\ & 5-41 \end{aligned}$ |
|  | RST | (d) | - Resets devices |  | $\begin{aligned} & 5-38 \\ & 5-41 \end{aligned}$ |
|  | PLS | (d) | - Generates a pulse for one program cycle at the rising of the input signal | $\uparrow$ | 4 |
|  | PLF | (d) | - Generates a pulse for one program cycle at the falling of the input signal |  | 5-44 |
|  | FF | (5) | - Inverts device outputs | $\uparrow$ | 5-47 |
|  | DELTA | (d) | - Direct output pulse |  | 5-49 |
|  | DELTAP | (d) |  |  |  |

### 2.3.4 Shift instructions

Table 2.6 Shift instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Shift | SFT | (d) | - 1-bit shift of a device | $\sqrt{\square}$ | 5-51 |
|  | SFTP |  |  | $\uparrow$ |  |

### 2.3.5 Master control instructions

Table 2.7 Master control instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Master control | MC | n*, (d) | - Starts master control |  | 5-55 |
|  | MCR | n* | - Resets master control |  |  |

### 2.3.6 End instructions

Table 2.8 End instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Program end | FEND | - | - Ends main program |  | 5-60 |

2.3.7 Other instructions

Table 2.9 Other instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Stop | STOP | - | - Stops sequence program operations after the input condition is satisfied <br> - Executes the sequence program when the RUN/STOP (key) is switched back to the RUN position | $\sqrt{\square}$ | 5-62 |

### 2.4 Basic Instructions

### 2.4.1 Comparison operation instructions

Table 2.10 Comparison operation instructions


Table 2.10 Comparison operation instructions (continued)

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 32-bit BIN data comparison | LDD= | (31), (2) | - Conduction state when $((\sqrt{1})+1,(\Im 1)=(\Omega 2)+1, \text { (®2) })$ <br> - Non-conduction state when $\left.((11)+1, \text { (s1) }) \neq\left({ }^{(\Omega 2}\right)+1, \text { (®2) }\right)$ |  | 6-5 |
|  | ANDD= | (1), (2) |  |  |  |
|  | ORD= | (31), (2) |  |  |  |
|  | LDD<> | (1), (2) | - Conduction state when $\left((11)+1,(\Im 1) \neq\left({ }^{(\Omega 2}\right)+1, \text { (®2) }\right)$ <br> - Non-conduction state when $((1)+1,(\Im 1))=(\text { (s2) }+1,(\Omega 2)$ |  |  |
|  | ANDD<> | (1), (2) |  |  |  |
|  | ORD<> | (31), (3) |  |  |  |
|  | LDD<= | (31), (2) | - Conduction state when $\left.(ङ 1)+1,(\Im 1) \leqq(® 2)+1, \Omega_{2}\right)$ <br> - Non-conduction state when $((11)+1,(\Im 1))>(\S 2)+1,(\Omega 2)$ |  |  |
|  | ANDD<= | (51), (2) |  |  |  |
|  | ORD<= | (1), (2) |  |  |  |
|  | LDD< | (1), (3) | - Conduction state when $((11)+1, \text { (§1) })<(® 2)+1, \text { (®2) })$ <br> - Non-conduction state when |  |  |
|  | ANDD< | (31), (2) |  |  |  |
|  | ORD< | (1), (2) |  |  |  |
|  | LDD>= | (31), (3) | - Conduction state when $((11)+1,(\Im 1) \geqq(® 2)+1, \text { (®2) })$ <br> - Non-conduction state when |  |  |
|  | ANDD>= | (1), (2) |  |  |  |
|  | ORD>= | (1), (3) |  |  |  |
|  | LDD> | (31), (3) | - Conduction state when $((1)+1, \text { (s1) })>\left(\text { ®2 }^{2}+1, \text { (s2 }\right)$ <br> - Non-conduction state when $((11)+1, \text { (s1) }) \leqq(\text { (®2) }+1, \text { (®2) })$ |  |  |
|  | ANDD> | (1), (2) |  |  |  |
|  | ORD> | (1), (3) |  |  |  |
| Floating-point data comparison (single precision) | LDE= | (1), (2) | - Conduction state when <br> - Non-conduction state when $((11)+1, \text { (s1) }) \neq(\text { (®2) }+1, \text { (®2) })$ |  | 6-9 |
|  | ANDE= | (1), (2) |  |  |  |
|  | ORE= | (31), (2) |  |  |  |
|  | LDE<> | (31), (2) | - Conduction state when <br> - Non-conduction state when $\left(\Im_{1}+1,(\Im 1)=\left({ }^{(\Omega 2}\right)+1, \Im_{2}\right)$ |  |  |
|  | ANDE<> | (1), (2) |  |  |  |
|  | ORE<> | (31), (2) |  |  |  |
|  | LDE<= | (31), (2) | - Conduction state when $((1)+1, \text { (s1) }) \leqq\left(\text { ®2 }^{2}+1, \text { (®2) }\right)$ <br> - Non-conduction state when $((11)+1,(\Im 1))>(® 2)+1, \text { (®2) })$ |  |  |
|  | ANDE<= | (1), (2) |  |  |  |
|  | ORE<= | (1), (3) |  |  |  |
|  | LDE< | (11), (2) | - Conduction state when $((11)+1, \text { (s1) })<(® 2)+1, \text { (s2) })$ <br> - Non-conduction state when $(\text { ®1 } 1+1, \text { (s1) }) \geqq(\text { (®2) }+1, \text { (®2) }$ |  |  |
|  | ANDE< | (51), (2) |  |  |  |
|  | ORE< | (31), (2) |  |  |  |
|  | LDE>= | (1), (3) | - Conduction state when $((1)+1, \text { (s1) }) \geqq(® 2)+1, \text { (®2) })$ <br> - Non-conduction state when $(\text { (1) }+1, \text { (1) })<(\text { (2) })+1, \text { (2) })$ |  |  |
|  | ANDE>= | (1), (2) |  |  |  |
|  | ORE>= | (1), (2) |  |  |  |
|  | LDE> | (31), (3) | - Conduction state when $((11)+1, \text { (s1) })>(\text { (®2 })+1, \text { (®2) })$ <br> - Non-conduction state when $\left.()_{1}+1, \text { (11) }\right) \leqq(\S 2)+1, \text { (®2) }$ |  |  |
|  | ANDE> | (1), (2) |  |  |  |
|  | ORE> | (1), (3) |  |  |  |

Floating-point data comparison (single precision)

Table 2.10 Comparison operation instructions (continued)


Table 2.10 Comparison operation instructions (continued)

*2 : The following shows the comparison conditions when character strings are compared.

- Match condition: All character strings are matched.
- Conditions for larger character string: For different character strings, character strings with larger character codes.
For different character string lengths, larger character string lengths.
- Conditions for smaller character string:For different character strings, character strings with smaller character codes.
For different character string lengths, smaller character string lengths.

Table 2.10 Comparison operation instructions (continued)

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 16-bit BIN block data comparison | BKCMP= | (1), (3), n , (1) | - Compares the 16 -bit BIN data $n$ points from the device specified for (st) or constant with the 16 -bit BIN data $n$ points from the device specified for ${ }^{2}$, , and stores the result to (d) and the following devices. | $\sqrt{L}$ | 6-21 |
|  | BKCMP<> | (1), (22), n , (d) |  |  |  |
|  | BKCMP $<=$ | (1), (2), n, (1) |  |  |  |
|  | BKCMP< | (1), (2), n , (d) |  |  |  |
|  | BKCMP>= | (1), (2), n, (d) |  |  |  |
|  | BKCMP> | (1), (3), n, (d) |  |  |  |
|  | BKCMP $=$ P | (1), (2), n, (d) |  |  |  |
|  | BKCMP<>P | (1), (2), n , (d) |  |  |  |
|  | BKCMP<=P | (1), (2), n, (d) |  |  |  |
|  | BKCMP<P | (1), (2), n, (d) |  |  |  |
|  | BKCMP>=P | (1), (2), n, (d) |  |  |  |
|  | BKCMP>P | (1), (2), n, (d) |  |  |  |
| 32-bit BIN block data comparison | DBKCMP= | (1), (2), n, (1) | - Compares the 32-bit BIN data $n$ points from the device specified for ( ${ }^{(1)}$ ) or constant with the 32 -bit BIN data $n$ points from the device specified for (22), and stores the result to (d) and the following devices. | $\sqrt{\square}$ | 6-25 |
|  | DBKCMP<> | (1), (3), n, (1) |  |  |  |
|  | DBKCMP<= | (1), (3), n, (1) |  |  |  |
|  | DBKCMP< | (1), (3), n, (1) |  |  |  |
|  | DBKCMP>= | (1), (3), n, (1) |  |  |  |
|  | DBKCMP> | (15), (2), n, (d) |  |  |  |
|  | DBKCMP=P | (1), (2), n, (d) |  | $\uparrow$ |  |
|  | DBKCMP<>P | (1), (3), n, (1) |  |  |  |
|  | DBKCMP<=P | (1), (3), n, (1) |  |  |  |
|  | DBKCMP<P | (1), (2), n, (d) |  |  |  |
|  | DBKCMP>=P | (1), (3), n, ( () |  |  |  |
|  | DBKCMP>P | (1), (2), n, (d) |  |  |  |

### 2.4.2 Arithmetic operation instructions

Table 2.11 Arithmetic operation instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 16-bit BIN data addition and subtraction |  | (51), (32), (d) | - (51) + (32) $\rightarrow$ (d) |  | 6-29 |
|  | -P | (51), (32), (d) | - (31) - (32) $\rightarrow$ (d) |  |  |
| 32-bit BIN data addition and subtraction | $\begin{array}{\|l\|} \mathrm{D}+ \\ \hline \mathrm{D}+\mathrm{P} \\ \hline \end{array}$ | (51), (22), (d) | - $($ (1) +1, (s1) $)+($ (®2) +1, (2) $) \rightarrow($ (d) +1 , (d) $)$ |  | 6-31 |
|  | D- | (51), (32), (d) | -((51) +1 , (51) $)-($ (22 $)+1$, (22) $) \rightarrow($ (d) +1 , (d) $)$ |  |  |
| 16-bit BIN data multiplication and division | $\begin{array}{\|l} * \\ \hline * P \end{array}$ | (51), (22), (d) | - (51) $\times$ (32) $\rightarrow$ (d) +1 , (d) $)$ |  | 6-33 |
|  | \| $\mid / P$ | (51), (22), (d) | - (51)/(2) $\rightarrow$ Quotient (d), Remainder ( ${ }_{\text {( })+1 \text { ) }}$ |  |  |
| 32-bit BIN data multiplication and division | $\begin{array}{\|l\|} \hline D^{*} \\ \hline D^{*} P \end{array}$ | (51), (32), (d) | $\begin{aligned} & \text { • }(\text { (s1 })+1, \text { (s1) }) \times(\text { (s2 })+1, \text { (®2 }) \rightarrow(\text { (d) }+3 \text {, (d) }+2, \\ & \text { (d) }+1, \text { (d) }) \end{aligned}$ |  | 6-36 |
|  | D/ <br> D/P | (31), (22), (d) | $\cdot(\text { (s1) }+1, \text { (s1) }) /(\text { (®2) }+1, \text { (s2 }) \rightarrow \text { Quotient }(\text { (d) }+1 \text {, }$ <br> (d) ), Remainder (d) +3 , (d) +2 ) |  |  |
| 4-digit BCD data addition and subtraction | $\begin{array}{\|l} \hline B+ \\ \hline B+P \\ \hline \end{array}$ | (51), (32), (d) | - (51) + (22) $\rightarrow$ (d) | $\frac{\sqrt{\square}}{5}$ | 6-39 |
|  | B- <br> B-P | (51), (22), (d) | - (31) - (22) $\rightarrow$ (d) | $\frac{\square}{5}$ |  |
| 8-digit BCD data addition and subtraction | $\begin{aligned} & \mathrm{DB}+ \\ & \mathrm{DB}+\mathrm{P} \end{aligned}$ | (51), (32), (d) | - $($ (51) +1, (51) $)+($ (32 $)+1$, (32) $) \rightarrow($ (d) +1 , (d) $)$ | $\frac{\sqrt{4}}{5}$ | 6-41 |
|  | $\begin{array}{\|l\|} \hline \text { DB- } \\ \hline \text { DB-P } \\ \hline \end{array}$ | (51), (22), (d) | $\cdot($ (31) +1 , (s1) $)-($ (22) +1, (®2) $) \rightarrow($ (d) +1 , (d) $)$ | $\frac{\square}{5}$ |  |
| 4-digit BCD data multiplication and division | $\mathrm{B}^{*}$ $B * P$ | (31), (22), (d) | - (s1) $\times$ (32) $\rightarrow$ (d) +1 , (d) $)$ | $\frac{\sqrt{4}}{5}$ | 6-43 |
|  | B/ <br> B/P | (51), (22), (d) | -(①)/(2) $\rightarrow$ Quotient (d), Remainder ( ${ }_{\text {( })+1 \text { ) }}$ |  |  |

Table 2.11 Arithmetic operation instructions (continued)


Table 2.11 Arithmetic operation instructions (continued)

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Character string data concatenation | $\begin{array}{\|l\|} \hline \$+ \\ \hline \$+P \\ \hline \end{array}$ | (11), (2), (d) | - Connects the character string specified for <br> to the character string specified for $\qquad$ and stores the result to (d) and the following devices. | $\frac{\sqrt{L}}{\sqrt{\sim}}$ | 6-67 |
| BIN data increment | $\begin{array}{\|l\|} \hline \text { INC } \\ \hline \text { INCP } \\ \hline \end{array}$ | (d) | - (d) $+1 \rightarrow$ (d) |  | 6-69 |
|  | DINC <br> DINCP | (d) | -(®) +1, (®) $)+1 \rightarrow($ ( +1 , © $)$ | $\frac{\sqrt{4}}{5}$ | 6-72 |
|  | $\begin{array}{\|l\|} \hline \text { DEC } \\ \hline \text { DECP } \end{array}$ | (d) | - © $-1 \rightarrow$ (d) |  | 6-69 |
| BIN data decrement | $\begin{array}{\|l\|} \hline \text { DDEC } \\ \hline \text { DDECP } \\ \hline \end{array}$ | (a) |  |  | 6-72 |

### 2.4.3 Data conversion instructions

Table 2.12 Data conversion instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| BCD data conversion | BCD | (5), (d) |  | $\frac{\sqrt{4}}{4}$ | 6-74 |
|  | DBCD <br> DBCDP | (5), (d) | $\cdot \stackrel{(\mathrm{s}+1, \mathrm{~s})}{\mathrm{L}^{2}} \mathrm{BIN}(0 \text { to } 99999999)$ | $\frac{\sqrt{\square}}{\sqrt{5}}$ |  |
| BIN data conversion | BIN | (5), (d) |  | $\frac{\square}{\square}$ | 6-77 |
|  | DBIN <br> DBINP | (5), (d) |  | $\frac{\square}{\square}$ |  |
| BIN data to floating-point data conversion (single precision) | FLT <br> FLTP | (5), (d) | Conversion to real number |  | 6-80 |
|  | DFLT <br> DFLTP | (s), (d) | Conversion to real number |  |  |
| BIN data to floating-point data conversion (double precision) | FLTD <br> FLTDP | (5), (d) | Conversion to real number | $\frac{\square}{\square}$ | 6-83 |
|  | DFLTD <br> DFLTDP | (s), (d) | Conversion to real number |  |  |
| Floating-point data to BIN data conversion (single precision) | INT <br> INTP | (5), (d) | Conversion to BIN data <br> (s) +1, (s) $\longrightarrow$ (d) <br> 4 Real number (-32768 to 32767) | $\frac{\square}{\square}$ | 6-86 |
|  | DINT <br> DINTP | (5), (d) | Conversion to BIN data |  |  |
| Floating-point data to BIN data conversion (double precision) | INTD | (s), (d) | Conversion to BIN data | $\frac{\sqrt{\square}}{5}$ | 6-89 |
|  | DINTD <br> DINTDP | (5), (d) | Conversion to BIN data |  |  |

Table 2.12 Data conversion instructions (continued)

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 16-bit BIN data to 32-bit BIN data conversion | DBL | (5), (d) | $\cdot \stackrel{\text { s }}{\stackrel{\text { Conversion }}{\longrightarrow}}(\text { (d) }+1, \text { (d) })$ | $\sqrt{\square}$ | 6-92 |
|  | DBLP |  |  | - |  |
| 32-bit BIN data to 16-bit BIN data conversion | WORD | (s), (d) | $\cdot \frac{(\mathrm{s}+1, \mathrm{~s})}{\Delta} \xrightarrow{\text { Conversion }} \text { (d) }$ | $\sqrt{\square}$ | 6-94 |
|  | WORDP |  |  | $\uparrow$ |  |
| BIN data to Gray code data conversion | GRY | (5), (d) | Conversion to Gray code data | $\square$ | 6-96 |
|  | GRYP |  |  | ${ }^{5}$ |  |
|  | DGRY | (s), (d) | Conversion to Gray code data | $\square$ |  |
|  | DGRYP |  |  |  |  |
| Gray code data to BIN data conversion | GBIN | (5), (d) | Conversion to BIN data | $\square$ | 6-98 |
|  | GBINP |  |  | , |  |
|  | DGBIN | (5), (d) | Conversion to BIN data | $\square$ |  |
|  | DGBINP |  |  | $\uparrow$ |  |
| Sign inversion (two's complement) | NEG | (d) | $\stackrel{(\underset{\sim}{\mathrm{a}}}{\text { (d) }}$ | $\square$ | 6-100 |
|  | NEGP |  |  | $\uparrow$ |  |
|  | DNEG |  | - $\overline{(d)+1,(d)} \longrightarrow$ (d) +1 ,(d) | $\square$ |  |
|  | DNEGP | (a) | - BIN data | - |  |
| Floating-point sign inversion (single-precision) | ENEG | (d) | $\underset{(\mathrm{d})+1,(\mathrm{~d})}{\longrightarrow} \longrightarrow(\mathrm{d}+1, \mathrm{~d})$ | $\sqrt{\square}$ | 6-102 |
|  | ENEGP |  |  | $\uparrow$ |  |
| Floating-point sign inversion (double-precision) | EDNEG | (d) | $($ (d) +3 ,(d) +2 ,(d) +1 ,(d) $\longrightarrow($ (d) +3 ,(d) +2 ,(d) +1 ,(d) $)$ Real number data | $\square$ | 6-104 |
|  | EDNEGP |  |  | $\uparrow$ |  |
| Block data conversion | BKBCD | (5), n , (d) | - Batch converts BIN data n points from BCD data and stores the result to and the following devices. | $\square$ | 6-106 |
|  | BKBCDP |  |  |  |  |
|  | BKBIN | (5), n , (d) | - Batch converts BCD data $n$ points from to BIN data and stores the result to (d) and the following devices. | $\sqrt{\square}$ | 6-109 |
|  | BKBINP |  |  | $\uparrow$ |  |
| Floating-point data single precision to double precision conversion | ECON | (s), (d) | Conversion to double precision$\begin{aligned} -(\mathrm{s}+1, \mathrm{®}) \end{aligned} \xrightarrow{(\mathrm{d}+3,(\mathrm{~d}+2,(\mathrm{~d}+1, \text { (d) })}$ | $\sqrt{\square}$ | 6-112 |
|  | ECONP |  |  | - |  |
| Floating-point data double precision to single precision conversion | EDCON | (5), (d) | Conversion to single precision <br> - $\underset{64-\text { bit floating-point real number }}{(\mathrm{s}+3, \mathrm{~s}+2, \mathrm{~s}+1, \mathrm{~S})}(\mathrm{d}+1, \mathrm{~d})$ | $\sqrt{\square}$ | 6-114 |
|  | EDCONP |  |  | $\uparrow$ |  |

### 2.4.4 Data transfer instructions

Table 2.13 Data transfer instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 16-bit data transfer | MOV <br> MOVP | (5), (d) | - (s) $\longrightarrow$ (d) |  |  |
| 32-bit data transfer | DMOV | (5), (d) | - (s) $+1, \mathrm{~s}) \longrightarrow$ (d) +1 , (d) |  |  |
| Floating-point data transfer (single precision) | $\begin{aligned} & \text { EMOV } \\ & \hline \text { EMOVP } \end{aligned}$ | (5), (d) | $\text { -( } \mathrm{s}+1, \mathrm{~s}) \longrightarrow(\mathrm{s}) \longrightarrow 1, \mathrm{~d})$ |  | 6-119 |
| Floating-point data transfer (double precision) | EDMOV EDMOVP | (5), (d) | $\begin{gathered} \text { •( } \mathrm{s}+3, \mathrm{~s}+2, \mathrm{~s}+1, \mathrm{~s}) \rightarrow(\mathrm{s})+3, \text { (d) }+2, \text { (d) }+1, \text { (d) }) \\ \text { Real number data } \end{gathered}$ |  | 6-121 |
| Character string data transfer | \$MOV <br> \$MOVP | (5), (d) | - Transfers the character string data specified for (s) to (d) and the following devices. |  | 6-123 |
| 16-bit data negation transfer | CML <br> CMLP | (5), (d) | - $\overline{\text { (s) }} \longrightarrow$ (d) |  | 6-126 |
| 32-bit data negation transfer | DCML <br> DCMLP | (5), (d) | $\cdots(\mathrm{S}+1, \mathrm{~s}) \longrightarrow$ (d) +1 , (d) |  |  |
| Block data transfer | BMOV BMOVP | (s), n , (d) |  |  | 6-131 |
| Same 16-bit BIN data block transfer | $\begin{aligned} & \text { FMOV } \\ & \hline \text { FMOVP } \end{aligned}$ | (s), n , (d) |  |  | 6-135 |
| Same 32-bit BIN data block transfer | DFMOV DFMOVP | (s), n , (d) |  |  | 6-138 |
| 16-bit data exchange | XCH <br> XCHP | (d1), (12) | $\cdots$ (d1) (12) |  | 6-141 |
| 32-bit data exchange |  | (d1), (12) | - (d1) +1, (11) $) \longleftrightarrow($ (22) +1, (12) |  |  |
| Block data exchange |  | n, (11) , (12) |  |  | 6-144 |
| Upper and lower bytes exchange | SWAP <br> SWAPP | (5) | (s)8 to b 8 b 7 to b 8 bits <br> (s) 8 bits 88 b 7 to b0 8 bits |  | 6-147 |

### 2.4.5 Program branch instructions

Table 2.14 Program branch instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Jump | CJ | p | - Jumps to $p$ when the input condition is satisfied. | $\sqrt{\square}$ | 6-149 |
|  | SCJ | p | - Jumps to $p$ from the next scan after the input condition is satisfied. | $\sqrt{\square}$ |  |
|  | JMP | p | - Unconditionally jumps to p. |  |  |
|  | GOEND | - | - Jumps to the FEND instruction when the input condition is satisfied. |  | 6-154 |

### 2.4.6 Program execution control instructions

Table 2.15 Program execution control instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Disable interrupt | DI | - | - Disables the execution of interrupt programs. |  | 6-156 |
| Enable interrupt | El | - | - Cancels the execution disabled status of the interrupt program. |  |  |
| Disable/enable interrupt setting | IMASK | (5) | - Disables/enables interrupts for each interrupt program. |  |  |
| Recovery | IRET | - | - Returns to the sequence program from the interrupt program. |  | 6-165 |

### 2.4.7 I/O refresh instructions

Table 2.16 I/O refresh instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| I/O refresh | RFS | (s), n | - Performs partial refresh of the corresponding inputs/outputs during one scan. | $\sqrt{\square}$ | 6-168 |
|  | RFSP |  |  | $\stackrel{\square}{4}$ |  |

### 2.4.8 Other convenient instructions

Table 2.17 Other convenient instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Up/down counter | UDCNT1 | (s), n , (d) |  |  | 6-170 |
|  | UDCNT2 | (5), n , (d) | (5) +0 आทに! <br> (s) $+1 \rightarrow \square \longrightarrow$ ? leviver <br> Current Cn value $\begin{array}{llllllllllllll}13 & 1 & 2 & 13 & 4 & 5 & 4 & 3 & \mid 2 & 1 & 0 & -1\end{array}$ Cn contact point $\qquad$ $\qquad$ $\qquad$ |  | 6-173 |
| Teaching timer | TTMR | n, (d) | - (Time that TTMR is | $\sqrt{\square}$ | 6-176 |
| Special function timer | STMR | (5), n , (d) | - The 4 points from the bit device specified for (d) perform the following operations depending on the ON/OFF status of input condition of the STMR instruction: <br> (d) +0 : Off-delay timer output <br> (d) +1 : One-shot timer output after Off <br> (d) +2 : One-shot timer output after On <br> (d) +3 : On-delay + Off-delay timer output |  | 6-178 |
| Shortest direction control | ROTC | (s), $\mathrm{n} 1, \mathrm{n} 2$, (d) | - Rotates the rotary table divided into n1, from the stop position to the position specified for $($ s +1$)$ in the shortest path. | $\sqrt{\square}$ | 6-181 |
| Ramp signal | RAMP | n1, n2, n3, @11, <br> (d2) | - Changes the data of the device specified for (d1) from n 1 to n 2 in n 3 scans. | $\sqrt{\square}$ | 6-184 |
| Pulse density | SPD | (s), n , (d) | - Counts the pulse inputs of the device specified for (s) for the time specified by $n$ and stores the result to the device specified for (d). | $\sqrt{\square}$ | 6-187 |
| Fixed cycle pulse output | PLSY | n1, n2, (d) | -(n1) Hz (d) <br> Output n2 times | $\sqrt{\square}$ | 6-189 |
| Pulse width modulation | PWM | n1, n2, (d) |  | $\boxed{\square}$ | 6-191 |
| Matrix input | MTR | (5), n, (d1), (d2) | - Sequentially captures the data of 16 points multiplied by n columns from the device specified for (s), and stores the captured data to (ब12) and the following devices. | $\sqrt{\square}$ | 6-193 |

### 2.5 Application Instructions

### 2.5.1 Logical operation instructions

Table 2.18 Logical operation instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Logical AND | WAND | (11), (2), (1) | - (51) $\wedge$ (2) $\rightarrow$ (c) | $\sqrt{\square}$ | 7-3 |
|  | WANDP |  |  | $\uparrow$ |  |
|  | DAND | (13), (2), (d) | $\cdot($ (1) +1, (1) $) \wedge($ (2) +1, (®2) $\rightarrow$ (®) +1, (d) | $\sqrt{\square}$ |  |
|  | DANDP |  |  | $\uparrow$ |  |
|  | BKAND | (17), (2), n, © ${ }^{\text {d }}$ |  | $\sqrt{\square}$ | 7-6 |
|  | BKANDP |  |  | ヶ |  |
| Logical OR | WOR | (13), (2), (d) | - (51) $V$ (2) $\rightarrow$ (d) | $\sqrt{\square}$ | 7-9 |
|  | WORP |  |  | $\uparrow$ |  |
|  | DOR | (1), (2), (d) | $\cdot($ (1) +1, (13) $) \vee($ (2) +1, (2) $) \rightarrow($ (d) +1, (1) | $\sqrt{\square}$ |  |
|  | DORP |  |  | $\stackrel{\square}{ }$ |  |
|  | BKOR | (®1), (2), n, © |  | $\sqrt{\square}$ | 7-12 |
|  | BKORP |  |  | F |  |
| Exclusive OR | WXOR | (17), (2), (1) | - (31) $\forall$ (2) $\rightarrow$ (c) | $\sqrt{\square}$ | 7-15 |
|  | WXORP |  |  | $\uparrow$ |  |
|  | DXOR | (11), (2), (d) | $\cdot($ (31) +1, (1) $) \forall($ (2) +1, (2) $) \rightarrow($ (d) +1, (1) | $\sqrt{\square}$ |  |
|  | DXORP |  |  | $\checkmark$ |  |
|  | BKXOR | (®1), © , n, © (1) |  | $\sqrt{\square}$ | 7-18 |
|  | BKXORP |  |  |  |  |
| Exclusive NOR | WXNR | (13), (2), (d) | - $\overline{\text { (1) }} \forall^{(2)} \rightarrow$ (d) | $\sqrt{\square}$ | 7-21 |
|  | WXNRP |  |  | $\uparrow$ |  |
|  | DXNR | (13), (2), (d) |  | $\sqrt{\square}$ |  |
|  | DXNRP |  |  | $\uparrow$ |  |
|  | BKXNR | (®1), (2), n, © |  | $\sqrt{\square}$ | 7-24 |
|  | BKXNRP |  |  | $5$ |  |

### 2.5.2 Rotation instructions

Table 2.19 Rotation instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Right rotation | ROR <br> RORP | n, (d) | Rotates n bits to the right. Carry flag |  | 7-27 |
|  | $R C R$ <br> RCRP | n , (d) | Rotates $n$ bits to the right. Carry flag |  |  |
| Left rotation | ROL <br> ROLP | n , (d) | Carry flag Rotates n bits to the left. |  | 7-31 |
|  | $\begin{array}{\|l\|} \hline \text { RCL } \\ \hline \text { RCLP } \end{array}$ | n, (d) |  |  |  |
| Right rotation | DRORP | n, (d) | Rotates n bits to the right. Carry flag |  | 7-35 |
|  | DRCR <br> DRCRP | n , (d) | Rotates n bits to the right. Carry flag |  |  |
| Left rotation | DROLP | n , (d) | Carry flag Rotates n bits to the left. | $\frac{\sqrt{2}}{\sqrt{5}}$ | 7-38 |
|  | DRCL <br> DRCLP | n, (d) |  |  |  |

### 2.5.3 Shift instructions

Table 2.20 Shift instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| n -bit shift | SFR SFRP | n , (d) |  |  | 7-41 |
|  | SFL <br> SFLP | n , (d) |  |  |  |
| 1-bit shift of n-bit data | BSFR <br> BSFRP | n , (d) |  |  | 7-44 |
|  | BSFL BSFLP | n , (d) |  |  |  |
| $n$-bit shift of n -bit data | SFTBR SFTBRP | n1, n2, (d) |  |  | 7-47 |
|  | SFTBL SFTBLP | n1, n2, (d) |  |  |  |
| 1-word shift of n-word data | DSFR <br> DSFRP | n , (d) |  |  | 7-50 |
|  | DSFL <br> DSFLP | n , (d) |  |  |  |
| n-word shift of n-word data | SFTWR SFTWRP | n1, n2, © |  |  | 7-53 |
|  | SFTWL SFTWLP | n1, n2, © |  |  |  |

### 2.5.4 Bit processing instructions

Table 2.21 Bit processing instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Bit set/reset | BSET ${ }^{\text {BSETP }}$ | n, (d) |  |  | 7-56 |
|  | BRST <br> BRSTP | n , (d) |  | $\frac{\sqrt{\square}}{\sqrt{4}}$ |  |
| Bit test | TEST <br> TESTP | (13), (2), (d) |  |  | 7-59 |
|  | DTEST <br> DTESTP | (31), (2), (d) |  |  |  |
| Bit device batch reset | BKRST <br> BKRSTP | (5) n |  |  | 7-62 |

### 2.5.5 Data processing instructions

Table 2.22 Data processing instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Data search | SER <br> SERP | (51), (32), n, (d) | $\square$ |  | 7-64 |
|  | DSER <br> DSERP | (s1), (32), n, (d) |  |  |  |
| Bit check | SUM | (5), (d) | b15 <br> (s) b0 | $\square$ | 7-68 |
|  | SUMP |  | (d): Number of 1 s |  |  |
|  | DSUM | (s), (d) | (s)+1) (s) | $\square$ |  |
|  | DSUMP |  | $\xrightarrow{\text { (d) }}$ Number of 1s |  |  |
| Decode | DECO | (5), n, (d) | Decode from 8 to 256 | $\sqrt{\square}$ | 7-71 |
|  | DECOP |  |  |  |  |
| Encode | ENCO | (s), n , (d) | Decode from 256 to 8 | $\sqrt{\square}$ | 7-75 |
|  | ENCOP |  |  | $\uparrow$ |  |
| 7-segment decode | SEG | (5), (d) | b3 to b0 | $\square$ | 7-78 |
|  | SEGP |  |  | $\uparrow$ |  |
| Separation and connection | DIS | (5), n, (d) | - Separates 16 -bit data specified for in units of 4 bits and stores the result to lower 4 bits $n$ points from <br> (a) $(n \leqq 4)$. | $\sqrt{\square}$ | 7-81 |
|  | DISP |  |  |  |  |
|  | UNI | (5), n , (d) | - Connects lower 4-bit data $n$ points from the device specified for (s) and stores the result to the device specified for <br> (d) $(n \leqq 4)$. | $\square$ | 7-83 |
|  | UNIP |  |  | $\uparrow$ |  |
|  | NDIS | (s1), (22), (d) | - Separates the data in (s1) and the following devices per bits specified by the devices from (®2), and stores the result to (a) and the following devices. | $\square$ | 7-85 |
|  | NDISP |  |  |  |  |
|  | NUNI | (31), (32), (d) | - Connects the data in (s1) and the following devices per bits specified by the devices from (®2), and stores the result to (d) and the following devices. |  |  |
|  | NUNIP |  |  | 个 |  |
|  | WTOB | (5), n, (d) | - Separates 16-bit data specified for units of 8 bits for $n$ points of devices and stores the result in order from the device specified for (d). | $\square$ | 7-90 |
|  | WTOBP |  |  | $\uparrow$ |  |
|  | BTOW | (s), n , (d) | - Connects lower 8 bits of 16-bit data for n points from the device specified for (s) to 16 bits and stores the result sequentially to the devices starting from the one specified for (d). | $\sqrt{\square}$ |  |
|  | BTOWP |  |  | $\Psi$ |  |

Table 2.22 Data processing instructions (continued)


### 2.5.6 Structured instructions

Table 2.23 Structured instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Repeat | FOR | n | - Executes the operation n times within the$\square$ FOR to $\square$ NEXT instruction loop. |  | 7-112 |
|  | NEXT | - |  |  |  |
|  | BREAK | p, (d) | - Forcibly terminates the execution of the FOR to$\square$$\square$ NEXT instruction loop and jumps to the pointer $p$. |  | 7-115 |
|  | BREAKP |  |  |  |  |
| Subroutine program call | CALL | $p$ | - Executes the subroutine program $p$ when the input condition is satisfied. | $\square$ | 7-118 |
|  | CALLP |  |  | $\uparrow$ |  |
| Return from subroutine program | RET | - | - Returns from the subroutine program. | , | 7-120 |
| Selection of refresh instructions | COM | - | - Executes the auto refresh of intelligent function modules, and the auto refresh of general data and the CPU shared memory. | $\square$ | 7-122 |
|  | CCOM | - | - Executes the auto refresh of intelligent function modules, the auto refresh of the CPU shared memory, and the communication processes. |  | 7-130 |
|  | CCOMP |  |  | $\uparrow$ |  |

### 2.5.7 Data table operation instructions

Table 2.24 Data table operation instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Data table process | FIFW <br> FIFWP | (5), (d) |  |  | 7-132 |
|  | FIFR <br> FIFRP | (5), (d) | (s) |  | 7-135 |
|  | FPOP <br> FPOPP | (5), (d) | (5)Pointer Pointer - 1  <br>    <br>  (d)  <br>    |  | 7-138 |
|  | FDEL <br> FDELP | (5), n, (d) | (s) |  | 7-141 |
|  | FINS <br> FINSP | (5), n , (d) |  |  |  |

### 2.5.8 Buffer memory access instructions

Table 2.25 Buffer memory access instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Data read | FROM | n1, n2, n3, © | - Reads data in units of 16 bits from the intelligent function module. | $\checkmark$ | 7-144 |
|  | FROMP |  |  | - |  |
|  | DFRO | n1, n2, n3, (d) | - Reads data in units of 32 bits from the intelligent function module. | $\square$ |  |
|  | DFROP |  |  | $\uparrow$ |  |
| Data write | TO | (5), n1, n2, n3 | - Writes data in units of 16 bits to the intelligent function module. | $\square$ | 7-148 |
|  | TOP |  |  | $\uparrow$ |  |
|  | DTO | (5, n1, n2, n3 | - Writes data in units of 32 bits to the intelligent function module. | $\square$ |  |
|  | DTOP |  |  | $\uparrow$ |  |

### 2.5.9 Display instructions

Table 2.26 Display instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| ASCII print | PR | (5), (d) | - When SM701 is ON, outputs 8 points of ASCII code (16 characters) from the device specified for (s) to the output module. |  | 7-152 |
|  | PR | (5), (d) | - When SM701 is OFF, outputs ASCII code data from the device specified for (s) up to 00 H to the output module. |  |  |
|  | PRC | (5), (d) | - Converts comments in the device specified for (s) to ASCII code data and outputs the result to the output module. |  | 7-156 |
| Reset | LEDR | - | - Resets the error display or the annunciator. | - | 7-160 |

### 2.5.10 Debug/error diagnostics instructions

Table 2.27 Debug/error diagnostics instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Check | CHKST | - | - Executes the CHK instruction when the CHKST instruction is executed. <br> - Jumps to the step following the CHK instruction when the CHKST instruction is not executed. |  | 7-163 |
|  | CHK | - | - Normal operation $\rightarrow$ SM80: OFF, SD80: 0 <br> - Abnormal operation $\rightarrow$ SM80: ON, SD80: Error number |  |  |
|  | CHKCIR | - | - Start of change in a ladder pattern checked by the CHK instruction |  | 7-167 |
|  | CHKEND | - | - End of change in a ladder pattern checked by the CHK instruction |  |  |

### 2.5.11 String processing instructions

Table 2.28 String processing instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| BIN data to decimal ASCII data conversion | BINDA | (s), (d) | - Converts 1 word of BIN data specified for (s) to 5 digits of decimal ASCII data and stores the result to the word device specified for (d). |  | 7-171 |
|  | DBINDA <br> DBINDAP | (5), (d) | - Converts 2 words of BIN data specified for (s) to 10 digits of decimal ASCII data and stores the result to ©d and the following word devices. |  |  |
| BIN data to hexadecimal ASCII data conversion | BINHA | (5), (d) | - Converts 1 word of BIN data specified for (s) to 4 digits of hexadecimal ASCII data and stores the result to (d) and the following word devices. |  | 7-176 |
|  | DBINHA <br> DBINHAP | (5), (d) | - Converts 2 words of BIN data specified for (s) to 8 digits of hexadecimal ASCII data and stores the result to (d) and the following word devices. |  |  |
| BCD data to decimal ASCII data conversion | BCDDA | (5), (d) | - Converts 1 word of BCD data specified for (s) to 4 digits of decimal ASCII data and stores the result to (d) and the following word devices. |  | 7-181 |
|  | DBCDDA | (s), (d) | - Converts 2 words of BCD data specified for (s) to 8 digits of decimal ASCII data and stores the result to (d) and the following word devices. |  |  |
| Decimal ASCII data to BIN data conversion | DABIN | (5), (d) | - Converts 5 digits of decimal ASCII data specified for (s) to 1 word of BIN data and stores the result to the word device specified for (d). |  | 7-186 |
|  | DDABIN <br> DDABINP | (5), (d) | - Converts 10 digits of decimal ASCII data specified for (s) to 2 words of BIN data and stores the result to the word device specified for (d). |  |  |
| Hexadecimal ASCII data to BIN data conversion | HABIN <br> HABINP | (5), (d) | - Converts 4 digits of hexadecimal ASCII data specified for (s) to 1 word of (16-bit) BIN data and stores the result to the word device specified for (d). |  | 7-190 |
|  | DHABIN | (s), (d) | - Converts 8 digits of hexadecimal ASCII data specified for (s) to 2 words of BIN data and stores the result to the word device specified for (d). |  |  |
| Decimal ASCII data to BCD data conversion | DABCD | (s), (d) | - Converts 4 digits of decimal ASCII data specified for (s) to 1 word of BCD data and stores the result to the word device specified for (d). |  | 7-193 |
|  | DDABCD <br> DDABCDP | (5), (d) | - Converts 8 digits of decimal ASCII data specified for (s) to 2 words of BCD data and stores the result to the word device specified for (d). |  |  |
| Device comment data read | COMRD <br> COMRDP | (5), (d) | - Stores the comment data from the device specified for © ${ }^{\text {s }}$ and stores it to the device specified for ${ }^{(6)}$. |  | 7-196 |

Table 2.28 String processing instructions (continued)

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Character string length detection | LEN | (5), (d) | - Stores the length (number of characters) of character string data stored in the device specified for (s) to the device specified for (d). | $\square$ | 7-200 |
|  | LENP |  |  | $\uparrow$ |  |
| BIN data to decimal character string data conversion | STR | (51), (22), (d) | - Converts 1 word of BIN value specified for (s2) to character string data according to the total number of digits and number of digits in the fractional part specified for (s1), and stores the result to the device specified for (d). | $\square$ | 7-202 |
|  | STRP |  |  | $\uparrow$ |  |
|  | DSTR | (s1), (22), (d) | - Converts 2 words of BIN value specified for (32) to character string data according to the total number of digits and number of digits in the fractional part specified for (s1), and stores the result to the device specified for (d). | $\square$ |  |
|  | DSTRP |  |  | $\uparrow$ |  |
| Decimal character string data to BIN data conversion | VAL | ( 5 , (d1), (12) | - Converts character string data contain a decimal point specified for (s) to 1 word of BIN value and number of digits in the fractional part, and stores the result to the devices specified for (d1) and (12). | $\square$ | 7-209 |
|  | VALP |  |  | $\uparrow$ |  |
|  | DVAL | (5), (d1), (12) | - Converts character string data contain a decimal point specified for (s) to 2 words of BIN data and number of digits in the fractional part, and stores the result to the devices specified for (d1) and (d2). | $\square$ |  |
|  | DVALP |  |  | $\uparrow$ |  |
| Floating-point data to character string data conversion | ESTR | (51), (22), (d) | - Converts floating-point data specified for (s) to character string data according to the display specification specified for $\qquad$ and stores the result to the device specified for (d). | $\square$ | 7-214 |
|  | ESTRP |  |  | $\uparrow$ |  |
| Character string data to floatingpoint data conversion | EVAL | (5), (d) | - Converts character string data specified for (s) to floating-point data and stores the result to the device specified for | $\square$ | 7-221 |
|  | EVALP |  |  | $\uparrow$ |  |
| Hexadecimal BIN data to ASCII data conversion | ASC | (5), n , (d) | - Converts 1 word of BIN value in (s) and the following word devices to hexadecimal ASCII data and stores the number of characters specified for $n$ to (d) and the following word devices. | $\square$ | 7-226 |
|  | ASCP |  |  | ${ }^{+}$ |  |
| ASCII data to hexadecimal BIN data conversion | HEX | (5), n , (d) | - Converts n numbers of hexadecimal ASCII data in (s) and the following word devices to BIN data and stores the result to (d) and the following word devices. | $\square$ | 7-229 |
|  | HEXP |  |  | $\uparrow$ |  |

Table 2.28 String processing instructions (continued)

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| String processing | RIGHT <br> RIGHTP | (5), n , (d) | - Stores n characters from the end of the character string data specified for (s) to the device specified for (a). |  | 7-232 |
|  | LEFT | (5), n, (d) | - Stores n characters from the start of the character string data specified for (s) to the device specified for (®). |  |  |
|  | MIDR | (13), (2), (d) | - Stores the specified number of characters from the position specified for (2) of the character string data specified for (31) to the device specified for ( ${ }^{\text {( }}$. |  | 7-235 |
|  | MIDW | (51), (2), (d) | - Stores the specified number of characters from the character string data (s1) to the position specified for (2) of the character string data © . | $\boxed{\square}$ |  |
|  | INSTR <br> INSTRP | (31), (32), n, (c) | - Searches for the character string data (51) from the nth character of the character string data $(2)$ and stores matched positions to (d). |  | 7-240 |
|  | STRINS <br> STRINSP | (5), n, (d) | - Inserts the character string data specified for (s) to the n-th character (insert position) from the top of the character string data specified for © ${ }^{(d)}$ |  | 7-243 |
|  | STRDEL <br> STRDELP | n1, n2, (c) | - Deletes n2 characters from n1-th character (deletion start position) from the top of the character string data specified for (d). | $\sqrt{\square}$ | 7-246 |
| Floating-point data to BCD format data conversion | EMOD | (31), (2), (d) | - Converts floating-point data (s1) to BCD data of the fractional part digits specified for $\circledR_{2} 2$ and stores the result to the device specified for (d). |  | 7-248 |
| BCD format data to floating-point data conversion | EREXP | (1), (2), (d) | - Converts BCD data (st) to floating-point data with the fractional part digits specified for (®2) and stores the result to the device specified for © ${ }^{(d)}$. |  | 7-251 |

### 2.5.12 Special function instructions

Table 2.29 Special function instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Trigonometric functions (singleprecision floating-point data) | SIN | (5), (d) | - $\operatorname{Sin}(\mathrm{s}+1, \mathrm{~s}) \longrightarrow$ (d) $\longrightarrow$, (d) |  | 7-253 |
|  | COS | (5), (d) | $\cdot \operatorname{Cos}(\mathrm{s}+1, \mathrm{~s}) \longrightarrow(\mathrm{d})+1, \mathrm{~d})$ | $\frac{\sqrt{4}}{5}$ | 7-257 |
|  | TAN <br> TANP | (5), (d) | $\text { - } \operatorname{Tan}(\mathrm{s}+1, \text { (s) } \longrightarrow(\mathrm{d})+1,(\mathrm{~d})$ |  | 7-261 |
|  | ASIN | (5), (d) | $\left.\cdot \operatorname{Sin}^{-1}(\mathrm{~s})+1, \mathrm{~s}\right) \longrightarrow$ (d) +1 , (d) |  | 7-265 |
|  | ACOS | (5), (d) | $\cdot \operatorname{Cos}^{-1}(\mathrm{~s}+1, \mathrm{~s}) \longrightarrow(\mathrm{C})+1$, (d) | $\frac{\sqrt{\square}}{5}$ | 7-269 |
|  | ATAN <br> ATANP | (5), (d) | $\cdot \operatorname{Tan}^{-1}(\mathrm{~s}+1, \mathrm{~s}) \longrightarrow$ (d)+1,(d) | $\frac{\square}{5}$ | 7-273 |
| Trigonometric functions (doubleprecision floating-point data) | SIND <br> SINDP | (5), (d) | $\text { - } \sin (\mathrm{s}+3, \mathrm{~s})+2, \mathrm{~s}+1, \mathrm{~s}) \text { ) }$ |  | 7-255 |
|  |  | (5), (d) | $\text { - } \operatorname{Cos}(\mathrm{s}+3, \mathrm{~s}+2, \mathrm{~s})+1, \mathrm{~s})$ |  | 7-259 |
|  | TAND <br> TANDP | (5), (d) | $\text { - } \operatorname{Tan}(\mathrm{s}+3, \mathrm{~s}+2, \mathrm{~s}+1, \mathrm{~s}) \text { ) }$ |  | 7-263 |
|  | ASIND | (5), (d) | $\text { - } \sin ^{-1}(\mathrm{~s}+3, \mathrm{~s}+2, \text { (s) }+1, \text { (s) })$ |  | 7-267 |
|  | ACOSD | (5), (d) | $\text { - } \operatorname{Cos}^{-1}(\mathrm{~s}+3, \mathrm{~s}+2, \mathrm{~s}+1, \mathrm{~s})$ |  | 7-271 |
|  | ATAND <br> ATANDP | (5), (d) | $\text { - } \operatorname{Tan}^{-1}(\mathrm{~s}+3,(\mathrm{~s}+2,(\mathrm{~s}+1, \mathrm{~s})$ |  | 7-275 |

Table 2.29 Special function instructions (continued)


Table 2.29 Special function instructions (continued)

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Square root | BSQR BSQRP | (5), (d) | $\begin{array}{rl\|l} \bullet \sqrt{(s)} \longrightarrow r & \text { (d)+0 } & \text { Integer part } \\ & +1 & \text { Fractional part } \end{array}$ |  | 7-311 |
|  | BDSQR <br> BDSQRP | (5), (d) |  |  |  |
| Trigonometric functions (BCD format) | BSIN | (5), (d) | $\bullet$ Sin $(s) \longrightarrow \quad(d)+0$ Sign  <br>  +1 Integer part <br>  +2 Fractional part <br>    |  | 7-315 |
|  |  | (5), (d) |  |  |  |
|  | BTAN <br> BTANP | (5), (d) |  |  | 7-321 |
|  | $\qquad$ <br> BASINP | (5), (d) |  |  | 7-324 |
|  | BACOS | (5), (d) |  | $\frac{\square}{5}$ | 7-327 |
|  | BATAN | (5), (d) |  |  | 7-330 |

### 2.5.13 Data control instructions

Table 2.30 Data control instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Upper/lower limit control | LIMIT <br> LIMITP | (51), (32), (3), (d) | -When (33) < (1), stores the value of (s1) to © (d) <br> - When $(11) \leqq$ (33) $\leqq$ ( 22 , <br> stores the value of (3) to © . <br> -When (22) < (33), <br> stores the value of $(22)$ to © . |  |  |
|  | DLIMIT <br>  <br> DLIMITP | (31), (32), (3), (d) | - When $($ (3) +1 , (33) $)<($ (51) +1 , (51) $)$, <br> stores the values of $($ (S1) +1 , (ㄷ1) $)$ to (d) +1 , (d) . <br>  stores the values of $($ (3) +1 , (3) $)$ to (d) +1 , (d) . <br> - When $\left({ }^{(32}\right)$, (32) +1$)<($ (33), (33) +1$)$, stores the values of $\left(\mathrm{S}_{2}\right)+1$, ( 2 ) to (d +1 , (d). |   | 7-332 |
| Dead band control | BAND <br> BANDP | (31), (32), (3), (d) | - When (51) $\leqq$ (33) $\leqq$ (2), $0 \rightarrow$ (d) <br> -When (33) < (51), (33)-(1) $\rightarrow$ (d) <br> - When (52) < (33), (33)-(®2) $\rightarrow$ (d) |  | 7-336 |
|  | DBAND <br> DBANDP | (31), (32), (3), (d) |  | $\frac{\sqrt{4}}{5}$ |  |
| Zone control |  | (51), (32), (3), (d) | - When (3) $=0,0 \rightarrow$ (d) <br> - When (3) $>0$, (3) + (2) $\rightarrow$ (d) <br> - When (33) $<0$, (33) + (51) $\rightarrow$ (d) |  | 7-340 |
|  | DZONE <br> DZONEP | (31), (32), (3), (d) | - When $\begin{aligned} & \hline \text { (③)+1, (3) })=0, \\ & 0 \rightarrow(\text { (d) }+1, \text { © }) \end{aligned}$ <br> - When $($ (3) +1, (33) $)>0$, $(\text { (33) }+1, \text {, (33) })+(\text { (®2 })+1, \text { (32 }) \rightarrow(\text { (d) }+1, \text { (d) })$ <br> - When $($ (3) +1 , (33) $)<0$, $(\text { (3) }+1 \text {, (33) })+(\text { (① }+1 \text {, (51) }) \rightarrow(\text { (d) }+1 \text {, (d) })$ |  |  |

Table 2.30 Data control instructions (continued)

\begin{tabular}{|c|c|c|c|c|c|}
\hline Category \& Instruction name \& Argument \& Processing details \& Executing condition \& Page \\
\hline \multirow{2}{*}{Coordinate by point data} \& \begin{tabular}{l} 
SCL \\
\\
\hline SCLP
\end{tabular} \& (51), (22), (d) \& - Process the scaling on the conversion data (unit of 16-bit data) specified for (22) with the input value specified for © \({ }^{(1)}\), and stores the result to the device specified for © \({ }^{(d)}\). The scaling conversion is processed in accordance with the conversion data for scaling stored in ©2 and the following devices. \&  \& \multirow{2}{*}{7-344} \\
\hline \& \begin{tabular}{l} 
DSCL \\
\\
\hline DSCLP
\end{tabular} \& (31), (2), (d) \& - Process the scaling on the conversion data (unit of 32-bit data) specified for \({ }^{(22}\) ) with the input value specified for © \((11\), and stores the result to the device specified for (d). The scaling conversion is processed in accordance with the conversion data for scaling stored in ©2 and the following devices. \&  \& \\
\hline \multirow{2}{*}{Coordinate by \(\mathrm{X} / \mathrm{Y}\) data} \& SCL2

SCL2P \& ( (1), (32), (d) \& - Process the scaling on the conversion data (unit of 16-bit data) specified for ${ }^{(2)}$ with the input value specified for ${ }_{\text {®11 }}$, and stores the result to the device specified for (d). The scaling conversion is processed in accordance with the conversion data for scaling stored in ©2 and the following devices. \& 
 \& \multirow{2}{*}{7-348} <br>

\hline \& | DSCL2 |
| :--- |
|  |
| DSCL2P | \& (31), (32), (d) \& - Process the scaling on the conversion data (unit of 32-bit data) specified for $\S_{2}$ ) with the input value specified for (s1), and stores the result to the device specified for (d). The scaling conversion is processed in accordance with the conversion data for scaling stored in (22) and the following devices. \&  \& <br>

\hline
\end{tabular}

### 2.5.14 File register switching instructions

Table 2.31 File register switching instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Block number switch | RSET | (s) | - Changes the block number of the extended file register to the number specified for (s). | $\sqrt{\square}$ | 7-351 |
|  | RSETP |  |  | $\ldots$ |  |
| File set | QDRSET | (5) | - Sets the file name used as a file register. | $\square$ | 7-354 |
|  | QDRSETP |  |  | $\uparrow$ |  |
|  | QCDSET | (5) | - Sets the file name used as a comment file. | $\square$ | 7-357 |
|  | QCDSETP |  |  |  |  |

### 2.5.15 Clock instructions

Table 2.32 Clock instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Clock data read/write | DATERD <br> DATERDP | (d) |  | $\frac{\square}{\square}$ | 7-360 |
|  | DATEWR DATEWRP | (5) | (d) +0 Year <br> +1 Month <br> +2 (Clock elements) <br> +2 Day <br> +3 Hour <br> +4 Minute <br> +5 Sec. <br> +6 Day of week <br>   | $\frac{\square}{\square}$ | 7-363 |
| Clock data addition/subtraction | DATE+ + DATE+P | (s1), (22), (d) |  |  | 7-366 |
|  | DATE- DATE-P | (51), (82), (d) |  |  | 7-369 |
| Clock data conversion | SECOND SECONDP | (s), (d) | (s) (d) <br> Hour  <br> Minute  <br> Sec. $\rightarrow$Sec. (Lower level) <br> Sec. (Upper level) |  | 7-372 |
|  | HOUR HOURP | (5), (d) |  |  | 7-374 |

Table 2.32 Clock instructions (continued)


Table 2.32 Clock instructions (continued)


### 2.5.16 Extended clock instructions

Table 2.33 Extended clock instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Extended clock data read | S_DATERD SP_DATERD | (d) |  | $\frac{\square}{\square}$ | 7-386 |
| Extended clock data addition/ subtraction | S_DATE+ SP_DATE+ | (31), (32), (d) |  |  | 7-389 |
|  | S_DATE- SP_DATE- | (31), (22), (d) | (51) <br> (2) <br> (d) |  | 7-392 |

### 2.5.17 Program control instructions

Table 2.34 Program control instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Program control instructions | PSTOP | (s) | - Sets the specified program in the standby status. | $\square$ | 7-395 |
|  | PSTOPP |  |  | 5 |  |
|  | POFF | (5) | - Turns OFF the coil of the OUT instruction of the specified program and sets the program in the standby status. | $\square$ | 7-397 |
|  | POFFP |  |  | $\uparrow$ |  |
|  | PSCAN | (5) | - Registers the specified program as a scan execution type program. | $\square$ | 7-399 |
|  | PSCANP |  |  | $\uparrow$ |  |
|  | PLOW | (5) | - Registers the specified program as a lowspeed execution type program. | $\sqrt{\square}$ | 7-401 |
|  | PLOWP |  |  | $\uparrow$ |  |
|  | LDPCHK | (5) | - Conduction state when the program of the specified file name is in execution. <br> - Non-conduction state when the program of the specified file name is not in execution. | $\square$ | 7-403 |
|  | ANDPCHK | (5) |  | $\sqrt{\square}$ |  |
|  | ORPCHK | (5) |  | $\sqrt{\square}$ |  |

### 2.5.18 Other instructions

Table 2.35 Other instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| WDT reset | WDT | - | - Resets WDT in the sequence program. | $\sqrt{\square}$ | 7-406 |
|  | WDTP |  |  | $\uparrow$ |  |
| Timing clock | DUTY | n1, n2, (d) |  | $千$ | 7-408 |
| Time check | TIMCHK | (31), (32), (d) | - Measures the ON time of the input condition, and when the input condition remains ON over the preset time, turns ON the device specified for (d). | $\sqrt{\square}$ | 7-410 |
| Direct 1-byte read/write | ZRRDB <br> ZRRDBP | n , (d) | 0 Lower 8 bits  <br> 1 Upper 8 bits  <br> 2 Lower 8 bits  <br> 3 Upper 8 bits  <br> $n$ 8 ZR1  <br> $n$ 8 bits (d) |  | 7-412 |
|  | ZRWRB | n, (d) | (s) $\square$ <br>  |  | 7-415 |
|  | ADRSET <br> ADRSETP | (5), (d) | (5) (d) <br> Indirect address of specified device <br> Device name |  | 7-418 |
| Numeric input from keyboard | KEY | (5), n, (d1), (d2) | - Imports ASCII data into 8 points of the input module specified for (s), converts the data to hexadecimal data and stores the result to (d1) and the following devices. |  | 7-419 |
| Batch save of index register | ZPUSH | (d) | - Saves the contents of index registers to (d) and the following devices. | $\boxed{\square}$ | 7-423 |
| Batch recovery of index register | ZPOP | (d) | - Reads data saved in (d) and the following devices into the index registers. | $\square$ |  |
| Module information read | UNIRD | n1, n2, (d) | - Reads the module information whose amount is specified for n 2 from the start I/O number specified for n 1 , and stores the information to © and the following devices. |  | 7-426 |
| Module type read | TYPERD | n, (d) | - Reads the module type of the start I/O number specified for n , and stores the information to (d) and the following devices. | $千$ | 7-432 |

Table 2.35 Other instructions (continued)

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Trace set | TRACE | - | - Stores trace data set in the peripheral in the number of set times when SM800, SM801, or SM802 is turned ON to the sampling trace file. | $\uparrow$ | 7-437 |
| Trace reset | TRACER | - | - Resets data set by the TRACE instruction. | f |  |
| Data write to specified file | SP_FWRITE | ((3), (ㄷ1), (2), (©0), <br> (d1) | -Writes data to the specified file. | $\hat{f}$ | 7-440 |
| Data read from specified file | SP_FREAD | $\begin{aligned} & \text { (50), (1), (d0), (d1), } \\ & \text { (d2) } \end{aligned}$ | - Reads data from the specified file. | $f$ | 7-452 |
| Data write to standard ROM | SP_DEVST | (31), ©2), n, © | - Writes data to the device data storage file in the standard ROM. | $\uparrow$ | 7-466 |
| Data read from standard ROM | S_DEVLD | (s), n, (d) | - Reads data from the device data storage file in the standard ROM. |  | 7-469 |
|  | SP_DEVLD |  |  |  |  |
| Program load from memory | PLOADP | (5), (d) | - Transfers a program stored in the memory card or standard ROM (other than drive 0) to drive 0 and sets the program in the standby type. | $\uparrow$ | 7-471 |
| Program unload from program memory | PUNLOADP | (5), (d) | - Deletes a standby type program stored in the program memory (drive 0) from the memory. | $\uparrow$ | 7-475 |
| Load + unload | PSWAPP | (51), (22), (d) | - Deletes a standby type program stored in the program memory (drive 0 ) specified for (s1) from the memory, and transfers a program stored in the memory card or standard ROM (other than drive 0 ) specified for $(22)$ to drive 0 and places the program in the standby type. | $\uparrow$ | 7-478 |
| File register high-speed block transfer | RBMOV | (5), n, (d) | - Batch transfers n points of 16-bit data from the device specified for (s) to n points of devices starting from the one specified for (d). | $\square$ | 7-481 |
|  | RBMOVP |  |  | 千 |  |
| User message | UMSG | (5) | - Displays specified character strings on the display module as a user message. | $\checkmark$ | 7-486 |

### 2.6 Data Link Instructions

### 2.6.1 Network refresh instructions

Table 2.36 Network refresh instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Network refresh | S_ZCOM_J | $\mathrm{Jn}{ }^{\text {* }}$ | - Refreshes specified network. | $\square$ | 8-2 |
|  | SP_ZCOM_J |  |  | $\uparrow$ |  |
|  | S_ZCOM_U | Un* |  | $\sqrt{\square}$ |  |
|  | SP_ZCOM_U |  |  | $\uparrow$ |  |

### 2.6.2 Reading/Registering routing information

Table 2.37 Reading/Registering routing information

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Routing information read | S_RTREAD | n , (d) | - Reads data set in the routing parameter. | $\square$ | 8-7 |
|  | SP_RTREAD |  |  | $\uparrow$ |  |
| Routing information registration | S_RTWRITE | n , (5) | - Writes routing data to the location specified in the routing parameter. | $\sqrt{\square}$ | 8-9 |
|  | SP_RTWRITE |  |  |  |  |

### 2.6.3 Refresh device write/read instructions

Table 2.38 Refresh device write/read instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Refresh device write | S_REFDVWRB | $\mathrm{n} 1, \text { ( ①), (32, n2, }$ <br> (d1) | - Writes data in 1-bit units to the specified refresh device. | $\square$ | 8-12 |
|  | SP_REFDVWRB |  |  | $\uparrow$ |  |
|  | S_REFDVWRW | $\mathrm{n} 1, \text { ( (1) , (s2), n2, }$ <br> (d1) | - Writes data in 16-bit units to the specified refresh device. | $\square$ | 8-17 |
|  | SP_REFDVWRW |  |  | $\uparrow$ |  |
| Refresh device read | S_REFDVRDB | $\mathrm{n} 1, \text { ( (1) , (s2), n2, }$ <br> (d1) | - Reads data in 1-bit units from the specified refresh device. | $\sqrt{\square}$ | 8-23 |
|  | SP_REFDVRDB |  |  | $\uparrow$ |  |
|  | S_REFDVRDW | $\mathrm{n} 1, \text { (s1), (s2), n2, }$ <br> (d1) | - Reads data in 16-bit units from the specified refresh device. | $\square$ | 8-28 |
|  | SP_REFDVRDW |  |  | $\uparrow$ |  |

### 2.7 Multiple CPU Dedicated Instructions

### 2.7.1 Writing data to host CPU shared memory

Table 2.39 Writing data to host CPU shared memory

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Data write to host CPU shared memory | S_TO <br> SP_TO | (31), (32), (3), (34), <br> (d) | - Writes devices of the host station to the CPU shared memory of the host station CPU module. | $\frac{\square}{5}$ | 9-4 |
|  | TO | (5), n1, n2, n3 | - Writes devices of the host station to the CPU shared memory of the host station CPU module. | $\square$ |  |
|  | TOP |  |  | $\uparrow$ |  |
|  | DTO | (5), n1, n2, n3 | - Writes devices of the host station to the CPU shared memory of the host station CPU module in units of 32-bit data. | $\square$ |  |
|  | DTOP |  |  | $\stackrel{\square}{5}$ |  |

### 2.7.2 Reading data from other CPU shared memory

Table 2.40 Reading data from other CPU shared memory

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Data read from other CPU shared memory | FROM | n1, n2, n3, © | - Reads devices from the CPU shared memory of another station CPU module to the host station. | $\square$ | 9-15 |
|  | FROMP |  |  | $\ldots$ |  |
|  | DFRO | n1, n2, n3, (d) | - Reads devices from the CPU shared memory of another station CPU module to the host station in units of 32-bit data. | $\square$ |  |
|  | DFROP |  |  | $\uparrow$ |  |

### 2.8 Multiple CPU High Speed Transmission Dedicated Instructions

### 2.8.1 Multiple CPU high speed transmission dedicated instructions

Table 2.41 Multiple CPU high speed transmission dedicated instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Device data write to other CPU | D_DDWR | $\begin{aligned} & \mathrm{n}, \text { (15), (2), © (1), } \\ & \text { (®2) } \end{aligned}$ | - Writes data stored in the device $\Theta_{2}$ and the following devices in the host CPU at the time of multiple CPU system configuration, to the device @11 and the following devices specified in the other CPU ( $n$ ) for the number of points specified for $(₫ 1)+1$. | $\sqrt{\square}$ | 10-12 |
|  | DP_DDWR | $\begin{aligned} & \mathrm{n}, \text { (11), (22), ©(1), } \\ & \text { (22) } \end{aligned}$ |  |  |  |
| Device data read from other CPU | D_DDRD | $\begin{aligned} & \mathrm{n}, \text { (15), (2), ©(1), } \\ & \text { (®2) } \end{aligned}$ | - Stores data read from the device (22) and the following devices specified in the other CPU ( $n$ ) for the number of points specified for (51) +1 , to the device @ll and the following devices specified in the host CPU at the time of multiple CPU system configuration. | $\sqrt{\square}$ | 10-16 |
|  | DP_DDRD | $\begin{aligned} & \mathrm{n}, \text { (11), (2), @1), } \\ & \text { (22) } \end{aligned}$ |  |  |  |

### 2.9 Redundant System Instructions (For redundant CPU)

### 2.9.1 Redundant system instructions (For redundant CPU)

Table 2.42 Redundant system instructions

| Category | Instruction name | Argument | Processing details | Executing condition | Page |
| :---: | :---: | :---: | :---: | :---: | :---: |
| System switching | SP_CONTSW | (5), (d) | - Switches between the control system and standby system at the END processing of the scan executed with the SP_CONTSW instruction. | $5$ | 11-2 |

MEMO

## CONFIGURATON OF INSTRUCTONS

3.1 Configuration of Instructions ..... 3-2
3.2 Precautions on Programming ..... 3-4
3.3 Executing Conditions for Instructions ..... 3-13
3.4 Operation of OUT Instructions, SET/RST Instructions, or PLS/PLF Instructions Using Same Device ..... 3-14
3.5 Precautions on Using File Registers ..... 3-19

### 3.1 Configuration of Instructions

The instructions available for CPU modules can be divided into an instruction name and arguments.

The application of an instruction name and arguments are as follows:

- Instruction name ?? Indicates the function of the instruction.
- Argument ??????? Indicates the I/O data used in the instruction.

Arguments are classified into source data, destination data, number of devices, executing condition, and execution status.
(1) Source (s)
(a) A source is data used in an operation.
(b) The following source types are available depending on the device specified in an instruction.

- Constant ???????????????????Specifies a numeric value used in an operation. Constants are set during programming so that they cannot be changed while the program is being executed. Perform index setting when using them as variable data.
- Bit device and word device ????Specifies the device in which the data used in the operation are stored. Data must be stored to the specified device before executing the operation. By changing the data to be stored to the specified device while a program is being executed, the data used in the instruction can be changed.
(c) Contacts cannot be directly input to sources that use bit devices.
(2) Destination (d)
(a) Data after the operation are stored to a destination. Some instructions require to store data used in the operation to the destination before the operation.
Example: Addition instruction of 16-bit BIN data

(b) Set a device in which data are to be stored to a destination.
(c) Coils cannot be directly connected to destinations which store bit devices.
(3) Number of devices and number of transfers (n)
(a) The number of devices and number of transfers used in an instruction that uses multiple devices are specified for $n$.
Example: Block data transfer instruction

(b) A value in the range of 0 to 32767 can be set for the number of devices and number of transfers. When the number of devices or number of transfers is 0 , the instruction performs no processing.
(4) Executing condition (EN)

An input variable EN inputs an executing condition of an instruction.
(5) Execution status (ENO)

An output variable ENO outputs an execution status.

## 区POINT

For details of the instruction configurations of labels and structures etc., refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

### 3.2 Precautions on Programming

The following cases cause an operation error during the execution of basic instructions and application instructions in a CPU module.

- Errors described on each page of instruction explanation are occurred.
- No intelligent function module is mounted at the specified I/O number position when an intelligent function module device is used.
- The specified buffer memory address does not exist when an intelligent function module device is used.
- The corresponding network does not exist when a link device is used.
- No network module is mounted at the specified I/O number position when a link device is used.
- A CPU module is not mounted at the start I/O number position of the specified CPU module when a multiple CPU area device is used. (For Universal model QCPU (except for Q00UJCPU) only)
- The specified shared memory address does not exist when a multiple CPU area device is used. (For Universal model QCPU (except for Q00UJCPU) only)
- The setting that crosses internal user devices and extended data registers (D)/extended link registers $(\mathrm{W})$ is set. (For Universal model QCPU (except for Q00UJCPU) and LCPU)


## XPOINT

The following are the results of the file register data write/read when the file register setting is not set, or when the file register setting is set, but a file register file does not exist.
(1) For High Performance model QCPU, Process CPU, and Redundant CPU An error does not occur even when writing/reading data to/from file registers. However, " OH " is stored when reading from file registers.
(2) For Universal model QCPU and LCPU

The OPERATION ERROR (error code: 4101) occurs when writing/reading data to/ from file registers. By setting the PLC parameter to not check the device range, an error is not detected. (Refer to POINT(1).)

## (1) Device range check

The following explains how the range is checked for devices used in basic instructions and application instructions in a CPU module.
(a) Instructions that handle fixed-length devices (such as MOV or DMOV)

1) For Basic model QCPU, High Performance model QCPU, Process CPU, and Redundant CPU

The device range is not checked.
If the data exceed the corresponding device range, the data are written to another devices. *1

For example, when the data registers are assigned in 12K points, an error does not occur even if the data exceed D12287.

The device range is not checked when the index setting is applied. If the result of the index setting exceeds the corresponding device range, the data are written to another devices. *1
*1: For the assignment order of internal user devices, refer to (c) Character string data in this section.
2) For Universal model QCPU and LCPU

The device range is checked. When the data exceed the corresponding device range, an operation error occurs.
For example, when data registers are assigned in 12 K points, an error occurs if the data exceed D12287.


This specifies D12287 and D12288 as the target devices for executing the DMOV instruction. However, since D12288 does not exist, an operation error occurs.

The device range is checked even when the index setting is applied. By changing the settings of the PLC parameter, the device range is not checked. ${ }^{*}$ 2
*2: For changing the setting of device range check, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module to be used.
(b) Instructions that handle variable-length devices (such as BMOV or FMOV that specifies the number of transfers)

1) For Basic model QCPU, High Performance model QCPU, Process CPU, and Redundant CPU

The device range is checked.
When the data exceed the corresponding device range, an operation error occurs.
For example, when data registers are assigned in 12 K points, an error occurs if the data exceed D12287.

$\longrightarrow$ This specifies D12287 and D12288 as the target devices for executing the BMOV instruction. However, since D12288 does not exist, an operation error occurs.

Device range is checked even when the index setting is applied.
However, when the start number of the device exceeds the corresponding device range with the index setting, an error occurs.


## 2) For Universal model QCPU and LCPU

The device range is checked. When the data exceed the corresponding device range, an operation error occurs.
For example, when data registers are assigned in 12 K points, an error occurs if the data exceed D12287.


This specifies D12287 and D12288 as the target devices for executing the BMOV instruction. However, since D12288 does not exist, an operation error occurs.

The device range is checked even when the index setting is applied. Also, when the start number of the device exceeds the corresponding device range with the index setting, an error occurs.


By changing the settings of the PLC parameter, the device range is not checked. ${ }^{*}{ }^{2}$
*2: For changing the setting of device range check, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module to be used.
(c) Character string data

1) For Basic model QCPU, High Performance model QCPU, Process CPU, and Redundant CPU
Since character string data are treated as variable-length data without exception, the device range is checked.
When the data exceed the corresponding device range, an operation error occurs.
For example, when data registers are assigned in 12 K points, an error occurs if the data exceed D12287.


This specifies D12287 and D12288 as the target devices for executing the \$MOV instruction. However, since D12288 does not exist, an operation error occurs.
When the index modification is executed, the device range is checked. However, when the index modification is executed, an error does not occur if the start device number exceeds the corresponding device range. Other devices are accessed.
2) For Universal model QCPU and LCPU

Since character string data are treated as variable-length data without exception, the device range is checked.
When the data exceed the corresponding device range, an operation error occurs.
For example, when data registers are assigned in 12 K points, an error occurs if the data exceed D12287.


When the index modification is executed, the device range is checked. When the index modification is executed, an error occurs if the start device number exceeds the corresponding device range.
The PLC parameter setting enables the device range check not to be performed. ${ }^{*}$
*2 : For the setting method which the device range check is not performed when the index modification is executed, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module used.
(d) When the index setting is applied to direct access output (DY), the device range is checked.
(e) Precautions on using extended data registers (D) and extended link registers (W) (Universal model QCPU and LCPU other than Q00UJCPU)
An index setting that crosses internal user devices and extended data registers (D)/ extended link registers (W) cannot be applied in the following specification methods as shown below. If it is specified, an OPERATION ERROR (Error code: 4101) occurs.

- Index setting
- Indirect specification
- Specification in the instruction that uses block data*1

*1: Block data are the data described below.
- Data used for the instruction in which multiple words (such as FMOV, BMOV, BK+) are the operating target.
- Control data configured with more than two words and specified for SP_FWRITE, SP_FREAD.
- Data format data more than 32 bits. (BIN 32-bit, real number, indirect address of device)
(1) For the Universal model QCPU and LCPU, an error (error code: 4101) occurs when the following access are executed with the instructions or data shown
(a) to (c).
(a) Instructions that handle fixed-length devices (such as MOV or DMOV)
(b) Instructions that handle variable-length devices (such as BMOV or FMOV that specifies the number of transfers)
(C) Character string data
- Data access that exceeds the device boundary by the index modification (range of area A)
The following shows the assignment order of devices.

- Data access that exceeds the file register boundary by the index modification
- Data access to file registers (R, $Z R$ ) when file register files are not set
- Data access to file registers $(R, Z R)$ that exceeds the range of file register files

However, by setting the PLC parameter to not check the device range, an error is not detected if the above access are executed.
For the Universal model QCPU, the operation differs according to the serial number as shown by the following table. ${ }^{*}{ }^{2}$

| Device range setting at | First five digits of the serial number (Universal model QCPU) |  |  |
| :---: | :---: | :---: | :---: |
| index modification | '10021' or earlier | '10022' or later |  |
| Applied | Errors are detected in the data access of 1) to 4) |  |  |
| Not applied | Errors are detected in the data access of 2) to 4) | No error is detected |  |

*2 : For the setting method which the device range check is not performed when the index modification is executed, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module used.
(2) For the Universal model QCPU and LCPU, the index modification that crosses internal user devices (SW) and file registers (R) cannot be applied. (Error code: 4101)

## Remark

For the method for changing the assignment of internal user devices, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module used.
(2) Device data check

The following explains how the data are checked for devices used in basic instructions and application instructions in a CPU module.
(a) BIN data

Even if the operation results in overflow or underflow, an error does not occur, and the carry flag (SM700) is not turned ON.
(b) BCD data

1) Checks whether each digit has a $B C D$ value ( 0 to 9 ). If each digit has a value other than 0 to 9 (A to $F$ ), an operation error occurs.
2) Even if the operation results in overflow or underflow, an error does not occur, and the carry flag (SM700) is not turned ON.
(c) Floating-point data
3) If the operation results are as follows in an instruction that performs single-precision floating-point operation, an operation error occurs.
$1.0 \times 2^{-127}$ or less
$1.0 \times 2^{128}$ or more
4) If the operation results are as follows in an instruction that performs double-precision floating-point operation, an operation error occurs.
$1.0 \times 2^{-1023}$ or less
$1.0 \times 2^{1024}$ or more
(d) Character string data

Data check is not performed.
(3) Buffer memory access

For accessing buffer memories, using instructions with intelligent function module devices (from Un\G0) is recommended.
(4) Multiple CPU shared memory access

For accessing multiple CPU shared memories, using instructions with multiple CPU area devices (from U3En\G10000) is recommended.
(5) Number of characters in character string type label check

In the operation of character string type labels, the number of characters which has been declared to character string type labels is not checked.
When using instructions/functions which process character strings, check not to exceed the number of characters which has been declared to character string type labels.

(6) Branch point of line from destination

When a line is branched, a temporary variable is created on the branch point. When a line which is from destination is branched as shown below, the operation result may not be output to the devices or labels of branch destination correctly since the operation result is stored on a temporary variable which is created on the branch point. For details of temporary variable, refer to GX Works2 Version 1 Operating Manual (Structured Project).


The operation result is stored on a temporary variable.

### 3.3 Executing Conditions for Instructions

This section explains the four types of executing condition for sequence instructions, basic instructions, and application instructions executed in a CPU module.

- Non-conditional execution .... Instructions executed regardless of ON/OFF status of the device.
Example: LD= instruction
- Executed while ON ............. Instructions executed while the input condition is ON. Examples: MOV instruction, FROM instruction
- Executed at rising edge Instructions executed only at the rising edge (OFF to ON) of the input condition.
Example: PLS instruction, MOVP instruction
- Executed at falling edge ....... Instructions executed only at the falling edge (ON to OFF) of the input condition.
Example: PLF instruction

For basic instructions and application instructions that are equivalent to coils, when both 'Executed while ON' and 'Executed at rising edge' are applicable, ' P ' is appended to an instruction name to distinguish executing conditions.

- An instruction executed while ON Instructions name
- An instruction executed at rising edge Instructions name P

The following shows how the executing conditions of 'Executed while ON' and 'Executed at rising edge' are specified for the MOV(P) instruction.


### 3.4 Operation of OUT Instructions, SET/RST Instructions, or PLS/PLF Instructions Using Same Device

This section explains the operation when the OUT instructions, SET/RST instructions, or PLS/ PLF instructions that use a same device are executed multiple times in one scan.
(1) OUT instructions that use a same device

Avoid executing the OUT instructions that use a same device multiple times during one scan. If the OUT instructions that use a same device are executed multiple times during one scan, the specified device is turned ON/OFF according to the operation result obtained by each execution of the OUT instruction. Since the ON/OFF status of the specified device is determined by each execution of the OUT instruction, turning ON/OFF may be repeated during one scan.
The following figures show the operation example of the ladder that turns ON/OFF the same internal relay (M0) by the inputs X0 and X1.
[Ladder]

[Timing chart]


MO turns ON because XO is ON . MO remains OFF because XO is OFF.
For refresh type CPU modules, the ON/OFF status of the OUT instruction executed last during one scan is output when output $(\mathrm{Y})$ is specified for the OUT instruction.
(2) SET/RST instructions that use a same device
(a) The SET instruction turns ON the specified device when the execution command is ON, while it performs no operation when the execution command is OFF.
Thus, if the SET instructions that use a same device are executed multiple times during one scan, the specified device is turned ON when any of the execution commands is turned ON.
(b) The RST instruction turns OFF the specified device when the execution command is ON, while it performs no operation when the execution command is OFF. Thus, if the RST instructions that use a same device is executed multiple times during one scan, the specified device is turned OFF when any of the execution commands is turned ON.
(c) If the SET and RST instructions use the same device in one scan, the SET instruction turns ON the specified device when the execution command is ON, while the RST instruction turns OFF the specified device when the execution command is ON. When the execution command of the SET and RST instructions is OFF, the ON/OFF status of the specified device does not change.
[Ladder]

[Timing chart]


For refresh type CPU modules, the ON/OFF status of the SET/RST instruction executed last during one scan is output when output $(\mathrm{Y})$ is specified for the SET/RST instruction.
(3) PLS instructions that use a same device

The PLS instruction turns ON the specified device when the execution command turns from OFF to ON. When the status is other than turning from OFF to ON, (OFF to OFF, ON to ON or ON to OFF), the PLS instruction turns OFF the specified device.
If the PLS instructions that use a same device are executed multiple times during one scan, the PLS instructions turn ON the specified device when the execution command turns from OFF to ON. In the status other than turning from OFF to ON, they turn OFF the specified device.
Thus, when the PLS instructions that use a same device are executed multiple times during one scan, the devices turned ON in the PLS instructions may not be turned ON for one scan.
[Ladder]


## [Timing chart]

- The ON/OFF timing differs between X0 and X1 (The specified device is not turned ON for one scan)

- The timings of turning from OFF to ON between X0 and X1 are same


For refresh type CPU modules, the ON/OFF status of the PLS instruction executed last during one scan is output when output $(\mathrm{Y})$ is specified for the PLS instruction.
(4) PLF instructions that use a same device

The PLF instruction turns ON the specified device when the execution command turns from ON to OFF. If the status is other than turning from ON to OFF (OFF to OFF, OFF to ON or ON to OFF), the PLF instruction turns OFF the specified device.
If the PLF instructions that use a same device are executed multiple times during one scan, the PLF instructions turn ON the specified device when the execution command turns from ON to OFF. In the status other than turning from ON to OFF, they turn OFF the specified device.
Thus, when the PLF instructions that use a same device are executed multiple times during one scan, the devices turned ON in the PLF instruction may not be turned ON for one scan.
[Ladder]

[Timing chart]

- The ON/OFF timing differs between $\mathrm{X0}$ and X 1 (The specified device is not turned ON for one scan)

- The timings of turning from On to OFF between $\mathrm{X0}$ and X 1 are same


For refresh type CPU modules, the ON/OFF status of the PLF instruction executed last during one scan is output when output $(\mathrm{Y})$ is specified for the PLF instruction.

### 3.5 Precautions on Using File Registers

This section explains the precautions on using file registers in QCPU (Q mode) and LCPU.
(1) CPU module that does not support file registers

Q00JCPU and Q00UJCPU do not support file registers.
To use file registers, use CPU modules other than Q00JCPU and Q00UJCPU.
(2) Setting file registers to be used

To use file registers, they need to be set in the PLC parameter or the QDRSET instruction. (With Q00CPU, Q01CPU or LCPU, they do not need to be set because the PLC parameter is set to 'Use file register'. The QDRSET instruction cannot be used for LCPU.)
If the file registers are not set, normal operations cannot be performed in the instructions that use the file registers.

## XPOINT

Even when file registers are not set in the PLC parameter, a program that uses file registers can be created. For CPU modules other than Universal model QCPU and LCPU, an error does not occur when that program is written to the CPU module.
Note that the data cannot be written/read correctly to/from file registers.
For Universal model QCPU and LCPU, an error occurs if the program that uses file registers is executed.
(3) Reserving the file register area
(a) For Basic model QCPU

The file register area is reserved in the standard RAM in advance. Users do not need to reserve the file register area.
(b) For High Performance model QCPU, Process CPU, Redundant CPU, and Universal model QCPU (except for High-speed Universal model QCPU) To use file registers, register them to the standard RAM/memory card and reserve the file register area.
(c) High-speed Universal model QCPU, and LCPU

To use file registers, register them to the standard RAM and reserve the file register area.

Remark
For the method for setting file registers and the memory which can be used the file registers of each CPU module, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module used.
(4) Specifying the file register number exceeding the registered number
(a) For Basic model QCPU, High Performance model QCPU, Process CPU, and Redundant CPU
An error does not occur when data are written/read to/from the file register numbers that are greater than the registered number of points.
Note that the data cannot be written/read correctly to/from file registers.
(b) For Universal model QCPU and LCPU

When data are written/read to/from the file register numbers that are greater than the registered number of points, an error occurs. (Error code: 4101)
(5) Method for specifying file registers

The methods for specifying file registers are the block switching method and the serial number access method.
(a) Block switching method

The block switching method specifies file register points being used in units of 32 K points (one block).
File registers exceeding 32K points is specified by switching block numbers of the file registers used in the RSET instruction. File register points are specified in the range from R0 to R32767 for each block.

(b) Serial number access method

The serial number access method specifies file registers exceeding 32 K points with serial device numbers. File registers of multiple blocks can be used as continuous file registers. Use 'ZR' as a device name.

(6) Settings and restrictions when specifying file registers to refresh devices
(a) Setting refresh devices

Refresh devices can be set by the following settings.

- Refresh settings of CC-Link IE module (Not available for LCPU)
- Refresh settings of CC-Link IE Field Network (Not available for Basic model QCPU, High Performance model QCPU, Process CPU, Redundant CPU, Universal model QCPU with a serial number whose first five digits are "12011" or earlier, and LCPU with a serial number whose first five digits are "13011" or earlier.)
- Refresh settings of MELSECNET/H (Not available for LCPU)
- Refresh settings of CC-Link
- Auto refresh settings of the intelligent function module
- Auto refresh settings of the multiple CPU system (Not available for LCPU)
(b) Restrictions

The following are the restrictions when specifying file registers to refresh devices.

1) For QCPU (Q mode), the refresh cannot be performed correctly if the file register which has the same name as the program is specified by the PLC parameter.
When the file register whose file name is same as the program name is used, the refresh is performed to the file register whose file name is same as the program name that is set at the last number in the program settings. To read or write the refresh data, use the QDRSET instruction to switch to the corresponding file register.
2) The refresh cannot be performed correctly if the file name of file register or drive number is changed by using the QDRSET instruction.
(The QDRSET instruction cannot be used for LCPU.)
When the file name or drive number is changed on QDRSET instruction, the link refresh is performed to the set file when the END processing is carried out. To read or write the refresh data, specify the set file at the END processing. Note that If the drive number is changed by using the QDRSET instruction when " $Z R$ " is specified for the device in a CPU module other than Universal model QCPU, an error (LINK PARA ERROR (3101)) occurs. (If "R" is specified for the device, an error does not occur.)
3) When a block number is selected by using the RSET instruction, the refresh is performed to the file register $(R)$ in the selected block number. When a block number is switched on RSET instruction, the refresh is performed to the file register ( R ) in the block number when the END processing is carried out. To read or write the refresh data, specify the file register of the block number at the END processing.
(7) Precautions on using file registers in the flash memory

This section explains the precautions on the flash memory that can use file registers.
(a) The following flash memory is used.

- Flash card
(b) File registers in a flash memory can be only read in sequence programs.
(File registers cannot be written to a flash memory in sequence programs.)


When using a flash memory for file registers, write data to a flash memory in advance.
(c) Use the programming tool to write data to a flash card.

## HOW TO READ INSTRUCTIONS

Chapter 5 to 10 provide detailed explanation on each instruction in the layout as shown below.

(1) Indicates a section number and an outline of an instruction.
(2) Indicates an instruction to be explained.
(3) Indicates the CPU modules that can use the instruction

| Icon |  |  |  |  |  | Description |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Basic model QCPU | High Performance model QCPU | Process CPU | Redundant CPU | Universal model QCPU | LCPU |  |
| Basic |  | Process | Redundant | Universal | LCPU | Icons without any marks indicate that the CPU module can use the corresponding instructions. |
|  | Ver. <br> $\substack{\text { High } \\ \text { Hefromance }}$ |  |  | Ver. Universal | $\begin{gathered} \text { Ver. } \\ \text { LCPU } \end{gathered}$ | Icons with a "Ver." mark indicate that the CPU module can use the corresponding instructions under certain restrictions. related to function version and software version. |
|  | $\underset{\substack{\text { High } \\ \text { pefromance }}}{\sim}$ |  |  |  |  | Icons with an " $\times$ " mark indicate that the CPU module cannot use the corresponding instructions. |

(4) Indicates the instruction name and executing condition of the instruction.

| Executing condition | Non- <br> conditional <br> execution | Executed while <br> ON | Executed once at <br> ON | Executed while <br> OFF | Executed once at <br> OFF |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Symbols on the <br> corresponding page | No symbol | - | - | - | - |

(5) Indicates the instruction names that can be described.
(6) Indicates the description format of the instruction in the structured ladder/FBD/ST language.
(7) Indicates the names of input and output arguments, and the data type of each argument. For details of each data type, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
(8) Devices that can be used in the instruction are marked with $\bigcirc$.

The following table shows applicable classification for usable devices.

| Device classification | Internal device (system, user) |  | File register R, ZR | Link direct device ${ }^{* 4}{ }^{*} 6$ J荿: |  | Intelligent function moduleU:...IG:...: | $\begin{aligned} & \text { Index } \\ & \text { register } \mathrm{Zn} \end{aligned}$ | Constant ${ }^{* 5}$ | Others ${ }^{* 5}$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Bit | Word |  | Bit | Word |  |  |  |  |
| Usable *1 device | $\begin{aligned} & \mathrm{X}, \mathrm{Y}, \mathrm{M}, \\ & \mathrm{~L}, \mathrm{SM}, \mathrm{~F}, \\ & \mathrm{~B}, \mathrm{SB}, \\ & \mathrm{FX}, \mathrm{FY}^{* 2} \end{aligned}$ | $\begin{aligned} & \text { T, ST, C, *3 } \\ & \text { D, W, SD, } \\ & \text { SW, FD, } \\ & \text { @ } \end{aligned}$ | R, ZR |  | $\begin{aligned} & \text { JinW } \\ & \text { JISWW } \end{aligned}$ |  | Z | K, H, E, \$ | $\begin{aligned} & \text { P, I, J, U, } \\ & \text { DX, DY, } \\ & \mathrm{N}, \\ & \mathrm{BL}, \mathrm{TR}, \\ & \mathrm{BLIS}, \mathrm{~V} \end{aligned}$ |

*1: For description of each device, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module used.
*2: FX and FY can be used in bit data only, and FD can be used in word data only.
*3: T, ST or C can be used in word data only (they cannot be used in bit data) when they are used in the instructions other than the following instructions.
[Usable instructions in bit data] LD, LDI, AND, ANI, OR, ORI, LDP, LDF, ANDP, ANDF, ORP, ORF, LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI, OUT, RST, and BKRST
*4: These devices can be used in CC-Link IE, MELSECNET/H, and MELSECNET/10
*5: The Constant and Others columns describe settable devices.
*6: Link direct devices (J.ji.j) cannot be used for LCPU.
(9) Indicates the processing performed by the instruction.
(10) Indicates whether to exist the related error. When an error exists, conditions that cause an error are described.
(11) Indicates program examples in the structured ladder/FBD/ST language.

## SEQUENCE INSTRUCTIONS

5.1 Contact Instructions ..... 5-2
5.2 Bond Instructions ..... 5-12
5.3 Output Instructions ..... 5-24
5.4 Shift Instructions ..... 5-51
5.5 Master Control Instructions ..... 5-55
5.6 End Instructions ..... 5-60
5.7 Other Instructions ..... 5-62

### 5.1 Contact Instructions

### 5.1.1 Operation start, series connection, parallel connection

 LD, LDI, AND, ANI, OR, ORI
(S) : Devices used as contacts (Bit)

| Setting data |  | vice | R, ZR | J...1.a |  | U...igal | Zn | Constant | Others <br> DX |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Bit | Word |  | Bit | Word |  |  |  |  |
| (s) | $\bigcirc$ |  |  |  |  |  | - |  | $\bigcirc$ |

®POINT
When BL, S, TR, BLIS, or BLITR is used, refer to SFC control instructions of the MELSEC-Q/L/QnA Programming Manual (SFC).

## LD, LDI

(1) LD is the normally open operation start instruction, and LDI is the normally closed operation start instruction. They read ON/OFF information from the specified device*1, and use that as the operation result.
*1 : For a bit-specified word device, the device turns ON or OFF depending on the $1 / 0$ status of the specified bit.

## AND, ANI

(1) AND is the normally open series connection instruction, and ANI is the normally closed series connection instruction. They read the ON/OFF data of the specified bit device*2, perform an AND operation on that data and the operation result to that point, and take this value as the operation result.
*2 : For a bit-specified word device, the device turns ON or OFF depending on the $1 / 0$ status of the specified bit.

## OR, ORI

(1) OR is the normally open single parallel connection instruction, and ORI is the normally closed single parallel connection instruction. They read ON/OFF information from the specified device*3, and perform an OR operation with the operation results to that point, and use the resulting value as the operation result.
*3 : For a bit-specified word device, the device turns ON or OFF depending on the $1 / 0$ status of the specified bit.

Remark
A bit of word device is specified in hexadecimal.
Bit b11 of D0 would be D0.B.
For more information on bit specification of word device, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

## 0 <br> Operation Error

No operation error occurs in the execution of the LD, LDI, AND, ANI, OR, and ORI instructions.

## $\square$ Program Example

(1) The program using the LD, AND, OR, and ORI instructions.
[Structured ladder/FBD]

(2) The program connecting contacts using the ANB and ORB instructions.
[Structured ladder/FBD]

(3) The parallel program of the OUT instructions.
[Structured ladder/FBD]


### 5.1.2 Edge operation start, edge series connection, edge parallel connection

LDP, LDF, ANDP, ANDF, ORP, ORF




| instructions. |  |
| :--- | :--- |
| LDP | LDF |
| ANDP | ANDF |
| ORP | ORF |
|  |  |
|  |  |


| Input argument, | EN: | Executing condition | :Bit |
| :--- | :--- | :--- | :--- |
|  | $\mathrm{s}:$ | Devices used as contacts | :Bit |
| Output argument, | ENO: | Execution result | Bit |


| Setting data | Internal device |  | R, ZR | J.ala |  | U...ida | Zn | Constant | Others DX |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Bit | Word |  | Bit | Word |  |  |  |  |
| (s) |  |  | $\bigcirc$ |  |  |  | - |  | $\bigcirc$ |

## Function

## LDP, LDF

(1) LDP is the rising edge operation start instruction, and it is ON only at the rising edge of the specified bit device (when it turns from OFF to ON). For a bit-specified word device, it is ON only when the specified bit changes from 0 to 1.
In cases where there is only an LDP instruction, it acts identically to instructions for the creation of a pulse that are executed while $\mathrm{ON}(\ldots \mathrm{P})$.

(2) LDF is the falling edge operation start instruction, and it is ON only at the falling edge of the specified bit device (when it turns from ON to OFF).
For a bit-specified word device, it is ON only when the specified bit changes from 1 to 0 .

## ANDP, ANDF

(1) ANDP is a rising edge series connection instruction, and ANDF is a falling edge series connection instruction. They perform AND operations with the operation result to that point, and take the resulting value as the operation result.
The ON/OFF data used by ANDP and ANDF are indicated in the table below.

| Device specified in ANDP or ANDF |  |  | ANDP status |
| :---: | :---: | :---: | :---: | ANDF status

## ORP, ORF

(1) ORP is a rising edge parallel connection instruction, and ORF is a falling edge serial connection instruction. They perform OR operations with the operation result to that point, and take the resulting value as the operation result.
The ON/OFF data used by ORP and ORF are indicated in the table below.

| Device specified in ORP or ORF |  | ORP status | ORF status |
| :---: | :---: | :---: | :---: |
| Bit device | Bit-specified word <br> device |  |  |
| OFF to ON | 0 to 1 |  | OFF |
| OFF | 0 | OFF |  |
| ON | 1 |  |  |
| ON to OFF | 1 to 0 |  | ON |

(2) The ORP or ORF instruction performs comparison operation of operation result between (s) and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400.
When using the ORP or ORF instruction, connect EN and ENO in series as shown below.


## Operation Error

No operation error occurs in the execution of the LDP, LDF, ANDP, ANDF, ORP, and ORF instructions.

## $\square$ Program Example

In the following program, the MOV instruction is executed at the rising edge of $g_{-}$bool 1 and g_bool2, or b10 (bit 10) of data register D0.
[Structured ladder/FBD (Symbolic description)]

[Structured ladder/FBD (Functional description)]

[ST (When using IF syntax)]
IF (LDP(TRUE,g_bool1) AND LDP(TRUE,g_bool2)) OR (LDP(TRUE,D0.A)) THEN MOV(TRUE,0,g_int1);
END_IF;
[ST (When not using IF syntax)]
MOV(ORP(ANDP(LDP(TRUE,g_bool1),g_bool2),D0.A),0,g_int1);
5.1.3 Negated edge operation start, negated edge series connection, negated edge pulse parallel connection

## LDPI, LDFI, ANDPI, ANDFI, ORPI, ORFI <br> 

- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported
$L D(P)(F) \mid$
AND(P)(F)!
$\operatorname{OR}(P)(F)!$


## Executing condition

OR(P)(F)I


## Structured ladder/FBD (Symbolic description) Bit device number/bit-specified word device (S)



ORFI


| Lins................... | indicates any of the following |
| :--- | :--- |
| instructions. |  |
| LDPI | LDFI |
| ANDPI | ANDFI |
| ORPI | ORFI |
|  |  |


| Input argument, | EN: | Executing condition | :Bit |
| :--- | :--- | :--- | :--- |
|  | $\mathrm{s}:$ | Devices used as contacts | :Bit |
| Output argument, | ENO: | Execution result | :Bit |


| Setting data | Internal device |  | R, ZR | J\%el |  | U\%1G | Zn | Constant | Others DX |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Bit | Word |  | Bit | Word |  |  |  |  |
| (s) | $\bigcirc$ | - | $\bigcirc$ | $\bigcirc$ | - | $\bigcirc$ |  |  | $\bigcirc$ |

## Function

## LDPI, LDFI

(1) LDPI is a negated rising edge operation start instruction, and it is ON when the specified bit device is OFF or ON, or at the falling edge of the specified bit device (when it turns from ON to OFF). For a bit-specified word device, it is ON only when the specified bit is 0 or 1 , or it changes from 1 to 0 .
(2) LDFI is a negated falling edge operation start instruction, and it is ON when the specified bit device is OFF or ON, or at the rising edge of the specified bit device (when it turns from OFF to ON ). For a bit-specified word device, it is ON only when the specified bit is 0 or 1 , or it changes from 0 to 1.

| Device specified in LDPI or LDFI |  | LDPI status | LDFI status |
| :---: | :---: | :---: | :---: |
| Bit device | Bit-specified word <br> device |  | ON |
| OFF to ON | 0 to 1 | OFF | ON |
| OFF | 0 | ON | ON |
| ON | 1 | ON | ON |
| ON to OFF | 1 to 0 | ON | OFF |

## ANDPI, ANDFI

(1) ANDPI is a negated rising edge series connection instruction, and ANDFI is a negated falling edge series connection instruction. They perform AND operations with the operation result to that point, and take the resulting value as the operation result.
The ON/OFF data used by ANDPI and ANDFI are indicated in the table below.

| Device specified in ANDPI or ANDFI |  | ANDPI status | ANDFI status |
| :---: | :---: | :---: | :---: |
| Bit device | Bit-specified word <br> device |  |  |
| OFF to ON | 0 to 1 | OFF | ON |
| OFF | 0 | ON | ON |
| ON | 1 | ON | ON |
| ON to OFF | 1 to 0 | ON | OFF |

## ORPI, ORFI

(1) ORPI is a negated rising edge parallel connection instruction, and ORFI is a negated falling edge serial connection instruction. They perform OR operations with the operation result to that point, and take the resulting value as the operation result.
The ON/OFF data used by ORPI and ORFI are indicated in the table below.

| Device specified in ORPI or ORFI |  | ORPI status | ORFI status |
| :---: | :---: | :---: | :---: |
| Bit device | Bit-specified word <br> device |  | OF |
| OFF to ON | 0 to 1 | OFF | ON |
| OFF | 0 | ON | ON |
| ON | 1 | ON | ON |
| ON to OFF | 1 to 0 | ON | OFF |

(2) The ORPI or ORFI instruction performs comparison operation of operation result between (s) and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400. When using the ORPI or ORFI instruction, connect EN and ENO in series as shown below.


## Operation Error

No operation error occurs in the execution of the LDPI, LDFI, ANDPI, ANDFI, ORPI, and ORFI instructions.

## $\square$ Program Example

(1) In the following program, the value 0 is stored to Var_D0 in any of the following status: XO is ON/OFF, X0 turns from ON to OFF, M0 is ON/OFF, or M0 turns from OFF to ON.
[Structured ladder/FBD (Symbolic description)]

[Structured ladder/FBD (Functional description)]

[ST (When using IF syntax)]
IF (LDPI(TRUE,X0) OR LDFI(TRUE,M0)) THEN
MOV(TRUE,0,Var_D0);
END_IF;
[ST (When not using IF syntax)]
MOV(ORFI(LDPI(TRUE,X0),M0),0,Var_D0);
(2) In the following program, the value 0 is stored to Var_D0 when X0 is ON and b10 (bit 10) of D10 is ON/OFF or turns from ON to OFF.
[Structured ladder/FBD (Symbolic description)]

[Structured ladder/FBD (Functional description)]

[ST (When using IF syntax)]
IF (X0 \& LDPI(TRUE,D0.A)) THEN MOV(TRUE,0,Var_D0);
END_IF;
[ST (When not using IF syntax)]
MOV(ANDPI(TRUE,X0),D0.A),0,D5);

### 5.2 Bond Instructions

### 5.2.1 Ladder block series connection and parallel connection

ANB, ORB


## Function

## ANB

(1) Performs an AND operation on block $A$ and block $B$, and takes the resulting value as the operation result.
(2) The symbol for ANB is not the contact symbol, but rather is the connection symbol.

## ORB

(1) Conducts an OR operation on Block $A$ and Block $B$, and takes the resulting value as the operation result.
(2) ORB is used to perform parallel connections for ladder blocks with two or more contacts. For ladder blocks with only one contact, use OR or ORI.
[Structured ladder/FBD]

(3) The ORB symbol is not the contact symbol, but rather is the connection symbol.

## Operation Error

No operation error occurs in the execution of the ANB and ORB instructions.

The program using the ANB and ORB instructions.
[Structured ladder/FBD]


### 5.2.2 Operation result push/read/pop

MPS, MRD, MPP

MPS
MRD
MPP


## Function

## MPS

(1) Stores the memory of the operation result (ON or OFF) immediately prior to the MPS instruction.
(2) Up to 16 MPS instructions can be used successively.

If the MPP instruction is used during this process, the number of uses calculated for the MPS instruction will be decremented by one.

## MRD

(1) Reads the operation result stored in the MPS instruction, and uses that result to perform the operation in the next step.

## MPP

(1) Reads the operation result stored in the MPS instruction, and uses that result to perform the operation in the next step.
(2) Clears the operation results stored in the MPS instruction.
(3) Subtracts 1 from the number of MPS instruction times of use.

1. The following shows the ladders using and without using the MPS, MRD, and MPP instructions.

2. The MPS and MPP instructions must be used the same number of times.

## O Operation Error

No operation error occurs in the execution of the MPS, MRD and MPP instruction.

## Program Example

(1) The program using the MPS, MRD, and MPP instructions.
[Structured ladder/FBD]


## [ST]

OUT(MPS(g_bool1 AND g_bool2) AND g_bool3,g_bool4);
OUT(MRD(TRUE) AND g_bool5,g_bool6);
OUT(MRD(TRUE) AND g_bool7,g_bool8);
OUT(MPP(TRUE),g_bool9);
(2) The program using the MPS and MPP instructions successively. [Structured ladder/FBD]

[ST]
OUT(MPS(MPS(MPS(MPS(g_bool1) AND g_bool2) AND g_bool3) AND g_bool4)
AND g_bool5,g_bool6);
OUT(MPP(TRUE),g_bool7);
OUT(MPP(TRUE),g_bool8);
OUT(MPP(TRUE),g_bool9);
OUT(MPP(TRUE),g_bool10);
OUT(MPP(TRUE),g_bool11);

### 5.2.3 Operation result inversion

INV


## $\Sigma$ Function

Inverts the operation result immediately prior to the INV instruction.

| Operation result immediately prior to the <br> INV instruction | Operation result following the execution of <br> the INV instruction |
| :---: | :---: |
| OFF | ON |
| ON | OFF |

## 0 Operation Error

No operation error occurs in the execution of the INV instruction.

## $\square$ Program Example

The program which inverts the X0 ON/OFF data, and outputs from Y10.
[Structured ladder/FBD]


## [ST]

OUT(INV(X0),Y10);
[Timing chart]

®POINT

1. 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 AND. The INV instruction cannot be used at the LD and OR positions.
2. When using a ladder block, the operation result is inverted within the range of the ladder block.
To operate a ladder using the INV instruction in combination with the ANB instruction, pay attention to the range which is inverted.


For details of the ANB instruction, refer to Section 5.2.1.

### 5.2.4 Pulse conversion of operation result

MEP
MEF


Function
MEP
(1) If operation results up to the MEP instruction are rising edge (from OFF to ON), turns ON (conduction state).
If operation results up to the MEP instruction are anything other than rising edge, turns OFF (non-conduction state).
(2) Use of the MEP instruction simplifies pulse conversion processing when multiple contacts are connected in series.
MEF
(1) If operation results up to the MEF instruction are falling edge (from ON to OFF), turns ON (conduction state).
If operation results up to the MEF instruction are anything other than falling edge, turns OFF (non-conduction state).
(2) Use of the MEF instruction simplifies pulse conversion processing when multiple contacts are connected in series.

## Operation Error

No operation error occurs in the execution of the MEP and MEF instructions.

## $\triangle$ Program Example

The program which performs pulse conversion to the operation results of X 0 and X 1
[Structured ladder/FBD]


```
[ST]
IF X0 AND X1 THEN
    SET(MEP(TRUE),M0);
END_IF;
```


## XPOINT

1. The MEP and MEF instructions will occasionally not function properly when pulse conversion is conducted for a contact that has been indexed by a subroutine program or FOR to NEXT instruction loop. If pulse conversion is to be conducted for a contact that has been indexed by a subroutine program or FOR to NEXT instruction loop, use the EGP/EGF instructions.
2. Because the MEP and MEF instructions operate with the operation results immediately prior to the MEP and MEF instructions, AND should be used at the same position. The MEP and MEF instructions cannot be used at the LD or OR position.

### 5.2.5 Pulse conversion of edge relay operation result

EGP, EGF

```
Basic High Process Redundant Universal LCPU

EGP
EGF

EGP: Executing condition
EGF: Executing condition
\begin{tabular}{ll}
\hline EGP & \\
EGF & \(: \AA\) \\
EGP: Executing condition & \(: \neq\) \\
EGF: Executing condition &
\end{tabular}


\section*{Function}

\section*{EGP}
(1) Operation results up to the EGP instruction are stored in memory by the edge relay (V).
(2) Turns ON (conduction state) at the rising edge (OFF to ON) of the operation result up to the EGP instruction.
If the operation result up to the EGP instruction is other than a rising edge (ON to ON, ON to OFF, or OFF to OFF), it turns OFF (non-conduction state).
(3) The EGP instruction is used when conducting pulse operations for indexed programs in subroutine programs and FOR to NEXT instruction loop.
(4) The EGP instruction can be used like AND.

\section*{EGF}
(1) Operation results up to the EGF instruction are stored in memory by the edge relay (V).
(2) Turns ON at the falling edge (from ON to OFF) of the operation result up to the EGF instruction.
If the operation result up to the EGF instruction is other than a falling edge (OFF to ON, ON to ON, or OFF to OFF), it turns OFF (non-conduction state).
(3) The EGF instruction is used when conducting pulse operations for indexed programs in subroutine programs and FOR to NEXT instruction loop.
(4) The EGF instruction can be used like AND.

\section*{O Operation Error}

No operation error occurs in the execution of the EGP and EGF instructions.

\section*{\(\triangle\) Program Example}

The program using the EGP instruction.
[Structured ladder/FBD]

[ST]
INC(EGP(XOZO,VOZO),DOZO);
[Operation]


\section*{XPOINT}
1. Since the EGP and EGF instructions are executed according to the operation results performed immediately before the EGP and EGF instructions, these instructions must be used at the same position as AND. (Refer to Section 5.1.1.)

The EGP and EGF instructions cannot be used at the LD or OR position.
2. EGP and EGF instructions cannot be used at the ladder block positions shown below.


\subsection*{5.3 Output Instructions}

\subsection*{5.3.1 Out (excluding timers, counters and annunciators)}

OUT


Input argument, Output argument,

\section*{EN:}

ENO: Execution result
d : \(\quad\) Device number to be turned ON or OFF


\section*{3 Function}
(1) Operation results up to the OUT instruction are output to the specified device.
(a) For bit devices
\begin{tabular}{|c|c|}
\hline Operation result & Coil \\
\hline OFF & OFF \\
\hline ON & ON \\
\hline
\end{tabular}
(b) For bit-specified word devices
\begin{tabular}{|c|c|}
\hline Operation result & Bit specified \\
\hline OFF & 0 \\
\hline ON & 1 \\
\hline
\end{tabular}

\section*{Operation Error}

No operation error occurs in the execution of the OUT instruction.
\(\triangle\) Program Example
(1) For bit devices
[Structured ladder/FBD]

[ST]
OUT(X5,Y33);
OUT(X6,Y34);
OUT(X6,Y35);
(2) For bit-specified word devices
[Structured ladder/FBD]

[ST]
OUT(X5,D0.5);
OUT(X6,D0.6);
OUT(X6,D0.7);

\subsection*{5.3.2 Timer}

OUT_T
OUTH_T

inn instructions.

OUT_T
OUTH_T

Input argument,
Executing condition
:Bit
s1 (TCoil): Timer number
s2 (TValue): \(\quad\) Timer set value (Setting range: 1 to 32767 )
Output argument,
ENO:
Execution result
:Bit
:ANY16
:Bit
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..f)} & \multirow[b]{2}{*}{U...1a} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & O(TC only) & - & - & - & & - & - & - & - \\
\hline (52) & - & (Other than Tor C) & \(\bigcirc\) & - & & \(\bigcirc\) & - & \(\bigcirc\) & - \\
\hline
\end{tabular}

\section*{\(\sqrt{3}\) Function}
(1) When the operation results up to the OUT(H)_T instruction are ON, the timer coil turns ON and the timer counts up to the value that has been set, when the time up status (total numeric value is equal to or greater than the setting value), the contact responds as follows.
\begin{tabular}{c|c}
\hline Normally open contact & Timer conduction \\
\hline Normally closed contact & Timer non-conduction \\
\hline
\end{tabular}
(2) The contact responds as follows when the operation result up to the OUT(H)_T instruction is a change from ON to OFF.
\begin{tabular}{l|c|c|c|c|c|c|c} 
& Timer coil & \begin{tabular}{c} 
Current value of \\
timer
\end{tabular} & \multicolumn{2}{|c|}{\begin{tabular}{c} 
Prior to time up \\
\cline { 4 - 7 }
\end{tabular}} & \begin{tabular}{c} 
Normally open \\
contact
\end{tabular} & \begin{tabular}{c} 
Normally \\
closed \\
contact
\end{tabular} & \begin{tabular}{c} 
Normally open \\
contact
\end{tabular}
\end{tabular} \begin{tabular}{c} 
Normally \\
closed contact
\end{tabular}
(3) To clear the current value of a retentive timer and turn the contact OFF after time up, use the RST instruction.
(4) A negative number \((-32768\) to -1\()\) cannot be set as the setting value for the timer. If the set value is 0 , the timer will time out when the time the OUT(H)_T instruction is executed.
(5) The following processing is conducted when the OUT(H)_T instruction is executed.
- OUT(H)_T T. coil turned ON or OFF
- OUT(H)_T T contact turned ON or OFF
- OUT(H)_T T.j current value updated

In cases where the JMP instruction or the like is used to jump to the OUT(H)_T instruction while the \(\mathrm{OUT}(\mathrm{H})_{\mathbf{1}} \mathrm{T}\) instruction is ON , no current value update or contact ON/OFF operation is conducted.
Also, if the same OUT(H)_T instruction is conducted two or more times during the same scan, the current value of the number of repetitions executed will be updated.
(6) Index setting for timer coils or contacts can be conducted only by Z0 or Z 1 .

Timer setting value has no limitation for index setting.

Remark
Timer's time limit
Time limit of the timer is set in the PLC system of the PLC parameter dialog box.
\begin{tabular}{l|c|c|c|c}
\multirow{2}{*}{ Type of timer } & \multicolumn{2}{|c|}{\begin{tabular}{c} 
Basic model QCPU, \\
High Performance QCPU
\end{tabular}} & \multicolumn{2}{c}{ Universal model QCPU, LCPU } \\
\cline { 2 - 5 } & Setting range & \begin{tabular}{c} 
Setting \\
unit
\end{tabular} & Setting range & \begin{tabular}{c} 
Setting \\
unit
\end{tabular} \\
\hline \begin{tabular}{l} 
Low speed timer \\
Low speed retentive timer
\end{tabular} & \begin{tabular}{c}
1 ms to 1000 ms \\
(Default: 100 ms )
\end{tabular} & 1 ms & \begin{tabular}{c}
1 ms to 1000 ms \\
(Default: 100 ms )
\end{tabular} & 1 ms \\
\hline \begin{tabular}{l} 
High speed timer \\
High speed retentive timer
\end{tabular} & \begin{tabular}{c}
0.1 ms to 100 ms \\
(Default: 10 ms )
\end{tabular} & 0.1 ms & \begin{tabular}{c}
0.01 ms to 100.0 ms \\
(Default: 10.0 ms )
\end{tabular} & 0.01 ms \\
\hline
\end{tabular}

For information on timer counting methods, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.

\section*{Operation Error}

No operation error occurs in the execution of the OUT(H)_T instruction.

\section*{Caution}
(1) When creating a program in which the operation of the timer contact triggers the operation of other timer, create the program for the timer that operates later first.
In any of the following cases, all timers turn ON at the same scan if the program is created in the order the timers operate.
- If the set value is smaller than a scan time.
- If 1 is set

\section*{Example}
- For timers T0 to T2, the program is created in the order the timer operates later.

- For timers T0 to T2, the program is created in the order of timer operation.

(2) When using timer devices in a program, it is necessary to separate the usage of devices into following classification depending on the position of use.
When T or ST is entered, they are treated as TN or STN respectively.
- When using it for a contact: TS, STS
- When using it for a coil: TC, STC
- When using it for a current value: TN, STN

\section*{\(\triangle\) Program Example}
(1) In the following program, Y 10 and Y 14 are turned \(\mathrm{ON}, 10\) seconds after X 0 has turned ON . [Structured ladder/FBD]

[ST]
OUT_T(X0,TC1,100);
OUT(TS1,Y10);
OUT(TS1,Y14);
*2: The setting value of the low-speed timer indicates its default time limit ( 100 ms ).
(2) In the following program, the BCD data of X10 to X1F are set as the timer's setting value. [Structured ladder/FBD]

[ST]
BINP(X0,K4X10,Var_D10);
OUT_T(X2,TC2,Var_D10);
OUT(TS2,Y15);
(3) In the following program, Y10 is turned ON, 250 milliseconds after X0 has turned ON. [Structured ladder/FBD]

[ST]
OUTH_T(X0,TC0,25); OUT(TS0,Y10);

\footnotetext{
*3: The setting value of the high-speed timer indicates its default time limit (10ms).
}

\subsection*{5.3.3 Counter}

\section*{OUT_C}

\begin{tabular}{|l|l|}
\hline OUT_C \\
\\
\\
\\
\\
\\
\\
\hline
\end{tabular}
\begin{tabular}{llll} 
Input argument, & EN: & Executing condition & :Bit \\
& s1 (TCoil): & Counter number & :Bit \\
& s2 (TValue): & Counter set value (Setting range: 1 to 32767) & :ANY16 \\
Output argument, & ENO: & Execution result & :Bit
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, Z} & \multicolumn{2}{|c|}{} & \multirow[b]{2}{*}{U} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & O(ccoonly) & - & - & - & & - & - & - & - \\
\hline (52) & - & \[
\begin{aligned}
& \text { O(Other than } T \\
& \text { or C) }
\end{aligned}
\] & \(\bigcirc\) & - & & & - & \(\bigcirc\) & - \\
\hline
\end{tabular}

\section*{Function}
(1) When the operation results up to the OUT_C instruction change from OFF to ON, 1 is added to the current value (count value) and the count up status (current value \(\geqq\) set value), and the contacts respond as follows:
\begin{tabular}{c|c}
\hline Normally open contact & Conduction \\
\hline Normally closed contact & Non-conduction \\
\hline
\end{tabular}
(2) No count is performed with the operation results at ON. (There is no need to perform pulse conversion on count input.)
(3) After the count up status is reached, there is no change in the count value or the contacts until the RST instruction is executed.
(4) A negative number \((-32768\) to -1\()\) cannot be set as the setting value for the timer. If the set value is 0 , the processing is identical to that which takes place for 1 .
(5) Index setting for the counter coil and contact can use only Z0 and Z1.

Counter setting value has no limitation for index setting.
(6) When using counter devices in a program, it is necessary to separate the usage of devices into following classification depending on the position of use.
When C is entered, it is treated as CN .
- When using it for a contact: CS
- When using it for a coil: CC
- When using it for a current value: CN

Remark
For counter counting methods, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.

\section*{O Operation Error}

No operation error occurs in the execution of the OUT_C instruction.

\section*{\(\triangle\) Program Example}
(1) In the following program, Y 30 is turned ON after XO has turned ON 10 times, and the counter is reset when X 1 turns ON .
[Structured ladder/FBD]

[ST]
OUT_C(X0,CC10,10);
OUT(CS10,Y30);
(2) In the following program, the setting value for C 10 is set at 10 when X 0 turns ON , and at 20 when X1 turns ON.
[Structured ladder/FBD]

[ST]
IF X0 AND NOT(X1) THEN
MOVP(TRUE,10,Var_D0);
END_IF;
IF NOT(X0) AND X1 THEN
MOVP(TRUE,20,Var_D0);
END_IF;
OUT_C(X3,CC10,Var_D0);
OUT(CS10,Y30);

\subsection*{5.3.4 Annunciator output}

Process
Redundant Universal LCPU

\section*{OUT}

\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline Input argument, Output argument, & \begin{tabular}{l}
EN: \\
ENO: \\
d:
\end{tabular} & \multicolumn{5}{|l|}{\begin{tabular}{l}
Executing condition \\
Execution result \\
Annunciator number to be turned ON
\end{tabular}} & \multicolumn{4}{|l|}{\begin{tabular}{l}
:Bit \\
:Bit \\
:Bit
\end{tabular}} \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{\(\mathrm{R}, \mathrm{ZR}\)} & \multicolumn{2}{|c|}{Jabind} & \multirow[t]{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (d) & OF only & & & & & - & & & \\
\hline
\end{tabular}
(1) Operation results up to the OUT instruction are output to the specified annunciator.
(2) The following responses occur when an annunciator \((F)\) is turned ON .
[With Q3A, Q4A, or Q4ARCPU]
- The "USER" LED*1 turns ON.
- The annunciator numbers which are ON (F numbers) are stored to special registers (SD64 to SD79).
- The value of SD63 is incremented by 1 .
(3) If the value of SD63 is 16 (which happens when 16 annunciators are already ON), even if a new annunciator is turned ON, its number will not be stored to SD64 to SD79.
(4) The following responses occur when the annunciator is turned OFF by the OUT instruction. The coil turns OFF, but there are no changes in the status of the "USER" LED** and the content of the values stored to SD63 to SD79.
Use the RST instruction to make the "USER" LED*1 turn OFF as well as to delete the annunciator which was turned OFF by the OUT instruction from SD63 to SD79.
*1: For Basic model QCPU, the "ERR." LED displays on the front of the CPU module.

For details of annunciators, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module to be used.

\section*{Operation Error}

No operation error occurs in the execution of the OUT instruction.

\section*{\(\triangle\) Program Example}

In the following program, F 7 is turned ON when X 0 turns ON , and the value 7 is stored to the devices from SD64 to SD79.
[Structured ladder/FBD]

[ST]
OUT(X0,F7);
[Operation]
\begin{tabular}{|c|c|c|c|}
\hline \multirow[b]{2}{*}{SD63} & \multicolumn{3}{|c|}{XO ON} \\
\hline & 0 & Adds \(1 \longrightarrow\) SD63 & 1 \\
\hline SD64 & 0 & SD64 & 7 \\
\hline SD65 & 0 & SD65 & 0 \\
\hline SD66 & 0 & SD66 & 0 \\
\hline SD67 & 0 & SD67 & 0 \\
\hline & ) & & ) \\
\hline SD79 & 0 & SD79 & 0 \\
\hline
\end{tabular}

\subsection*{5.3.5 Setting devices (excluding annunciators)}

SET

\begin{tabular}{lll} 
Input argument, & EN: & Executing condition \\
Output argument, & ENO: & Execution result \\
& d: & Bit device number or bit-specified word device to be turned ON :Bit
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{\(\mathrm{R}, \mathrm{ZR}\)} & \multicolumn{2}{|c|}{J:..1:} & \multirow[b]{2}{*}{U:} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant} & \multirow[t]{2}{*}{Others DY} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (d) & \(\bigcirc\) & \multicolumn{2}{|l|}{(Other than T or C )} & \multicolumn{3}{|c|}{\(\bigcirc\)} & \multicolumn{2}{|c|}{-} & \(\bigcirc\) \\
\hline
\end{tabular}

\section*{XPOINT}

When BL, S, TR, BLIS, or BLITR is used, refer to SFC control instructions of the MELSEC-Q/L/QnA Programming Manual (SFC).

\section*{Function}
(1) When the execution command is turned ON, the status of the specified devices becomes as shown below:
\begin{tabular}{|l|l|}
\hline \multicolumn{1}{|c|}{ Device } & \multicolumn{1}{c|}{ Device status } \\
\hline Bit device & Turns coils and contacts ON \\
\hline Bit-specified word device & Sets the value of specified bit to 1 \\
\hline
\end{tabular}
(2) Devices turned ON remain ON even when the execution command is turned OFF.

Devices turned ON by the SET instruction can be turned OFF by the RST instruction.

(3) When the execution command is OFF, the status of devices does not change.

\section*{O Operation Error}

No operation error occurs in the execution of the SET instruction.

\section*{\(\triangle\) Program Example}
(1) In the following program, Y8B is set (ON) when X8 turns ON, and Y8B is reset (OFF) when X9 turns ON.
[Structured ladder/FBD]

[ST]
SET(X8,Y8B);
RST(X9,Y8B);
(2) In the following program, the value of D 0 b 5 (bit 5 ) is set to 1 when X 8 turns ON , and the bit value is set to 0 when X9 turns ON.
[Structured ladder/FBD]

[ST]
SET(X8,D0.5);
RST(X9,D0.5);

When using \(X\) as a device, use the device numbers that are not used for the actual input. If the same number is used for the actual input device and input \(X\), the data of the actual input will be written over with the input \(X\) specified in the SET instruction.

\subsection*{5.3.6 Resetting devices (excluding annunciators)}

\section*{RST}

\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \begin{tabular}{l}
Input argument, \\
Output argument,
\end{tabular} & \begin{tabular}{l}
EN: \\
ENO: \\
d:
\end{tabular} & \multicolumn{5}{|l|}{\begin{tabular}{l}
Executing condition \\
Execution result \\
Bit device number or bit-specified word device to be reset Word device number to be reset
\end{tabular}} & \multicolumn{4}{|l|}{\begin{tabular}{l}
:Bit \\
:Bit \\
:ANY_SIMPLE
\end{tabular}} \\
\hline & Setting data & \multicolumn{2}{|l|}{Internal device} & R, ZR & \multicolumn{2}{|r|}{小...)} &  & Zn & Constant & Others DY \\
\hline & (d) & \multicolumn{7}{|c|}{\(\bigcirc\)} & - & \(\bigcirc\) \\
\hline
\end{tabular}

\section*{XPOINT}

When BL, S, TR, BLIS, or BLITR is used, refer to SFC control instructions of the MELSEC-Q/L/QnA Programming Manual (SFC).

\section*{Function}
(1) When the execution command is turned ON, the status of the specified devices becomes as shown below.
\begin{tabular}{l|l}
\hline \multicolumn{1}{c|}{ Device } & \multicolumn{1}{c}{ Device status } \\
\hline Bit device & Turns coils and contacts OFF \\
\hline Timer and counter & Sets the current value to 0, and turns coils and contacts OFF \\
\hline Bit-specified word device & Sets the value of specified bit to 0 \\
\hline Word device other than timer and counter & Sets the value to 0 \\
\hline
\end{tabular}
(2) When the execution command is OFF, the status of devices does not change.
(3) The functions of the word devices specified by the RST instruction are identical to the following ladder.


\section*{O Operation Error}

No operation error occurs in the execution of the RST instruction.

\section*{\(\triangle\) Program Example}
(1) In the following program, the value of the data register is set to 0 .
[Structured ladder/FBD]

[ST]
MOVP(X0,K4X10,Var_D8);
RST(X5,Var_D8);
(2) In the following program, the 100 ms retentive timer and counter are reset. [Structured ladder/FBD]

```

[ST]
OUT_T(X4,STC225,18000);
IF STS225 THEN
OUT_C(TRUE,CC23,16);
RST(TRUE,STN225);
END_IF;
OUT(CS23,Y55);
RST(X5,CN23);

```

\section*{Remark}
1. When resetting timers and counter devices, only coils (TC, STC, CC) and current values (TN, STN, CN) can be specified for output argument © . Contacts (TS, STS, CS) cannot be specified.

\subsection*{5.3.7 Setting and resetting annunciators}

SET, RST

\section*{Basic}

High
performanc
Process
Redundant Universal

LCPU


\section*{Function}

\section*{SET}
(1) The annunciator specified for (d) is turned ON when the execution command is turned ON.
(2) The following responses occur when an annunciator (F) is turned ON.
- The "USER" LED*1 turns ON.
- The annunciator numbers which are ON (F numbers) are stored to special registers (SD64 to SD79).
- The value of SD63 is incremented by 1.
*1 : For Basic model QCPU, the "ERR" LED displays on the front of the CPU module.
(3) If the value of SD63 is 16 (which happens when 16 annunciators are already ON), even if a new annunciator is turned ON, its number will not be stored to SD64 to SD79.

\section*{RST}
(1) The annunciator specified for (d) is turned OFF when the execution command is turned ON.
(2) The annunciator numbers ( \(F\) numbers) of annunciators that have turned OFF are deleted from the special registers (SD64 to SD79), and the value of SD63 is decremented by 1.
(3) When the value of SD63 is "16", the annunciator numbers are deleted from SD64 to SD79 by the use of the RST instruction. If the annunciators whose numbers are not registered to SD64 to SD79 are ON, these numbers are registered.
If all annunciator numbers from SD64 to SD79 are turned OFF, the "USER" LED*2 is turned OFF.
*2 : For Basic model QCPU, the "ERR" LED displays on the front of the CPU module.

\section*{[Operations which take place when SD63 is 16]}


\section*{Remark}

For details of annunciators, refer to the User's Manual (Function Explanation, Program Fundamentals) of the CPU module to be used.

\section*{Operation Error}

No operation error occurs in the execution of the SET and RST instructions.

\section*{\(\square\) Program Example}

In the following program, the annunciator F11 is turned ON when X 1 turns ON , and the value 11 is stored to the special register (SD64 to SD79). Further, the annunciator F11 is reset when X2 turns ON, and the value 11 is deleted from the special registers (SD64 to SD79).
[Structured ladder/FBD]


\section*{[ST]}

SET(X1,F11);
RST(X2,F11);
[Operation]


\subsection*{5.3.8 Rising edge and falling edge outputs}


Input argument,
\begin{tabular}{ll} 
EN: & Executing condition \\
ENO: & Execution result \\
d: & Device to be set as pulse output
\end{tabular}
:Bit
Output argument,
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...ala} & \multirow[b]{2}{*}{U:} & \multirow[t]{2}{*}{Zn} & \multirow{2}{*}{Constant} & \multirow[t]{2}{*}{Others DY} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (d) & \multicolumn{6}{|c|}{\(\bigcirc\)} & \multicolumn{2}{|c|}{-} & \(\bigcirc\) \\
\hline
\end{tabular}

\section*{Function}

\section*{PLS}
(1) Turns ON the specified device when the execution command is turned from OFF to ON, and turns OFF the device in any other case the execution command is turned from OFF to ON (ON to ON, ON to OFF or OFF to OFF).

When there is one PLS instruction for the device specified for (d) during one scan, the specified device turns ON one scan.
Refer to Section 3.4 for the operation to be performed when the PLS instruction for the same device is executed more than once during one scan.

(2) If the switch is changed from RUN to STOP after the execution of the PLS instruction, the PLS instruction will not be executed again even if the switch is set back to RUN.

(3) When specifying a latch relay ( L ) for the execution command and turning the power supply from OFF to ON with the latch relay ON, the execution command turns from OFF to ON at the first scan, executing the PLS instruction and turning ON the specified device.
The device turned ON at the first scan after power-ON turns OFF at the next PLS instruction.

\section*{PLF}
(1) Turns ON the specified device when the execution command is turned from ON to OFF, and turns OFF the device in any other case the execution command is turned from ON to OFF (OFF to OFF, OFF to ON or ON to ON).

When there is one PLF instruction for the device specified for (d) during one scan, the specified device turns ON one scan.
Refer to Section 3.4 for the operation to be performed when the PLF instruction for the same device is executed more than once during one scan.

(2) If the switch is changed from RUN to STOP after the execution of the PLF instruction, the PLF instruction will not be executed again even if the switch is set back to RUN.

\section*{XPOINT}

Note that the device specified for (d) may remain ON for more than one scan if the PLS or PLF instruction is jumped by the CJ instruction or if the executed subroutine program was not called by the CALL instruction.

No operation error occurs in the execution of the PLS and PLF instructions.

\section*{\(\square\) Program Example}
(1) In the following program, the PLS instruction is executed when X9 turns ON. [Structured ladder/FBD]

[ST]
PLS(X9,M9);
[Timing chart]

(2) In the following program, the PLF instruction is executed when X9 turns OFF. [Structured ladder/FBD]

[ST]
PLF(X9,M9);
[Timing chart]


\subsection*{5.3.9 Bit device output inversion}

Input argument,
EN: Executing condition
:Bit
s : \(\quad\) Device number whose status to be inverted
Output argument, ENO: Execution result
Bit
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..nlan} & \multirow[b]{2}{*}{U:..icam} & \multirow{2}{*}{Zn} & \multirow{2}{*}{Constant} & \multirow[t]{2}{*}{Others DY} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & \multicolumn{6}{|c|}{\(\bigcirc\)} & \multicolumn{2}{|c|}{-} & \(\bigcirc\) \\
\hline
\end{tabular}

\section*{Function}
(1) Inverts the output status of the device specified for (s) when the execution command is turned from OFF to ON.
\begin{tabular}{l|c|c|}
\hline \multirow{2}{*}{ Device } & \multicolumn{2}{|c}{ Device status } \\
\cline { 2 - 3 } & Prior to execution of the FF instruction & After execution of the FF instruction \\
\hline \multirow{2}{*}{ Bit device } & OFF & ON \\
\cline { 2 - 3 } & ON & OFF \\
\hline \multirow{2}{*}{ Bit-specified word device } & 0 & 1 \\
\cline { 2 - 3 } & 1 & 0 \\
\hline
\end{tabular}

\section*{0 Operation Error}

No operation error occurs in the execution of the FF instruction.

\section*{\(\square\) Program Example}
(1) In the following program, the output of Y 10 is inverted when X 9 turns ON .
[Structured ladder/FBD]

[ST]
FF(X9,Y10);
[Timing chart]

(2) In the following program, b10 (bit 10) of D10 is inverted when X0 turns ON.
[Structured ladder/FBD]

[ST]
FF(X0,D10.A);
[Timing chart]


\subsection*{5.3.10 Pulse conversion of direct output}

\begin{tabular}{llll} 
Input argument, & EN: & Executing condition & \(:\) Bit \\
Output argument, & ENO: & Execution result & \(:\) Bit \\
& d: & Bit to be set as pulse output & :Bit
\end{tabular}


\section*{Function}
(1) Performs pulse output of direct access output (DY) specified for (d).

If DYO of the DELTA instruction has been specified, the resulting operation will be identical to the ladder shown below, which uses the SET and RST instructions.

[Operation]

(2) The DELTA(P) instruction is used by commands for rising edge execution for an intelligent function module.

\section*{OO Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- A direct access output number specified for (d) has exceeded the CPU module output range.
(Error code: 4101)
\(\triangle\) Program Example
In the following program, CH 1 of the QD62 module mounted at slot 0 of the main base unit is preset, when X20 turns ON.
[Structured ladder/FBD]

[ST]
DMOVP(X20,K0,U0\G0);
DELTAP(X20,DY1);

\subsection*{5.4 Shift Instructions}

\subsection*{5.4.1 Bit device shift}
\(P\) : Executing condition
,


Input argument, Output argument,
\begin{tabular}{ll} 
EN: & Executing condition \\
ENO: & Execution result
\end{tabular}

\section*{Bit}

ExO: Execution result
d: Device number to be shifted
:Bit


\section*{Function}
(1) When a bit device is used
(a) Shifts to a device specified for (d) the ON/OFF status of the device immediately prior to the one specified for (d), and turns the prior device OFF.
For example, if M11 has been specified in the SFT instruction, when the SFT instruction is executed, it will shift the ON/OFF status of M10 to M11, and turns M10 OFF.
(b) Turn the first device to be shifted ON with the SET instruction.
(c) When the SFT and SFTP instructions are to be used consecutively, the program starts from the device with the larger number.

(2) When a bit-specified word device is used
(a) Shifts to a bit in the device specified for (d) the \(1 / 0\) status of the bit immediately prior to the one specified for (d), and turns the prior bit to 0.
For example, if 0.5 (bit 5 [b5] of D0) has been specified in the SFT instruction, when the SFT instruction is executed, it will shift the \(1 / 0\) status of b4 of D0 to b5, and turns b4 to 0 .

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device specified for (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, the devices from Y57 to Y5B are shifted when X8 turns ON.
[Structured ladder/FBD]

[ST]
IF LDP(TRUE,X8) THEN
SFT(TRUE,Y5B);
SFT(TRUE,Y5A);
SFT(TRUE,Y59);
SFT(TRUE,Y58);
END_IF;
IF LDP(TRUE,Y7) THEN
SET(TRUE,Y57);
END_IF;
[Timing chart]


When using the SFT instruction, do not branch a line from (d).
If branched, the operation result is not output to the devices or labels of branch destination correctly.

For details of branch point of line from destination, refer to Section 3.2.

\subsection*{5.5 Master Control Instructions}

\subsection*{5.5.1 Setting and resetting master control}

Process
Redundan
Universal
LCPU

MC
MCR


Input argument,

:Bit
\(\mathrm{n} \quad\) Nesting (N0 to N14)
Output argument,
ENO: Execution result
d : Device number to be turned ON (When using the MC
:ANY16
:Bit
instruction)
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J等:} & \multirow[b]{2}{*}{U...igat} & \multirow{2}{*}{Zn} & \multirow{2}{*}{Constant} & \multicolumn{2}{|l|}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & N & DY \\
\hline n & \multicolumn{6}{|c|}{-} & & & \(\bigcirc\) & - \\
\hline (d) & \multicolumn{6}{|c|}{\(\bigcirc\)} & & & - & \(\bigcirc\) \\
\hline
\end{tabular}

\section*{IT Function}

The master control instruction is used to enable the creation of highly efficient ladder switching sequence programs, through the opening and closing of a common bus for ladders.

A ladder using the master control is as follows.


\section*{MC}
(1) If the execution command of the MC instruction is ON when master control is started, the result of the operation from the MC instruction to the MCR instruction will be exactly as the instruction (ladder) shows.
If the execution command of the MC instruction is OFF, the result of the operation from the MC instruction to the MCR instruction will be as shown below.
\begin{tabular}{l|l}
\hline \multicolumn{1}{c|}{ Device } & \multicolumn{1}{c}{ Device Status } \\
\hline \begin{tabular}{l} 
High-speed timer \\
Low-speed timer
\end{tabular} & Count value turns to 0, coils and contacts all turn OFF. \\
\hline \begin{tabular}{l} 
High-speed retentive timer \\
Low-speed retentive timer \\
Counter
\end{tabular} & \begin{tabular}{l} 
Coils turn OFF, but counter values and contacts all retain \\
current status.
\end{tabular} \\
\hline Devices in the OUT instruction & All turned OFF \\
\hline \begin{tabular}{l} 
Devices in the SET, RST, SFT, basic, and application \\
instructions
\end{tabular} & Retain current status \\
\hline
\end{tabular}
(2) Even when the MC instruction is OFF, instructions from the MC instruction to the MCR instruction will be executed, so scan time will not be shortened.

\section*{XPOINT}

When a ladder with master control contains instructions that do not require any contact instruction (such as the FOR to NEXT instruction loop, EI and DI instructions), the CPU module executes these instructions regardless of the ON/ OFF status of the MC instruction execution command.
(3) By changing the device specified for © \({ }^{(d)}\), the MC instruction can use the same nesting ( N ) number as often as desired.
(4) Coils from devices specified for © are turned ON when the MC instruction is ON. Further, using these same devices with the OUT instruction or other instructions will cause them to become duplicated coils, so devices specified for © should not be used within other instructions.

\section*{MCR}
(1) This is the instruction for recovery from the master control, and indicates the end of the master control range of operation.
(2) Do not place contacts before the MCR instruction.
(3) Use the MC instruction and MCR instruction of the same nesting number as a set. However, when the MCR instructions are nested in one place, all master controls can be terminated with the lowest nesting number.
(Refer to the "Precautions for nesting" in the program example.)

\section*{O Operation Error}

No operation error occurs in the execution of the MC and MCR instructions.

\section*{\(\square\) Program Example}

The master control instruction can be used in nesting. The different master control regions are distinguished by nesting number. Nesting can be performed from N0 to N14.

The use of nesting enables the creation of ladders which successively limit the executing condition of the program.

A ladder using nesting would appear as shown below.
[Structured ladder/FBD]


\section*{Precautions when using nesting architecture}
(1) Nesting can be used up to 15 times (N0 to N14)

When using nesting, nests should be inserted from the lower to higher nesting number with the MC instruction, and from the higher to the lower order with the MCR instruction.
If this order is reversed, there will be no nesting architecture, and the CPU module will not be capable of performing correct operations. For example, if nesting is specified in the order N1 to N0 by the MC instruction, and also specified in the N1 to N0 order by the MCR instruction, the vertical bus will intersect and a correct master control ladder will not be produced.
(2) If the nesting architecture results in MCR instructions concentrated in one location, all master controls can be terminated by use of just the lowest nesting number.


\section*{Precautions when using MC/MCR instruction in structured ladder/FBD/ST}
(1) \(M C / M C R\) instruction is not required when using structured ladder/FBD/ST.

Although, if using MC/MCR instruction, nesting number should be used in the ascending order from 0 regardless of whether MC/MCR instruction is used or not in nesting.
(2) The released nesting number and the following number using a MCR instruction can be used in a next MC instruction.

When using the MC instruction, do not branch a line from (d).
If branched, the operation result is not output to the devices or labels of branch destination correctly.

For details of branch point of line from destination, refer to Section 3.2.

\subsection*{5.6 End Instructions}

\subsection*{5.6.1 Ending main routine program}

\section*{FEND}



Input argument,
EN: Executing condition
:Bit
Output argument,
ENO: Execution result
:Bit
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..al} & \multirow[b]{2}{*}{U:...iga} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline - & \multicolumn{9}{|c|}{-} \\
\hline
\end{tabular}

\section*{Function}
(1) The FEND instruction is used in cases where the CJ instruction or other instructions are used to cause a branch in the sequence program operations.
(2) Execution of the FEND instruction will cause the CPU module to terminate the program it was executing.
(3) Even sequence programs following the FEND instruction can be displayed in ladder display using a programming tool.

(4) When the multiple tasks or multiple POUs are registered in program files, tasks or POU which are set after POUs with FEND instruction cannot be executed without the CALL instruction or CJ instruction.


\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The FEND instruction is executed after the execution of the CALL instruction, and before the execution of the RET instruction.
(Error code: 4211)
- The FEND instruction is executed after the execution of the FOR instruction, and before the execution of the NEXT instruction.
(Error code: 4200)
- The FEND instruction is executed within the CHKCIR and CHKEND instruction loop.
(Error code: 4230)
- The FEND instruction is executed within the IX and IXEND instruction loop.
(Error code: 4231)

\subsection*{5.7 Other Instructions}

\subsection*{5.7.1 Sequence program stop}

STOP



Input argument,
EN: Executing condition
:Bit
Output argument,
ENO: Execution result
:Bit
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{小等)} & \multirow[b]{2}{*}{U:...icai..} & \multirow{2}{*}{Zn} & \multirow{2}{*}{Constant} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline - & \multicolumn{9}{|c|}{-} \\
\hline
\end{tabular}

\section*{Function}
(1) Resets the output ( Y ) and stops the CPU module operation when the execution command is turned ON.
(The same result will take place if the switch is turned to the STOP setting.)
(2) Execution of the STOP instruction will cause the value of b4 to b7 of the special register SD203 to become 3.

(3) In order to restart CPU module operations after the execution of the STOP instruction, return the switch from RUN to STOP, and back to the RUN position.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The STOP instruction was executed before the execution of the RET instruction and after the execution of the CALL instruction.
(Error code: 4211)
- The STOP instruction was executed before the execution of the NEXT instruction and after the execution of the FOR instruction.
(Error code: 4200)
- The STOP instruction was executed during an interrupt program prior to the execution of the IRET instruction.
(Error code: 4221)
- The STOP instruction was executed within the CHKCIR to CHKEND instruction loop.
(Error code: 4230)
- The STOP instruction was executed within the IX to IXEND instruction loop.
(Error code: 4231)
- The STOP instruction was executed in the fixed scan execution program.
(For Universal model QCPU and LCPU)
(Error code: 4223)

\section*{\(\triangle\) Program Example}

In the following program, the CPU module is stopped when X8 turns ON.
[Structured ladder/FBD]
\begin{tabular}{l|llll} 
\\
2
\end{tabular}

\section*{[ST]}

STOP(X8);
OUT(X0A,Y13);
OUT(XOB,Y23);

\section*{MEMO}
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)

\section*{BASIC INSTRUCTIONS}
6.1 Comparison Operation Instructions ..... 6-2
6.2 Arithmetic Operation Instructions ..... 6-29
6.3 Data Conversion Instructions ..... 6-74
6.4 Data Transfer Instructions ..... 6-116
6.5 Program Branch Instructions ..... 6-149
6.6 Program Execution Control Instructions ..... 6-156
6.7 I/O Refresh Instructions ..... 6-168
6.8 Other Convenient Instructions ..... 6-170

\subsection*{6.1 Comparison Operation Instructions}

\subsection*{6.1.1 16-bit BIN data comparison}
\(\square=, \square<>, \square<=, \square<, \square>=, \square>\)
Basic
Process
Redundar
Universal
LCPU


\section*{Function}
(1) Treats 16-bit BIN data from device specified for (51) and 16-bit BIN data from device specified for \(\Omega_{2}\) as a normally open contact, and performs comparison operation.
(2) The results of the comparison operations for each instruction are as follows.
\begin{tabular}{c|c|c|c|c|c|}
\hline \multicolumn{2}{c|}{\begin{tabular}{c} 
Instruction \\
symbol in
\end{tabular}} & Condition & Comparison result & \begin{tabular}{c} 
Instruction \\
symbol in
\end{tabular} & Condition
\end{tabular} Comparison result
(3) When assigning hexadecimal constants to ©s1) and (s2), and if the numeric value (8 to F) whose most significant bit (b15) is 1 is specified as a constant, the value is considered as a negative BIN value in comparison operation.
(4) When assigning "word [unsigned]/ bit string [16 bits]" type label to ©1) and ©2), the type is considered as word [signed] type in comparison operation. For comparison operation as "word [unsigned]/ bit string [16 bits]" type, use application functions EQ_E, NE_E, LE_E, LT_E, GE_E, GT_E.
For details of application functions, refer to MELSEC-Q/L Structured Programming Manual (Application Functions).
\begin{tabular}{|c|c|c|}
\hline Instruction symbol in \(\square\) & Condition & Corresponding application function \\
\hline \(\square=\) & (51) \(=\) (s2) & EQ_E \\
\hline \(\square<>\) & (s1) \(\neq\) (s2) & NE_E \\
\hline \(\square<=\) & (51) \(\leqq\) ( 22 & LE_E \\
\hline \(\square<\) & (s1) < (s2) & LT_E \\
\hline \(\square>=\) & (51) \(\geqq\) (s2) & GE_E \\
\hline \(\square>\) & (s1) \(>\) (s2) & GT_E \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline Program example & Input value & Returned value \\
\hline  & \[
\begin{aligned}
& \text { wordVar1 }:=1 \\
& \text { wordVar2 }:=32768
\end{aligned}
\] & \begin{tabular}{l}
OFF \\
Since 32768 of wordVar2 is considered as word [signed] type, executes comparison operation with -32768 , and the output becomes OFF.
\end{tabular} \\
\hline  & \[
\begin{aligned}
& \text { wordVar1 }:=1 \\
& \text { wordVar2 }:=32768
\end{aligned}
\] & \begin{tabular}{l}
ON \\
Using an application function executes comparison operation as "word [unsigned]/ bit string [16 bits]" type, and the output becomes ON.
\end{tabular} \\
\hline
\end{tabular}
(5) The \(\mathrm{OR}=, \mathrm{OR}<>, \mathrm{OR}<=, \mathrm{OR}<, \mathrm{OR}>=\), or \(\mathrm{OR}>\) instruction performs comparison operation of operation result between "(s1), (s2)" and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400.
When using the \(O R=, O R<>, O R<=, O R<, O R>=\), or \(O R>\) instruction, connect \(E N\) and ENO in series as shown below.


\section*{Operation Error}

No operation error occurs in the execution of the \(=,<>,<=,<,>=\), and \(>\) instructions.

\section*{Program Example}
(1) In the following program, the value in Var_D0 is compared with the value in Var_D3, and Y33 turns ON when the values are identical.
[Structured ladder/FBD]

(2) In the following program, the BIN value of 100 is compared with the value in Var_D3 when M3 is ON, and Y33 turns ON when the value in Var_D3 is other than 100.
[Structured ladder/FBD]

(3) In the following program, the BIN value of 100 is compared with the value in Var_D3 when M3 turns ON, and Y33 turns ON when the value in Var_D3 is less than 100 or when M8 is ON.
[Structured ladder/FBD]

(4) In the following program, the value in Var_D0 is compared with the value in Var_D3, and Y33 turns ON when the value in Var_D3 is equal or greater than the value in Var_D00 or when both M3 and M8 are ON.
[Structured ladder/FBD]

6.1.2 32-bit BIN data comparison
\(\square \mathrm{D}=, \square \mathrm{D}<>, \square \mathrm{D}<=, \square \mathrm{D}<, \square \mathrm{D}>=, \square \mathrm{D}>\)
\begin{tabular}{|c|c|c|c|c|}
\hline LD & ( \(\mathrm{D}=\) ) & ( \(\square \mathrm{D}=\) & = & \()\) \\
\hline AND & ( \(\mathrm{D}<>\) ) & \(\square \mathrm{D}<>\) & \# & \\
\hline OR & ( \(\mathrm{D}<=\) ) & \(\square \mathrm{D}<=\) & \(\leqq\) & \\
\hline & (D<) & \(\square \mathrm{D}<\) & < & \\
\hline & (D>=) & \(\square \mathrm{D}>=\) & : \(\geqq\) & \\
\hline & (D>) & - \(\square\) D> & > & \\
\hline
\end{tabular}

\begin{tabular}{|c|c|c|}
\hline & \multicolumn{2}{|l|}{indicates any of the following} \\
\hline instructions. & & \\
\hline LDD= & ANDD= & ORD= \\
\hline LDD<> & ANDD<> & ORD<> \\
\hline LDD<= & ANDD<= & ORD<= \\
\hline LDD< & ANDD< & ORD< \\
\hline LDD>= & ANDD>= & ORD>= \\
\hline LDD> & ANDD> & ORD> \\
\hline
\end{tabular}
\begin{tabular}{llll} 
Input argument, & EN: & Executing condition & \(:\) Bit \\
& \(\mathrm{s} 1, \mathrm{~s} 2:\) & \begin{tabular}{l} 
Comparison data, or start number of the device that stores \\
comparison data
\end{tabular} & ANY32 \\
Output argument, & ENO: & Execution result & \(:\) Bit
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...} & \multirow[t]{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & \multicolumn{8}{|c|}{\(\bigcirc\)} & - \\
\hline (s2) & \multicolumn{8}{|c|}{\(\bigcirc\)} & - \\
\hline
\end{tabular}

\section*{\(\sqrt{3}\) Function}
(1) Treats 32-bit BIN data from device specified for \$11 and 32-bit BIN data from device specified for \(\S_{2}\) as a normally open contact, and performs comparison operation.
(2) The results of the comparison operations for each instruction are as follows.
\begin{tabular}{|c|c|c|c|c|c|}
\hline Instruction symbol in \(\square\) & Condition & Comparison result & Instruction symbol in \(\square\) & Condition & Comparison result \\
\hline \(\square \mathrm{D}=\) & (s1) \(=\) s \({ }^{\text {a }}\) & \multirow{6}{*}{Conduction state} & \(\square \mathrm{D}=\) & (s1) \(\neq\) (s2) & \multirow{6}{*}{Non-conduction state} \\
\hline \(\square \mathrm{D}<>\) & (s1) \(\neq\) (s2) & & \(\square \mathrm{D}<>\) & (51) \(=\) (s2) & \\
\hline \(\square \mathrm{D}<=\) & (s1) \(\leqq\) ( 22 & & \(\square \mathrm{D}<=\) & (51) \(>\) (52) & \\
\hline \(\square \mathrm{D}<\) & (s1) \(<\) (22) & & \(\square \mathrm{D}<\) & (s1) \(\geqq\) (s2) & \\
\hline \(\square \mathrm{D}>=\) & (s1) \(\geqq\) (s2) & & \(\square \mathrm{D}>=\) & (51) < (s2) & \\
\hline \(\square\) D> & (51) \(>\) (s2) & & \(\square\) D> & (51) \(\leqq\) ( 52 & \\
\hline
\end{tabular}
(3) When assigning hexadecimal constants to ©1 and ©2, and the numeric value (8 to F) whose most significant bit (b31) is 1 is specified as a constant, the value is considered as a negative BIN value in comparison operation.
(4) Data used for comparison should be specified by a 32-bit instruction (such as DMOV instruction).
If specification is made with a 16-bit instruction (such as MOV instruction), comparisons of large and small values cannot be performed correctly.
(5) When assigning "double word [unsigned]/ bit string [32 bits]" type label to ©11 and ©2), the type is considered as double word [signed] type in comparison operation. For comparison operation as "double word [unsigned]/ bit string [32 bits]" type, use application functions EQ_E, NE_E, LE_E, LT_E, GE_E, GT_E.
For details of application functions, refer to MELSEC-Q/L Structured Programming Manual Application Functions).
\begin{tabular}{|c|c|c|}
\hline \begin{tabular}{l}
Instruction \\
symbol in \(\square\)
\end{tabular} & Condition & Corresponding application function \\
\hline \(\square \mathrm{D}=\) & (51) \(=\) ( 22 & EQ_E \\
\hline \(\square \mathrm{D}<>\) & (1) \(\neq\) (3) & NE_E \\
\hline \(\square \mathrm{D}<=\) & (11) \(\leqq\) (2) & LE_E \\
\hline \(\square \mathrm{D}<\) & (51) < (2) & LT_E \\
\hline \(\square \mathrm{D}>=\) & (11) \(\geqq\) (2) & GE_E \\
\hline \(\square \mathrm{D}>\) & (51) \(>\) (2) & GT_E \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline Program example & Input value & Returned value \\
\hline  & \[
\begin{aligned}
& \text { dwordVar1 }:=1 \\
& \text { dwordVar2 := } 2147483648
\end{aligned}
\] & \begin{tabular}{l}
OFF \\
Since 2147483648 of dwordVar2 is considered as double word [signed] type, executes comparison operation with -2147483648, and the output becomes OFF.
\end{tabular} \\
\hline \[
\begin{aligned}
& \text { TRUE-EN }{ }^{\text {LE EE }} \text { ENO } \\
& \text { dwordVar1 }-\mathbb{N} \\
& \text { dwordVar2- } \mathbb{N}
\end{aligned}
\] & \[
\begin{aligned}
& \text { dwordVar1 }:=1 \\
& \text { dwordVar2 := } 2147483648
\end{aligned}
\] & \begin{tabular}{l}
ON \\
Using an application function executes comparison operation as "double word [unsigned]/ bit string [32 bits]" type, and the output becomes ON.
\end{tabular} \\
\hline
\end{tabular}
(6) The \(\mathrm{ORD}=, \mathrm{ORD}<>, \mathrm{ORD}<=, \mathrm{ORD}<, \mathrm{ORD}>=\), or \(\mathrm{ORD}>\) instruction performs comparison operation of operation result between "(s1), (s2)" and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400.
When using the ORD=, ORD<>, ORD<=, ORD<, ORD>=, or ORD> instruction, connect EN and ENO in series as shown below.


\section*{Operation Error}

No operation error occurs in the execution of the \(D=, D<>, D<=, D<, D>=\), and \(D>\) instructions.

\section*{\(\square\) Program Example}
(1) In the following program, the value in Var_D0 is compared with the value in Var_D3, and Y33 turns ON, when the value in Var_D0 and the value in Var_D3 are matched.
[Structured ladder/FBD]

(2) In the following program, the BIN value of 38000 is compared with the value in Var_D3 when M3 is ON, and Y33 turns ON when the value in Var_D3 is other than 38000.
[Structured ladder/FBD]

(3) In the following program, the BIN value of -80000 is compared with the value in Var_D3 when M3 is ON, and Y33 turns ON when the value in Var_D3 is less than -80000, or when M8 is ON.
[Structured ladder/FBD]

(4) In the following program, the value in Var_D0 is compared with the value in Var_D3, and Y33 turns ON when the value in Var_D3 is equal or greater than the value in Var_D0 or when both M3 and M8 are ON.
[Structured ladder/FBD]


\subsection*{6.1.3 Floating-point data comparison (single precision)}

- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later
\begin{tabular}{|c|c|c|c|c|}
\hline LD & (E=) & ( \(\square \mathrm{E}=\) & : = & ) \\
\hline AND & (E<>) & \(\square \mathrm{E}<>\) & & \\
\hline OR & (E<=) & \(\square \mathrm{E}<=\) & \(: \leqq\) & \\
\hline & (E<) & \(\square \mathrm{E}<\) & : < & \\
\hline & (E>=) & \(\square E>=\) & & \\
\hline & (E>) & ( \(\square \mathrm{E}>\) & : > & ) \\
\hline
\end{tabular}

\begin{tabular}{|c|c|c|}
\hline & \multicolumn{2}{|l|}{indicates any of the following} \\
\hline instructions. & & \\
\hline LDE= & ANDE= & ORE= \\
\hline LDE<> & ANDE<> & ORE<> \\
\hline LDE<= & ANDE<= & ORE<= \\
\hline LDE< & ANDE< & ORE< \\
\hline LDE>= & ANDE>= & ORE>= \\
\hline LDE> & ANDE> & ORE> \\
\hline
\end{tabular}
\begin{tabular}{lll} 
Input argument, & EN: & Executing condition \\
s1, s2: & \begin{tabular}{l} 
Comparison data, or start number of the device that stores \\
comparison data
\end{tabular} \\
Output argument, & ENO: & \begin{tabular}{l} 
Execution result
\end{tabular}
\end{tabular}
: Bit
Single-precision real
: Bit
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J....} & \multirow[b]{2}{*}{U\#1....} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & - & & & - & & - & - & \(\bigcirc\) & - \\
\hline (2) & - & & & - & & O & - & \(\bigcirc\) & - \\
\hline
\end{tabular}

\section*{Function}
(1) Treats the floating-point data from device specified for (s1) and the floating-point data from device specified for \(\circledR^{2}\) as a normally open contact, and performs comparison operation.
(2) The results of the comparison operations for each instruction are as follows.
\begin{tabular}{c|c|c|c|c|c|}
\hline \multicolumn{2}{c|}{\begin{tabular}{c} 
Instruction \\
symbol in \(\square\)
\end{tabular}} & Condition & Comparison result & \begin{tabular}{c} 
Instruction \\
symbol in
\end{tabular} & Condition
\end{tabular} Comparison result
(3) The ORE=, ORE<>, ORE<=, ORE<, ORE>=, or ORE> instruction performs comparison operation of operation result between "(51), ②" and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400.
When using the ORE=, ORE<>, ORE<=, ORE<, ORE>=, or ORE> instruction, connect EN and ENO in series as shown below.


\section*{®POINT}

Note that when using the \(\square \mathrm{E}=\) instruction, the values of operation result may not be equal depending on the error.

(4) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the specified device is -0 . *1 (For Basic model QCPU and High Performance model QCPU only) (Error code: 4100)
*1: There are CPU modules that do not result in any operation error when -0 is specified. For details, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
- The value of the specified device is outside the following range.
(For Universal model QCPU and LCPU)
\(0,2^{-126} \leqq \mid\) value of the specified device \(\mid<2^{128}\) (Error code: 4140)
- The value of the specified device is -0 , unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)

\section*{\(\triangle\) Program Example}
(1) In the following program, the value in Var_D0 is compared with the value in Var_D3 and Y33 turns ON when the value in Var_D0 and the value in Var_D3 are matched.
[Structured ladder/FBD]

(2) In the following program, the floating-point real number 1.23 is compared with the value in Var_D3 when M3 is ON, and Y33 turns ON when the value in Var_D3 is other than 1.23. [Structured ladder/FBD]

(4) In the following program, the value in Var_D0 is compared with the floating-point real number 1.23, and Y33 turns ON when 1.23 is equal to or greater than the value in Var_D0 or when both M3 and M8 are ON.
[Structured ladder/FBD]

6.1.4 Floating-point data comparison (double precision) \(\square \mathrm{ED}=, \square \mathrm{ED}<>, \square \mathrm{ED}<=, \square \mathrm{ED}<, \square \mathrm{ED}>=, \square \mathrm{ED}>\)

\begin{tabular}{|c|c|c|c|c|}
\hline LD & (ED=) & ( \(\square\) ED \(=\) & = & ) \\
\hline AND & (ED<>) & ■ED<> & \# & \\
\hline OR & (ED<=) & \(\square E D<=\) & \(\leqq\) & \\
\hline & (ED<) & \(\square E D<\) & < & \\
\hline & (ED>=) & -ED>= & \(\geqq\) & \\
\hline & (ED>) & - \(\square\) ED> & > & ) \\
\hline
\end{tabular}

\begin{tabular}{|lll|}
\hline instructions. & & \\
LDED \(=\) & ANDED \(=\) & ORED \(=\) \\
LDED<> & ANDED<> & ORED<> \\
LDED<= & ANDED<= & ORED<= \\
LDED< & ANDED< \(<\) & ORED< \(<\) \\
LDED>= & ANDED>= & ORED>= \\
LDED> & ANDED \(>\) & ORED> \(>\) \\
\hline
\end{tabular}
\begin{tabular}{lll} 
Input argument, & EN: & Executing condition \\
s1, s2: & \begin{tabular}{l} 
Comparison data, or start number of the device that stores \\
comparison data
\end{tabular} \\
Output argument, & ENO: & \begin{tabular}{l} 
Execution result
\end{tabular}
\end{tabular}

\footnotetext{
: Bit
Double-precision real
: Bit
}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..flat} & \multirow[b]{2}{*}{U} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{\begin{tabular}{l}
Constant \\
E
\end{tabular}} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s1) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (s2) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline
\end{tabular}

\section*{Function}
(1) Treats the 64-bit floating-point real number specified for \((11\) and the 64-bit floating-point real number specified for \(\varsigma_{2}\) ) as a normally open contact, and performs comparison operation.
(2) The results of the comparison operations for each instruction are as follows.
\begin{tabular}{|c|c|c|c|c|c|}
\hline Instruction symbol in \(\square\) & Condition & Comparison result & Instruction symbol in \(\square\) & Condition & Comparison result \\
\hline \(\square E D=\) & (51) \(=\) s2 & \multirow{6}{*}{Conduction state} & \(\square E D=\) & (51) \(\neq\) (s2) & \multirow{6}{*}{Non-conduction state} \\
\hline \(\square \mathrm{ED}<>\) & (s1) \(\neq\) (s2) & & \(\square E D<>\) & (51) \(=\) (s2) & \\
\hline \(\square \mathrm{ED}<=\) & (s1) \(\leqq\) (s2) & & \(\square \mathrm{ED}<=\) & (51) \(>\) (s2) & \\
\hline \(\square E D<\) & (51) < (s2) & & \(\square E D<\) & (s1) \(\geqq\) (s2) & \\
\hline \(\square E D>=\) & (s1) \(\geqq\) (s2) & & \(\square E D>=\) & (51) \(<\) (s2) & \\
\hline \(\square E D>\) & (51) \(>\) (s2) & & \(\square E D>\) & (51) \(\leqq\) (52) & \\
\hline
\end{tabular}
(3) The ORED=, ORED<>, ORED<=, ORED<, ORED>=, or ORED> instruction performs comparison operation of operation result between "(51), ©2" and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400.
When using the ORED=, ORED<>, ORED<=, ORED<, ORDE>=, or ORED> instruction, connect EN and ENO in series as shown below.

(4) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The specified value is not within the following range.
(Error code: 4140)
\(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 .
(Error code: 4140)

\section*{Program Example}
(1) In the following program, the value in Var_D0 is compared with the value in Var_D4, and Y33 turns ON when the value in Var_D0 and the value in Var_D4 are matched.
[Structured ladder/FBD]

(2) In the following program, the floating-point real number 1.23 is compared with the value in Var_D4 when M3 is ON, and Y33 turns ON when the value in Var_D4 is other than 1.23.
[Structured ladder/FBD]

(3) In the following program, the value in Var_D00 is compared with the value in Var_D4, and Y33 turns ON when the value in Var_D4 is smaller than the value in Var_D0 or when M8 is ON.
[Structured ladder/FBD]

(4) In the following program, the value in Var_D0 is compared with the floating-point real number 1.23, and Y33 turns ON when 1.23 is equal to or greater than the value in Var_D0 or when both M3 and M8 are ON.
[Structured ladder/FBD]


\section*{Caution}

Since the maximum number of digits of real number that can be input by the programming tool is 15 digits, the comparison with the real number whose number of significant figures is 16 or more cannot be made by the instruction shown in this section.

When judging match/mismatch with the real number whose significant figures is 16 or more by the instruction in this section, compare it with the approximate values of the real number to be compared and judge by the sizes.

Example When judging the match of E1.234567890123456+10 (Number of significant figures is 16) and the double-precision floating-point data.


Check if values of D0 to D3 are within this range. (Boundary values are not included.)

Example Judging the mismatch of E1.234567890123456+10 (Number of significant figures is 16) and the double-precision floating-point data.

(Boundary values are included.)
6.1.5 Character string data comparison

\begin{tabular}{|c|c|c|c|c|}
\hline LD & (\$=) & \(\square \square^{\square}=\) & = & \\
\hline AND & (\$<>) & \(\square \square^{\text {a }}\) < & * & \\
\hline \multirow[t]{4}{*}{OR} & (\$<=) & \(\square\) ¢ \(<=\) & \(\leqq\) & \\
\hline & (\$<) & \(\square \square^{\text {c }}\) & < & \\
\hline & (\$>=) & \(\square \$>=\) & \(\geqq\) & \\
\hline & (\$>) & \(\square \$>\) & > & \\
\hline
\end{tabular}

\begin{tabular}{llll} 
Input argument, & EN: & Executing condition & \(:\) Bit \\
s1, s2: & \begin{tabular}{l} 
Comparison data, or start number of the device that stores \\
comparison data
\end{tabular} & \(:\) String
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...al} & \multirow[b]{2}{*}{U} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant
\$} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s1) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (s2) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline
\end{tabular}

\section*{Function}
(1) Treats the character string data specified for (51) and the character string data specified for (s2) as a normally open contact, and performs comparison operation.
(2) A comparison operation involves the character-by-character comparison of the ASCII code from the first character in the character string.
(3) The character string data of (51) and (52) for the comparison refers to the data up to 00 H .
(a) If all character string data are matched, the comparison result will be matched.
(51)
\begin{tabular}{|cc:cc|}
\hline 42 H & \((\mathrm{B})\) & \(41_{\mathrm{H}}\) & \((\mathrm{A})\) \\
\hline \(44_{\mathrm{H}}\) & \((\mathrm{D})\) & \(43_{\mathrm{H}}\) & \((\mathrm{C})\) \\
\hline \(00_{\mathrm{H}}\) & 45 H & \((\mathrm{E})\) \\
\hline \multicolumn{3}{c}{ "ABCDE" }
\end{tabular}
(s2)
\begin{tabular}{|ll:ll|}
\hline 42 H & \((\mathrm{B})\) & \(41_{\mathrm{H}}\) & \((\mathrm{A})\) \\
\hline \(44_{\mathrm{H}}\) & \((\mathrm{D})\) & \(43_{\mathrm{H}}\) & \((\mathrm{C})\) \\
\hline \(00_{\mathrm{H}}\) & 45 H & (E) \\
\hline \multicolumn{4}{c}{ "ABCDE" }
\end{tabular}
\begin{tabular}{c|c|c|c}
\hline Instruction symbol in & Comparison result & Instruction symbol in & Comparison result \\
\hline\(\square \$=\) & Conduction state & \(\square \$<\) & Non-conduction state \\
\hline\(\square \$<>\) & Non-conduction state & \(\square \$>=\) & Conduction state \\
\hline\(\square \$<=\) & Conduction state & \(\square \$>\) & Non-conduction state \\
\hline
\end{tabular}
(b) If the character string data are different, the character string with the larger character code will be larger.
(s1)
\begin{tabular}{|ll:ll|}
\hline 42 H & (B) & 41 H & \((\mathrm{A})\) \\
\hline \(44_{\mathrm{H}}\) & (D) & 43 H & (C) \\
\hline \(0 \mathrm{H}_{\mathrm{H}}\) & & 46 H & (F) \\
\hline \multicolumn{3}{c|}{ "ABCDF" }
\end{tabular}
\(\square\)
(s2)
\begin{tabular}{|ll:ll|}
\hline 42 H & \((\mathrm{B})\) & 41 H & \((\mathrm{A})\) \\
\hline \(44_{\mathrm{H}}\) & \((\mathrm{D})\) & 43 H & \((\mathrm{C})\) \\
\hline \(00_{\mathrm{H}}\) & & 45 H & \((\mathrm{E})\) \\
\hline \multicolumn{4}{c|}{ "ABCDE" }
\end{tabular}
\begin{tabular}{c|c|c|c}
\hline Instruction symbol in & Comparison result & Instruction symbol in & Comparison result \\
\hline\(\square \$=\) & Non-conduction state & \(\square \$<\) & Non-conduction state \\
\hline\(\square \$<>\) & Conduction state & \(\square \$>=\) & Conduction state \\
\hline\(\square \$<=\) & Non-conduction state & \(\square \$>\) & Conduction state \\
\hline
\end{tabular}
(c) If the character string data are different, the first different sized character code will determine whether the character string is larger or smaller.
(51)
\begin{tabular}{|ll:ll|}
\hline 32 H & \((2)\) & 31 H & \((1)\) \\
\hline 34 H & \((4)\) & 33 H & \((3)\) \\
\hline 00 H & & 35 H & \((5)\) \\
\hline
\end{tabular}
(s2) \begin{tabular}{|ll:ll|}
\hline 32 H & \((2)\) & 31 H & (1) \\
\hline 33 H & \((3)\) & \(34^{\prime}\) & \((4)\)
\end{tabular}
\(33_{\mathrm{H}} \quad(3): 34 \mathrm{H} \quad\) (4)
00н
"12435"
\begin{tabular}{c|c|c|c}
\hline Instruction symbol in & Comparison result & Instruction symbol in & Comparison result \\
\hline\(\square \$=\) & Non-conduction state & \(\square \$<\) & Conduction state \\
\hline\(\square \$<>\) & Conduction state & \(\square \$>=\) & Non-conduction state \\
\hline\(\square \$<=\) & Conduction state & \(\square \$>\) & Non-conduction state \\
\hline
\end{tabular}
(d) If the character string data specified for \((11)\) and \(\S_{2}\) are of different lengths, the data with the longer character string will be larger.
(s1)
\begin{tabular}{|cc:cc|}
\hline 32 H & \((2)\) & 31 H & \((1)\) \\
\hline 34 H & \((4)\) & \(33_{\mathrm{H}}\) & \((3)\) \\
\hline 36 H & \((6)\) & 35 H & \((5)\) \\
\hline 00 H & & 37 H & \((7)\) \\
\hline
\end{tabular}
"1234567"
(s2) \begin{tabular}{|cc:cc|}
\hline 32 H & \((2)\) & 31 H & (1) \\
\hline 34 H & \((4)\) & 33 H & \((3)\) \\
\hline 36 H & \((6)\) & 35 H & \((5)\) \\
\hline 00 H & 00 H \\
\hline
\end{tabular}
"123456"
\begin{tabular}{c|c|c|c|}
\hline Instruction symbol in & Comparison result & Instruction symbol in & Comparison result \\
\hline\(\square \$=\) & Non-conduction state & \(\square \$<\) & Non-conduction state \\
\hline\(\square \$<>\) & Conduction state & \(\square \$>=\) & Conduction state \\
\hline\(\square \$<=\) & Non-conduction state & \(\square \$>\) & Conduction state \\
\hline
\end{tabular}
(4) The \(\mathrm{OR} \$=, \mathrm{OR} \$<>, \mathrm{OR} \$<=, \mathrm{OR} \$<, \mathrm{OR} \$>=\), or \(\mathrm{OR} \$>\) instruction performs comparison operation of operation result between "(ร1), ©2" and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400.
When using the OR\$=, OR\$<>, OR\$<=, OR\$<, OR\$>=, or OR\$> instruction, connect EN and ENO in series as shown below.


In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The code 00 H does not exist within the corresponding device range, starting from the device number specified for (①) and (s2).
(Error code: 4101)
- The character string of (s1) and (®2) exceeds 16383 characters.
(Error code: 4101)

\section*{ZPOINT}
1. The character string data comparison operation instruction checks the device range while comparing the character string data. For this reason, if the 00 H code does not exist in the corresponding device range, the instruction outputs the comparison operation result instead of returning an operation error when mismatch of characters is detected.


If (s1) and (s2) data are as shown above, the second character of (s1) does not match with that of s2), and the comparison result is expressed as \((51) \neq\left(s_{2}\right.\) (the operation result is "non-conduction"). Though the 00 H code is not included within the © \(₫ 1\) device range, no operation error is returned, because the mismatch is detected at D12287, which is within the device range.

\section*{\(\square\) Program Example}
(1) In the following program, the character string data stored in g_string1 is compared with the character string data stored in g_string2, and g_bool1 turns ON when they are matched.
[Structured ladder/FBD]

(2) In the following program, the character string g_string1 is compared with the character string data stored in g_string2 and when g_bool1 is ON, and g_bool2 turns ON when the character string data stored in g_string2 is other than g_string1.
[Structured ladder/FBD]

(3) In the following program, the character string data stored in g_string1 is compared with the character string data stored in g_string2 when g_bool1 is ON, and g_bool2 turns ON when the character string data stored in g_string2 is equal to or shorter than the character string data stored in g_string1, or when g_bool3 is ON.
[Structured ladder/FBD]

(4) In the following program, the character string data stored in g_string1 is compared with the character string g_string2, and g_bool3 turns ON when g_string2 is equal to or longer than the character string data stored in g_string1 or when both g_bool1 and g_bool2 are ON.
[Structured ladder/FBD]


\subsection*{6.1.6 16-bit BIN block data comparison}

BKCMP=, BKCMP<>, BKCMP<=, BKCMP<, BKCMP>=, BKCMP>

Basic Hilitomance Process Redundant Universal LCPU


\begin{tabular}{|ll|}
\hline & indicates any of the following \\
\hline instructions. & \\
\(B K C M P=\) & \(B K C M P=P\) \\
\(B K C M P<>\) & \(B K C M P<>P\) \\
\(B K C M P<=\) & \(B K C M P<=P\) \\
\(B K C M P<\) & \(B K C M P<P\) \\
\(B K C M P>=\) & \(B K C M P>=P\) \\
\(B K C M P>\) & \(B K C M P>P\) \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline \multirow[t]{4}{*}{Input argument,} & EN: & Executing condition & : Bit \\
\hline & s1: & Data to be compared, or start number of the device that stores : data to be compared & ANY16 \\
\hline & s2: & Start number of the device that stores comparison data & : ANY16 \\
\hline & n: & Number of data to be compared & : ANY16 \\
\hline \multirow[t]{2}{*}{Output argument,} & ENO: & Execution result & : Bit \\
\hline & d: & Start number of the device that stores the operation result & : Bit \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J)} & \multirow[b]{2}{*}{U..ig\%} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{\[
\begin{gathered}
\text { Constant } \\
\text { K, H }
\end{gathered}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & - & & & & & - & & \(\bigcirc\) & - \\
\hline (3) & - & & & & & - & & - & - \\
\hline n & \(\bigcirc\) & & & & & \(\bigcirc\) & & \(\bigcirc\) & - \\
\hline (d) & \(\bigcirc\) & & & & & - & & - & - \\
\hline
\end{tabular}

\section*{\(\mathcal{Z}\) Function}
(1) Compares the 16 -bit BIN data \(n\) points from the device number specified for \((91)\) with the 16bit BIN data \(n\) points from the device number specified for \(\Im_{\Omega}\), and stores the operation result to the device specified for (d) and the following devices.
(a) If the comparison condition has been met, the corresponding device of (d) turns ON.
(b) If the comparison condition has not been met, the corresponding device of © turns OFF.

(2) The comparison operation is performed in units of 16 bits.
(3) The constant can be specified between -32768 and 32767 (BIN 16 bits) for © (

(4) The results of the comparison operations for each instruction are as follows.
\begin{tabular}{|c|c|c|c|c|c|}
\hline Instruction symbol & Condition & Comparison result & Instruction symbol & Condition & Comparison result \\
\hline BKCMP= & (51) \(=\) s2 & \multirow{6}{*}{ON (1)} & BKCMP= & (s1) \(\neq\) ( \(2^{2}\) & \multirow{6}{*}{OFF (0)} \\
\hline BKCMP<> & (s1) \(\neq\) (s2) & & BKCMP<> & (s1) \(=\) S 2 & \\
\hline BKCMP<= & (s1) \(\leqq\) (s2) & & BKCMP<= & (51) \(>\) (s2) & \\
\hline BKCMP< & (51) < (s2) & & BKCMP< & (s1) \(\geqq\) (s2) & \\
\hline BKCMP>= & (51) \(\geqq\) (52) & & BKCMP>= & (51) \(<\) (s2) & \\
\hline BKCMP> & (51) \(>\) (s2) & & BKCMP> & (s1) \(\leqq\) ( 22 & \\
\hline
\end{tabular}
(5) If all comparison results stored to devices n points from (d) are ON (1), SM704 (block comparison signal) turns ON.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The range of the n points from the device specified for (s1), (s2) or (d)exceeds the corresponding device.
(Error code: 4101)
- The range of \(n\) points from the device specified for © \((51)\) overlaps with the range of \(n\) points from the device specified for (d).
(Error code: 4101)
- The range of \(n\) points from the device specified for \(\left(\Omega^{2}\right)\) overlaps with the range of \(n\) points from the device specified for (d).
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the data stored in D100 to D103 is compared with the data stored in R0 to R3 when X20 turns ON, and the result is stored to M10 and the following devices. [Structured ladder/FBD]

[Operation]

(2) In the following program, the constant K1000 is compared with the data stored in D10 to D13 when X1C turns ON, and the result is stored to b4 to b7 of D0.
[Structured ladder/FBD]

[Operation]

(3) In the following program, the data in D10 to D12 are compared with the data in D30 to D32 when X20 turns ON, and the result is stored to M100 and the following devices.
The character string "ALL ON" is transferred to Var_D100 and the following variables when all devices of M100 and the following turn to 1 (ON status).
[Structured ladder/FBD]


\section*{[Operation]}


\subsection*{6.1.7 32-bit BIN block data comparison}

DBKCMP=, DBKCMP<>, DBKCMP<=, DBKCMP<, DBKCMP>=, DBKCMP>

- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported
\(\left.\begin{array}{clll}\hline \text { DBKCMP } & (=)(P) & := \\ & (<>) & \left(\begin{array}{l}\text { DBKCMP= } \\ \text { DBKCMP<> }\end{array}\right. & : \neq \\ \text { DBKCMP<= } & : \leqq \\ \text { DBKCMP< } & :< \\ & (<=) & : \geqq \\ \text { DBKCMP>= } & :> \\ \text { DBKCMP> } & \Delta \\ \text { P: Executing condition } & \leq\end{array}\right)\)

\begin{tabular}{|ll|}
\hline & \\
instructions. & indicates any of the following \\
DBKCMP \(=\) & DBKCMP \(=P\) \\
DBKCMP<> & DBKCMP<>P \\
DBKCMP<= & DBKCMP<=P \\
DBKCMP< & DBKCMP<P \\
DBKCMP>= & DBKCMP>=P \\
DBKCMP> & DBKCMP>P \\
\hline
\end{tabular}
\begin{tabular}{lll} 
Input argument, & EN: & Executing condition \\
s1: Bit \\
& & \begin{tabular}{l} 
Data to be compared, or start number of the device that stores : ANY32 \\
data to be compared
\end{tabular} \\
& \(\mathrm{s} 2:\) & \begin{tabular}{l} 
Start number of the device that stores comparison data
\end{tabular} \\
Output argument, ANY32
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...1.n} & \multirow[b]{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant
\[
\mathrm{K}, \mathrm{H}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & - & \(\bigcirc\) & \(\bigcirc\) & & & - & & \(\bigcirc\) & - \\
\hline (s2) & - & \(\bigcirc\) & \(\bigcirc\) & & & - & & - & - \\
\hline n & - & \(\bigcirc\) & \(\bigcirc\) & & & \(\bigcirc\) & & \(\bigcirc\) & - \\
\hline (d) & \(\bigcirc\) & - & \(\bigcirc\) & & & - & & - & - \\
\hline
\end{tabular}

\section*{\(\sum\) Function}
(1) Compares the 32-bit BIN data \(n\) points from the device number specified for \(\subseteq\) © with the 32bit BIN data \(n\) points from the device number specified for \(\varsigma_{2}\), and stores the operation result to the device specified for (d) and the following devices.
(a) If the comparison condition has been met, the corresponding device of © turns ON.
(b) If the comparison condition has not been met, the corresponding device of (d) turns OFF.
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline & \multicolumn{2}{|l|}{\multirow[t]{2}{*}{(4) b31 b \({ }^{\text {c }}\)}} & \multicolumn{2}{|l|}{0} & \multicolumn{2}{|c|}{b31} & 0 & Operation result \\
\hline (51) +1 , & & & & (52) +1 , & (32) & 1000 (BIN) & (d) & OFF (0) \\
\hline (51) +3 , & (51) +2 & 2080 (BIN) & & 2) + & (2) +2 & 2000 (BIN) & \(+1\) & OFF (0) \\
\hline (51) +5 , & (51) +4 & 5060 (BIN) & \(\mathrm{n}=\) & (2) + & (32) +4 & 5060 (BIN) & (d) +2 & ON (1) \\
\hline (11) \(+(2 n-1)\), & (51) \(+(2 n-2)\) & 1106 (BIN) & & ( \(22+(2 n-1)\), & (32) \(+(2 n-2)\) & 1106 (BIN) & (d) \(+(\mathrm{n}-1)\) & ON (1) \\
\hline
\end{tabular}
(2) The comparison operation is performed in units of 32 bits.
(3) The constant can be specified between -2147483648 and 2147483647 (BIN 32 bits) for © (S1)
(51) +1 ,
\begin{tabular}{|c|c|c|c|}
\hline b31 & b0 & \[
\begin{aligned}
& \text { (s) }+1 \text {, } \\
& \text { (s2) }+3,
\end{aligned}
\] & (s2)
\[
\text { (s2) }+2
\] \\
\hline (51) 32800 (BIN) & \(>=\) & (52) +5 , & (32) +4 \\
\hline
\end{tabular}
(32) \(+(2 n-1)\),
\[
\text { (s2) }+(2 n-2)
\]

Operation result
\begin{tabular}{|rr|}
\hline ON & \((1)\) \\
\hline OFF & \((0)\) \\
\hline ON \((1)\) \\
\hline \hline OFF & \\
\hline
\end{tabular}
(4) (d) is specified in the device range other than \(n\) points from \(\$_{51}\) and \(n\) points from \(\Im_{2}\).
(5) The results of the comparison operations for each instruction are as follows.
\begin{tabular}{|c|c|c|c|c|c|}
\hline Instruction symbol & Condition & Comparison result & Instruction symbol & Condition & Comparison result \\
\hline DBKCMP= & (s1) \(=\) s2 & \multirow{6}{*}{ON (1)} & DBKCMP= & (s1) \(\neq\) ( 22 & \multirow{6}{*}{OFF (0)} \\
\hline DBKCMP<> & (s1) \(\neq\) (s2) & & DBKCMP<> & (s1) \(=\) s 2 & \\
\hline DBKCMP<= & (51) \(\leqq\) ( \(2^{2}\) & & DBKCMP<= & (s1) \(>\) s 2 ) & \\
\hline DBKCMP< & (s1) < © 2 ) & & DBKCMP< & (s1) \(\geqq\) ( 22 & \\
\hline DBKCMP>= & (51) \(\geqq\) (s2) & & DBKCMP>= & (s1) \(<\) (2) & \\
\hline DBKCMP> & (s1) \(>\) (s2) & & DBKCMP> & (s1) \(\leqq\) ( 2 ) & \\
\hline
\end{tabular}
(6) If all comparison results stored to devices n points from (d) are ON (1), or any of the result is OFF (0), the following special relays turn ON/OFF according to the condition.
\begin{tabular}{c|c|c|c|c|c|c|c|c} 
& & \multicolumn{2}{|c|}{ All comparison results are ON (1) } & \multicolumn{2}{c}{ Any of the comparison result is OFF (0) } \\
\cline { 3 - 8 } & No. & \begin{tabular}{c} 
Special \\
relay
\end{tabular} & \begin{tabular}{c} 
Initial \\
execution/ \\
scan
\end{tabular} & \begin{tabular}{c} 
Interrupt \\
(other than 145)/ \\
fixed scan \\
execution
\end{tabular} & \begin{tabular}{c} 
Interrupt \\
\((145)\)
\end{tabular} & \begin{tabular}{c} 
Initial \\
execution/ \\
scan
\end{tabular} & \begin{tabular}{c} 
Interrupt \\
(other than 145)/ \\
fixed scan \\
execution
\end{tabular} & \begin{tabular}{c} 
Interrupt \\
\((145)\)
\end{tabular} \\
\hline 1 & SM704 & ON & ON & ON & OFF & OFF & OFF \\
\hline 2 & SM716 & ON & - & - & OFF & - & - \\
\hline 3 & SM717 & - & ON & - & - & OFF & - \\
\hline 4 & SM718 & - & - & ON & - & - & OFF \\
\hline
\end{tabular}

For the standby type programs, special relays based on the call source program are turned ON/ OFF.
(7) No processing is performed if the value specified for n is 0 .

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- A negative value is specified for \(n\).
(Error code: 4100)
- The range of the n points from the device specified for (s1), (s2) or (d) exceeds the corresponding device.
(Error code: 4101)
- The range of n points from the device specified for \(\$ 11)\) overlaps with the range of n points from the device specified for (d).
(Error code: 4101)
- The range of n points from the device specified for \(\S_{2}\) ) overlaps with the range of n points from the device specified for (d).
(Error code: 4101)

\section*{Program Example}
(1) In the following program, the data stored in R0 to R5 are compared with the data stored in D20 to D25 when M0 turns ON, and the results are stored to Y0 to Y2.
[Structured ladder/FBD]



[Operation]
\begin{tabular}{c|c|}
\multicolumn{1}{c}{ b31 } & \multicolumn{1}{c}{ b0 } \\
\cline { 2 - 2 } R1,R0 & -2147483000 \\
R3,R2 & 0 \\
R5,R4 & 2147483000 \\
\cline { 2 - 3 } &
\end{tabular}
\(<>\)
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|c|}{b31} \\
\hline D21,D20 & -2147483000 \\
\hline D23,D22 & 1 \\
\hline D25,D24 & 2147482999 \\
\hline
\end{tabular}
\begin{tabular}{l|rr|}
\hline Y0 & OFF & \((0)\) \\
\cline { 2 - 3 } & Y1 & ON \\
\cline { 2 - 3 } & \((1)\) \\
\cline { 2 - 3 } & ON & \((1)\) \\
\hline
\end{tabular}
(2) In the following program, the constant is compared with the data stored in D0 to D9 when M0 turns ON, and the results are stored to D10.5 to D10.9.
[Structured ladder/FBD]


[Operation]
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline & & & & & & \\
\hline & & D1,D0 & -70000 & D10.5 & ON & (1) \\
\hline b31 & & D3,D2 & 50000 & D10.6 & OFF & (0) \\
\hline -60000 & > & D5,D4 & -32768 & \(\square \mathrm{D} 10.7\) & OFF & (0) \\
\hline & & D7,D6 & 32767 & D10.8 & OFF & (0) \\
\hline & & D9,D8 & 0 & D10.9 & OFF & (0) \\
\hline
\end{tabular}

\section*{XPOINT}

When bits of word device are specified, only the specified bit devices that store the operation results are changed.

(3) In the following scan program, the data in D0 to D5 are compared with the data in D10 to D15 when M0 turns ON, and the results are stored to M20 to M22.
The character string data "ALL ON" are transferred to D100 and the following devices when all devices of M20 to M22 turn ON.
[Structured ladder/FBD]

[Operation]
\begin{tabular}{c|c|}
\multicolumn{2}{c}{ b31 } \\
\cline { 2 - 2 } D1,D0 & -2147483000 \\
\cline { 2 - 3 } & 60000 \\
D3,D2 & -900000 \\
&
\end{tabular}


When all results are ON (1), special relays correspond to each program turn ON (1). (Since this program example is a scan program, SM704 and SM716 turn ON (1) and SM717 and SM718 do not change.)

\subsection*{6.2 Arithmetic Operation Instructions}

\subsection*{6.2.1 16-bit BIN data addition and subtraction}

\begin{tabular}{llll} 
Input argument, & EN: & \begin{tabular}{l} 
Executing condition \\
s1:
\end{tabular} & \begin{tabular}{l} 
Data to be added or subtracted, or start number of the device \\
that stores data to be added or subtracted
\end{tabular} \\
Output argument,
\end{tabular}\(\quad\)\begin{tabular}{ll} 
s2: & ENO:
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{Ji.)} & \multirow[b]{2}{*}{U)igat} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & \multicolumn{7}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (52) & \multicolumn{7}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) & \multicolumn{7}{|c|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{Function}

\section*{\(+(P)\)}
(1) Adds 16-bit BIN data specified for (31) to 16-bit BIN data specified for \(\Theta_{2}\) ) and stores the result of the addition to the variable specified for \({ }^{(d)}\).

(2) Values can be specified between -32768 and 32767 (BIN 16 bits) for ©15 and (22).
(3) The judgment of whether the value is positive or negative is made by the most significant bit (b15).
- 0: Positive
- 1: Negative
(4) The following is the result when an underflow or overflow is generated by the operation. The carry flag (SM700) in this case does not turn ON.


\section*{-(P)}
(1) Subtracts 16-bit BIN data specified for ©11 from 16-bit BIN data for (s2) and stores the result of the subtraction to the variable specified for (d).

(2) Values can be specified between -32768 and 32767 (BIN 16 bits) for (51) and ©2).
(3) The judgment of whether the value is positive or negative is made by the most significant bit (b15).
- 0: Positive
- 1: Negative
(4) The following is the result when an underflow or overflow is generated by the operation.

The carry flag (SM700) in this case does not turn ON.


\section*{O Operation Error}

No operation error occurs in the execution of the \(+(\mathrm{P})\) and \(-(\mathrm{P})\) instructions.

\section*{\(\triangle\) Program Example}

In the following program, the value in Var_D3 is added to the value in Var_D0 when X5 turns ON, and outputs the result to Var_D3.
[Structured ladder/FBD]


\subsection*{6.2.2 32-bit BIN data addition and subtraction}
\(\mathrm{D}+(\mathrm{P})\)
\(\mathrm{D}-(\mathrm{P})\)\(\quad(\mathrm{P}:\) Executing condition \(\quad: \pm\).


\section*{Function}

\section*{D+(P)}
(1) Adds 32-bit BIN data specified for © \({ }^{(1)}\) to 32-bit BIN data specified for \({ }_{(2)}\), and stores the result of the addition to the variable specified for (d).

(2) Values can be specified between -2147483648 and 2147483647 (BIN 32 bits) for ©1 and (®2).
(3) Judgment of whether the value is positive or negative is made on the basis of the most significant bit (b31).
- 0: Positive
- 1: Negative
(4) The following is the result when an underflow or overflow is generated by the operation. The carry flag (SM700) in this case does not turn ON.
\begin{tabular}{|c|c|c|c|}
\hline \[
\begin{aligned}
& \cdot \\
& \stackrel{2147483647}{(7 F F F F F F F H)}
\end{aligned}
\] & \[
+{ }_{(00000002 \mathrm{H})}
\] & \[
\begin{aligned}
& -2147483647 \\
& (80000001 \mathrm{H})
\end{aligned}
\] & Since bit 31 value is "1", result of operation takes a negative valu \\
\hline \[
\begin{aligned}
& -2147483648 \\
& (80000000 \mathrm{H})
\end{aligned}
\] & \[
+\underset{\text { (FFFFFFFEH) }}{ }
\] & \[
\begin{aligned}
& 2147483646 \cdot \\
& \text { (7FFFFFFEн) }
\end{aligned}
\] & Since bit 31 value is " 0 ", result of operation takes a positive value. \\
\hline
\end{tabular}

\section*{D-(P)}
(1) Subtracts 32-bit BIN data specified for \({ }_{(11)}\) from 32-bit BIN data specified for \(\left.{ }^{(22}\right)\) and stores the result of the subtraction to the variable specified for (d).

(2) Values can be specified between -2147483648 and 2147483647 (BIN 32 bits) for (11) and (82).
(3) Judgment of whether the value is positive or negative is made on the basis of the most significant bit (b31).
- 0: Positive
- 1: Negative
(4) The following is the result when an underflow or overflow is generated by the operation. The carry flag (SM700) in this case does not turn ON.

- \(2147483647--2 \longrightarrow-2147483647 \cdots\) Since bit 31 value is "1",
(7FFFFFFFH) (FFFFFFFEн) (80000001н) result of operation takes a negative value.

\section*{0 Operation Error}

No operation error occurs in the execution of the \(D+(P)\) and \(D-(P)\) instructions.

\section*{\(\triangle\) Program Example}
(1) In the following program, the value in Var_D0 is added to the value in Var_D2 when X0 turns ON, and the result is stored to Var_D10.
[Structured ladder/FBD]

(2) In the following program, the value in Var_D2 is subtracted from the value in Var_D0 when XOB turns ON, and the result is stored to Var_D10.
[Structured ladder/FBD]


\subsection*{6.2.3 16-bit BIN data multiplication and division}


\section*{Function}

\section*{*(P)}
(1) Multiplies 16-bit BIN data specified for ©11 and 16-bit BIN data specified for \(\varsigma_{2}\), and stores the result to the variable specified for (d).

(2) When (d) is a bit device, specification is made from the lower bits.

\section*{Example}
- K1•••••Lower 4 bits (b0 to b3)
- K4•••••Lower 16 bits (b0 to b15)
- K8••••• 32 bits (b0 to b31)
(3) Values can be specified between -32768 and 32767 (BIN 16 bits) for \((51)\) and \(\S_{2}\).
(4) Judgments whether (11), (32), (a) [0], and (a) [1] are positive or negative are made on the basis of the most significant bit (b15 for (s1) and ©2), and b31 for (d)).
- 0: Positive
- 1: Negative

\section*{/(P)}
(1) Divides 16-bit BIN data specified for ©1) and 16-bit BIN data specified for ®2 \(^{2}\), and stores the result to the variable specified for (d).

(2) Quotient and remainder of the division result are stored in 32-bit data.

Quotient : Stored to (d) [0] (16 bits).
Remainder: Stored to (d) [1] (16 bits).
(3) Values can be specified between -32768 and 32767 (BIN 16 bits) for \((\$ 1)\) and \(\S_{2}\).
(4) Judgment whether values for (51), (s2), (d) [0], and (d) [1] are positive or negative is made on the basis of the most significant bit (b15). (Sign is attached to both the quotient and remainder.)
- 0: Positive
- 1: Negative

\section*{0 Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- Divisor (22) is 0.

\section*{\(\triangle\) Program Example}
(1) In the following program, the BIN value 5678 is multiplied by the BIN value 1234 when X 5 turns ON, and the result is stored to Var_D3.
[Structured ladder/FBD]

(2) In the following program, the BIN value in Var_D0 is multiplied by the BIN value in Var_D1, and the result is stored to Var_D2.
[Structured ladder/FBD]

(3) In the following program, the value in Var_D0 is divided by 3.14 when X 3 turns ON, and the result is stored to Var_D1.
[Structured ladder/FBD]


\subsection*{6.2.4 32-bit BIN data multiplication and division}


\section*{Function}

\section*{D*(P)}
(1) Multiplies the 32-bit BIN data specified for © \((11)\) by the 32-bit BIN data specified for \({ }_{(22}\), and stores the result to the variable specified for (d).
\begin{tabular}{|c|c|c|c|c|}
\hline (51) & & (s2) & (d) [1] & (d) [0] \\
\hline b31-b16 b15--b0 & & b31-b16 b15--b0 & b63-b48 b47 & b16 b15--b0 \\
\hline 567890 (BIN) & \(\times\) & 123456 (BIN) & & (BIN) \\
\hline
\end{tabular}
(2) If (d) is a bit device, only the lower 32 bits of the multiplication result is considered, and the upper 32 bits cannot be specified.

\section*{Example}
- K1•••••Lower 4 bits (b0 to b3)
- K4•••••Lower 16 bits (b0 to b15)
- K8•••••Lower 32 bits (b0 to b31)
(3) Values can be specified between -2147483648 and 2147483647 (BIN 32 bits) for © \(\subseteq 1\) and (82).
(4) Judgments whether (s1), (®2), © [0], and () [1] are positive or negative are made on the basis of the most significant bit (b31 for (s1) and ©2, and b63 for (d)).
- 0: Positive
- 1: Negative

\section*{D/(P)}
(1) Divides the 32-bit BIN data specified for ©s by the 32-bit BIN data specified for \({ }_{(22}\), and stores the result to the variable specified for (d).

(2) Quotient and remainder of the division result are stored in 64-bit data.

Quotient : Stored to © [0] (32 bits).
Remainder: Stored to © [1] (32 bits).
(3) Values can be specified between -2147483648 and 2147483647 (BIN 32 bits) for ©11) and (82).
(4) Judgment whether values for (51), (32), (d) [0], and (©) [1] are positive or negative is made on the basis of the most significant bit (b31).
(Sign is attached to both the quotient and remainder.)
- 0: Positive
- 1: Negative

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- Divisor ©22 is 0 .
(Error code: 4100)

\section*{\(\square\) Program Example}
(1) In the following program, the BIN value in Var_D7 is multiplied by the BIN value in Var_D18 when X 5 turns ON, and the result is stored to Var_D1.
[Structured ladder/FBD]

(2) In the following program, the value in Var_D0 is multiplied by 314 when X 3 turns ON, and the result is stored to Var_D2.
[Structured ladder/FBD]


\subsection*{6.2.5 4-digit BCD data addition and subtraction}

\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{} \\
\hline instr & \\
\hline B+ & \(B+P\) \\
\hline B- & B-P \\
\hline
\end{tabular}
\begin{tabular}{lll} 
Input argument, & EN: & Executing condition \\
& \(\mathrm{s} 1:\) & \begin{tabular}{l} 
Data to be added or subtracted, or start number of the device \(:\) ANY16 \\
that stores data to be added or subtracted
\end{tabular} \\
Ontput argument, & ENO: & \begin{tabular}{l} 
Start number of the device that stores addition or subtraction \\
data
\end{tabular} \\
Execution result ANY16
\end{tabular}


\section*{Function}

\section*{\(B+(P)\)}
(1) Adds the 4-digit BCD data specified for ©11 and the 4-digit BCD data specified for ®2 \(^{2}\), and stores the result of the addition to the variable specified for © \({ }^{(d)}\).

\(+\) \(\square\) \begin{tabular}{|l|l|l|l|}
\hline 6 & 9 & 1 & 2 \\
\hline
\end{tabular}
(2) Values can be specified between 0 and 9999 (4-digit BCD) for (51), (22) and (d).
(3) If the result of the addition operation exceeds 9999, the higher bits are ignored. The carry flag (SM700) in this case does not turn ON.
\begin{tabular}{|l|l|l|l|}
\hline 6 & 4 & 3 & 2 \\
\hline
\end{tabular}\(+\)\begin{tabular}{|l|l|l|l|}
\hline 3 & 5 & 8 & 3 \\
\hline
\end{tabular}\(\Rightarrow\)\begin{tabular}{|l|l|l|l|}
\hline 0 & 0 & 1 & 5 \\
\hline
\end{tabular}

\section*{B-(P)}
(1) Subtracts the 4-digit BCD data specified for ©11 and the 4-digit BCD data specified for \(\mathrm{S}_{2}\), and stores the result of the subtraction to the variable specified for (d).

(2) Values can be specified between 0 and 9999 (4-digit BCD) for (51), (s2) and (d).
(3) The following is the result when an underflow is generated by the subtraction operation. The carry flag (SM700) in this case does not turn ON.
\begin{tabular}{|l|l|l|l|}
\hline 0 & 0 & 0 & 1 \\
\hline
\end{tabular}\(-\)\begin{tabular}{|l|l|l|l|}
\hline 0 & 0 & 0 & 3 \\
\hline
\end{tabular}\(\Rightarrow\)\begin{tabular}{|c|c|c|c|}
\hline 9 & 9 & 9 & 8 \\
\hline
\end{tabular}

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The (ㄷ1), (s2) (d) or BCD value is outside the range of 0 to 9999.

\section*{\(\triangle\) Program Example}
(1) In the following program, the BCD value in Var_D0 is added to the BCD value in Var_D1 when X20 turns ON, and the result is stored to Var_D2.
[Structured ladder/FBD]

(2) In the following program, the BCD value in Var_D20 is subtracted from the BCD value in Var_D10 when X20 turns ON, and the result is stored to Var_D30.
[Structured ladder/FBD]


\subsection*{6.2.6 8-digit BCD data addition and subtraction}

\section*{DB+, DB- \\ Basic \\ Process \\ Redundan \\ Universal \\ LCPU}
\begin{tabular}{ll}
\hline DB+(P) \\
DB-(P) & \(: \pm\)
\end{tabular}


\section*{DB+(P)}

(1) Adds the 8-digit BCD data specified for (①) and the 8-digit BCD data specified for (®2), and stores the result of the addition to the variable specified for (d).

(2) Values can be specified between 0 and 99999999 (8-digit BCD) for (s1) and \(\Omega_{2}\).
(3) If the result of the addition operation exceeds 99999999 , the upper bits will be ignored. The carry flag (SM700) in this case does not turn ON.

\section*{DB-(P)}
(1) Subtracts the 8-digit BCD data specified for © \(\mathrm{Sl}^{1}\) and the 8-digit BCD data specified for \(\Im_{2}\), and stores the result of the subtraction to the variable specified for (d).

(2) Values can be specified between 0 and 99999999 (8-digit BCD) for © \({ }^{(11)}\) and \(\mathrm{S}_{2}\).
(3) The following is the result when an underflow is generated by the subtraction operation. The carry flag (SM700) in this case does not turn ON.
\[
\begin{array}{|l|l|l|l|l|l|l|l|}
\hline 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
\end{array}-\begin{array}{|l|l|l|l|l|l|l|}
\hline 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
\end{array} \rightarrow \begin{array}{|l|l|l|l|l|l|l|}
\hline 9 & 9 & 9 & 9 & 9 & 9 & 9 \\
\hline
\end{array}
\]

\section*{Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The (s1), ②, (d) or BCD value is outside the range of 0 to 99999999. (Error code: 4100)

\section*{\(\triangle\) Program Example}
(1) In the following program, the BCD value in Var_D3 is added to the BCD value in Var_ \(Z\) when X20 turns ON, and the result is stored to Var_R10.
[Structured ladder/FBD]

(2) In the following program, the BCD data in Var_ \(Z\) is subtracted from the BCD data in Var_D3 when X20 turns ON, and the result is stored to Var_R10.
[Structured ladder/FBD]


\subsection*{6.2.7 4-digit BCD data multiplication and division}

\author{
B*, B/ \\ Basic \\ Hilith \\ performanc \\ Process \\ Redundan \\ Universal \\ LCPU
}


\section*{\(\sqrt{2}\) Function}

\section*{B* \({ }^{*}\) )}
(1) Multiplies BCD data specified for (51) and BCD data specified for (32), and stores the result to the variable specified for (d).

(2) Values can be specified between 0 and 9999 (4-digit BCD) for (51) and (52).

\section*{\(B /(P)\)}
(1) Divides BCD data specified for \((11)\) by BCD data specified for \(\Omega_{22}\), and stores the result to the variable specified for (d).

(2) Quotient and remainder of the division result are stored in 32-bit data.

Quotient (4-digit BCD) : Stored to © [0] (16 bits).
Remainder (4-digit BCD): Stored to © [1] (16 bits).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The (51) or (s2) BCD value is outside the range of 0 to 9999.
(Error code: 4100)
- Divisor (s2) is 0 .
(Error code: 4100)

\section*{\(\triangle\) Program Example}
(1) In the following program, the BCD value in Var_D7 is multiplied by the BCD value in Var_D8 when X20 turns ON, and the result is stored to Var_D0.
[Structured ladder/FBD]

[Operation]

(2) In the following program, the BCD value 5678 is divided by the \(B C D\) value 1234 , and the result is stored to Var_D502.
[Structured ladder/FBD]

[Operation]


\subsection*{6.2.8 8-digit BCD data multiplication and division}


\section*{Function}

DB* \({ }^{*}\) )
(1) Multiplies the 8-digit BCD data specified for \((11)\) and the 8-digit BCD data specified for \({ }^{2} 2\), and stores the product to the variable specified for (d).

(2) If (d) is a bit device, the lower 8 digits (lower 32 bits) of the multiplication result is considered, and the upper 8 digits (upper 32 bits) cannot be specified.

\section*{Example}
- K1•••••Lower 1 bits (b0 to b3)
- K4•••••Lower 4 bits (b0 to b15)
- K8•••••Lower 8 bits (b0 to b31)
(3) Values can be specified between 0 and 99999999 (8-digit BCD) for \(\Im_{1}\) and \(\S_{2}\).

\section*{DB/(P)}
(1) Divides 8-digit BCD data specified for (s1) by 8-digit BCD data specified for ®2 \(^{2}\), and stores the result to the variable specified for (a).

(2) Quotient and remainder of the division result are stored in 64-bit data.

Quotient (8-digit BCD) : Stored to © [0] (32 bits).
Remainder (8-digit BCD): Stored to © [1] (32 bits).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The (51) or ©2 BCD value is outside the range of 0 to 99999999 . (Error code: 4100)
- Divisor ©22 is 0 .

\section*{\(\square\) Program Example}
(1) In the following program, the BCD value 68347125 is multiplied by the BCD value 573682 , and the result is stored to Var_D502.
[Structured ladder/FBD]

[Operation]

(2) In the following program, the BCD value in Var_D0 is divided by the BCD value in Var_D8 when XOB turns ON, and the result is stored to Var_D765.
[Structured ladder/FBD]

[Operation]


\subsection*{6.2.9 Floating-point data addition and subtraction (single precision)}


\begin{tabular}{|c|c|c|c|}
\hline \multirow[t]{3}{*}{Input argument,} & EN: & Executing condition & : Bit \\
\hline & s1: & Data to be added or subtracted, or start number of the device that stores data to be added or subtracted & : Single-precision real \\
\hline & s2: & Addition or subtraction data, or start number of the device that stores addition or subtraction data & : Single-precision real \\
\hline \multirow[t]{2}{*}{Output argument,} & ENO: & Execution result & : Bit \\
\hline & d: & Start number of the device that stores the operation result & : Single-precision real \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{} & \multirow{2}{*}{U...1a} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & \(\bigcirc\) & - & \(\bigcirc\) & - \\
\hline (s2) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & \(\bigcirc\) & - & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & \(\bigcirc\) & - & - & - \\
\hline
\end{tabular}

\section*{Function}

\section*{E+(P)}
(1) Adds the 32-bit floating-point real number specified for ©(1) and the 32-bit floating-point real number specified for \({ }_{s 2}\), and stores the sum to the variable specified for (d).

(2) Values which can be specified for (①), (22) and (d), and which can be stored, are as follows: \(0,2^{-126} \leqq \mid\) specified value (storing value) \(\mid<2^{128}\)
(3) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{E-(P)}
(1) Subtracts the 32-bit floating-point real number specified for (51) and the 32-bit floating-point real number specified for \(\varsigma_{52}\), and stores the result to the variable specified for (d).

(2) Values which can be specified for (91), (s2) and (a), and which can be stored, are as follows.
\(0,2^{-126} \leqq \mid\) specified value (storing value) \(\mid<2^{128}\)
(3) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the specified device is outside the following range.
\(0,2^{-126} \leqq \mid\) value of the specified device \(\mid<2^{128}\)
(For Basic model QCPU and High Performance model QCPU)
(Error code: 4100)
- The value of the specified device is \(-0 .{ }^{* 1}\)
(For Basic model QCPU and High Performance model QCPU) (Error code: 4100)
*1 : There are CPU modules that will not result in an operation error if -0 is specified.
For details, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
- The operation result exceeds the following range. (An overflow occurs.) (For Universal model QCPU and LCPU)
\(2^{128} \leqq\) | operation result |
(Error code: 4141)
- The value of the specified device is -0 , unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)
```

E+, E-

```

\section*{\(\square\) Program Example}
(1) In the following program, the floating-point real number Var_D3 is added to the floating-point real number Var_D10 when X20 turns ON, and the result is stored to Var_R0.
[Structured ladder/FBD]

[Operation]

(2) In the following program, the floating-point real number Var_D20 is subtracted from the floating-point real number Var_D10, and the result is stored to Var_D30.
[Structured ladder/FBD]

[Operation]
\begin{tabular}{|c|c|c|}
\hline Var_D10 & Var_D20 & Var_D30 \\
\hline 97365.203 & 76059.797 & 21305.406 \\
\hline
\end{tabular}

\subsection*{6.2.10 Floating-point data addition and subtraction (double precision)}
\(E D+(P)\)
ED-(P)
P: Executing condition

\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multicolumn{2}{|l|}{\[
\begin{aligned}
& \mathrm{ED}+(\mathrm{P}) \\
& \text { ED-(P) }
\end{aligned}
\]} & \multicolumn{3}{|l|}{} & Exec & con & & \multicolumn{5}{|r|}{: \(\quad\) )} \\
\hline \multicolumn{3}{|l|}{Structured ladder/FBD} & \multicolumn{5}{|c|}{ST
Not supported} & \multicolumn{5}{|l|}{\(\square\) indicates any of the folowwing instructions.} \\
\hline Input argument, & \[
\begin{aligned}
& \text { EN: } \\
& \text { s1: } \\
& \text { s2: }
\end{aligned}
\] & \multicolumn{6}{|l|}{\begin{tabular}{l}
Executing condition \\
Data to be added or subtracted, or start number of the device that stores data to be added or subtracted Addition or subtraction data, or start number of the device that stores addition or subtraction data
\end{tabular}} & \multicolumn{5}{|l|}{\begin{tabular}{l}
: Bit \\
Double-precision real \\
Double-precision real
\end{tabular}} \\
\hline Output argument, & ENO:
d: & \multicolumn{6}{|l|}{\begin{tabular}{l}
Execution result \\
Start number of the device that stores the operation result
\end{tabular}} & \multicolumn{5}{|l|}{\begin{tabular}{l}
: Bit \\
: Double-precision real
\end{tabular}} \\
\hline & Argument & \multicolumn{2}{|l|}{Intemal device} & R, zR & Bit & Word & \multicolumn{2}{|l|}{U19:10]} & Zn & \[
\underset{\mathrm{E}}{\substack{\text { Constant }}}
\] & \multicolumn{2}{|l|}{Others} \\
\hline & (3) & - & & & & & \multicolumn{3}{|l|}{-} & \(\bigcirc\) & - & \\
\hline & (2) & - & &  & & & \multicolumn{3}{|l|}{-} & \(\bigcirc\) & - & \\
\hline & ( \({ }^{\text {d }}\) & - & & & & & \multicolumn{3}{|l|}{-} & - & - & \\
\hline
\end{tabular}

\section*{ED+(P)}
(1) Adds the 64-bit floating-point real number specified for © \(\mathrm{Sl}^{1}\) and the 64-bit floating-point real number specified for \({ }_{(\Omega 2}\), and stores the sum to the variable specified for (d).


64-bit floating-point real number


64-bit floating-point real number


64-bit floating-point real number
(2) Values which can be specified for (①), (32) and (d), and which can be stored, are as follows: \(0,2^{-1022} \leqq \mid\) specified value (storing value) \(\mid<2^{1024}\)
(3) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{ED-(P)}
(1) Subtracts the 64-bit floating-point real number specified for ©2 from the 64-bit floating-point real number specified for (s1), and stores the result to (d).

(2) Values which can be specified for (s1), (s2) and (d), and which can be stored, are as follows:
\(0,2^{-1022} \leqq \mid\) specified value (storing value) \(\mid<2^{1024}\)
(3) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the specified device is not within the following range. (Error code: 4140) \(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 . (Error code: 4140)
- The operation result exceeds the following range. (An overflow occurs.) \(2^{1024} \leqq\) | operation result |
(Error code: 4141)

\section*{\(\triangle\) Program Example}
(1) In the following program, the 64-bit floating-point real number Var_D3 is added to the 64-bit floating-point real number Var_D10 when X20 turns ON, and the result is stored to Var_R0.
[Structured ladder/FBD]

[Operation]

(2) In the following program, the 64-bit floating-point real number Var_D20 is subtracted from the 64-bit floating-point real number Var_D10, and the result is stored to Var_D30.
[Structured ladder/FBD]

[Operation]


\subsection*{6.2.11 Floating-point data multiplication and division (single precision)}

- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later
\(\left.\begin{array}{ll}E^{*}(P) \\ E /(P) & : \pm\end{array}\right)\)


\section*{Function}

\section*{E*(P)}
(1) Multiplies the 32-bit floating-point real number specified for (31) by the 32-bit floating-point real number specified for \(\Im_{2}\) and stores the operation result to the variable specified for (d).

(2) Values which can be specified for (51), (s2) and (d), and can be stored, are as follows.
\(0,2^{-126} \leqq \mid\) specified value (storing value) \(\mid<2^{128}\)
(3) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{\(E /(P)\)}
(1) Divides the 32-bit floating-point real number specified for © \(\mathrm{St}^{1}\) by the 32-bit floating-point real number specified for (s2) and stores the operation result to the variable specified for (d).

(2) Values which can be specified for (51), (22) and (a), and can be stored, are as follows.
\(0,2^{-126} \leqq \mid\) specified value (storing value) \(\mid<2^{128}\)
(3) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the specified device is outside the following range.
\(0,2^{-126} \leqq \mid\) value of the specified device \(\mid<2^{128}\)
(For Basic model QCPU and High Performance model QCPU)
(Error code: 4100)
- The value of the specified device is \(-0 .{ }^{* 1}\)
(For Basic model QCPU and High Performance model QCPU) (Error code: 4100)
*1 : There are CPU modules that will not result in an operation error if -0 is specified.
For details, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
- The operation result exceeds the following range. (An overflow occurs.)
(For Universal model QCPU and LCPU)
\(2^{128} \leqq \mid\) operation result |
(Error code: 4141)
- The value of the specified device is -0 , unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)

\section*{\(\triangle\) Program Example}
(1) In the following program, the 32-bit floating-point real number in Var_D3 is multiplied by the 32-bit floating-point real number in Var_D10, and the result is stored to Var_R0.
[Structured ladder/FBD]

[Operation]

(2) In the following program, the 32-bit floating-point real number in Var_D10 is divided by the 32-bit floating-point real number in Var_D20, and the result is stored to Var_D30.
[Structured ladder/FBD]

[Operation]
\begin{tabular}{|c|c|c|c|}
\hline Var_D10 & & Var_D20 & Var_D30 \\
\hline 52171.39 & 1 & 9.73521 & 5359.041 \\
\hline
\end{tabular}

\subsection*{6.2.12 Floating-point data multiplication and division (double precision)}

ED*, ED/


ED* \((P)\)
ED/(P)
P: Executing condition


\section*{Function}

\section*{ED* \({ }^{*}\) )}
(1) Multiplies the 64-bit floating-point real number specified for (51) and the 64-bit floating-point real number specified for \(\Im_{2}\), and stores the result to the variable specified for (d).


64-bit floating-point real number


64-bit floating-point real number
(d)


64-bit floating-point real number
(2) Values which can be specified for (51), (22) and (d), and can be stored, are as follows:
\(0,2^{-1022} \leqq \mid\) specified value (storing value) \(\mid<2^{1024}\)
(3) When the operation results in -0 or an underflow, the result is processed as 0 .
(4) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{ED/(P)}
(1) Divides the 64-bit floating-point real number specified for (51) by the 64-bit floating-point real number specified for © \({ }_{2}\), and stores the result to (d).


64-bit floating-point real number


64-bit floating-point real number


64-bit floating-point real number
(2) Values which can be specified for (51), (22) and (d), and can be stored, are as follows.
\(0,2^{-1022} \leqq \mid\) specified value (storing value) \(\mid<2^{1024}\)
(3) When the operation results in -0 or an underflow, the result is processed as 0 .
(4) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value of the specified device is not within the following range. (Error code: 4140)
\(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 .
(Error code: 4140)
- The value of \((2)\) in the division instruction is 0 .
(Error code: 4100)
- The operation result exceeds the following range. (An overflow occurs.) \(2^{1024} \leqq\) | operation result |
(Error code: 4141)

\section*{\(\triangle\) Program Example}
(1) In the following program, the 64-bit floating-point real number in Var_D3 is multiplied by the 64-bit floating-point real number in Var_D10, and the result is stored to Var_R0.
[Structured ladder/FBD]

[Operation]

(2) In the following program, the 64-bit floating-point real number in Var_D10 is divided by the 64-bit floating-point real number in Var_D20, and the result is stored to Var_D30.
[Structured ladder/FBD]

[Operation]


\subsection*{6.2.13 16-bit BIN block data addition and subtraction}


Input argument,
\begin{tabular}{ll} 
EN: & Executing condition \\
s1: & Start number of the device that stores data to be added or \\
& subtracted
\end{tabular}
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{} \\
\hline instr & \\
\hline BK+ & \(B K+P\) \\
\hline BK- & BK-P \\
\hline
\end{tabular}
ourgument,
n
n: Number of addition or subtraction data : ANY16
ENO: Execution result : Bit
\(\mathrm{d}: \quad\) Start number of the device that stores the addition or \(:\) ANY16 subtraction operation result
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...al} & \multirow[b]{2}{*}{U} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant \$} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline (s2) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline n & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{Function}

\section*{BK+(P)}
(1) Adds BIN data \(n\) points from the device specified for ©1 and BIN data \(n\) points from the device specified for (s2), and stores the results to (d) and the following devices.

(2) Block addition is performed in units of 16 bits.
(3) The constant can be specified between -32768 and 32767 (BIN 16 bits) for (s2).


(4) The following is the result when an underflow or overflow is generated by the operation. The carry flag (SM700) in this case does not turn ON.
\[
\begin{aligned}
& \text { - } 32767+2 \longrightarrow-32767 \\
& \text { (7FFFH) (0002H) (8001H) } \\
& \underset{\substack{-3001 н)}}{-32767} \underset{(\text { FFFEH })}{ } 32767 \text { (7FFFH) }
\end{aligned}
\]

\section*{BK-(P)}
 device specified for © \({ }^{(11)}\), and stores the results to (d) and the following devices.
(51)
(51 +1
(51) +2

(s2)
(s2) +1
(s2 +2
(s2) \(+(n-2)\)
\begin{tabular}{|ll|}
\hline 1234 & \((\mathrm{BIN})\) \\
\hline 5678 & \((\mathrm{BIN})\) \\
\hline 9876 & \((\mathrm{BIN})\) \\
\hline 4321 & \((\mathrm{BIN})\) \\
\hline 4000 & \((\mathrm{BIN})\) \\
\hline
\end{tabular}
(d) \(\left.\begin{array}{l}\text { (d) }+1 \\ \text { (d) }+2 \\ \begin{array}{|rr|}\hline 7531 & (\mathrm{BIN}) \\ \hline-551 & (\mathrm{BIN}) \\ \text { (d) }+(n-2) & \text { (BIN) } \\ \text { (d) }+(n-1) & 379 \\ \hline & \text { (BIN) } \\ \hline\end{array}\end{array}\right)\).
(2) Block subtraction is performed in units of 16 bits.
(3) The constant can be specified between -32768 and 32767 (BIN 16 bits) for \(\left.{ }^{(22}\right)\).

\begin{tabular}{l} 
b15---------b0 \\
\hline \(8880 \quad(B I N)\) \\
\hline
\end{tabular}

(4) The following is the result when an underflow or overflow is generated by the operation. The carry flag (SM700) in this case does not turn ON.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{-32768-2} \\
\hline (8000h) & (0002H) & (7FFEн) \\
\hline - 32767 & -2 & -32767 \\
\hline (7FFFH) & (FFFEн) & (8001н) \\
\hline
\end{tabular}

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The range of \(n\) points from the device specified for (s1), (®2) or (d) exceeds the corresponding device.
(Error code: 4101)
- The device ranges of ©s1) and (d) overlap. (Except when the same device is assigned to © 51 and (d)
(Error code: 4101)
- The device ranges of \({ }^{2} 2\) ) and (d) overlap. (Except when the same device is assigned to \({ }^{2}\) ) and (d)
(Error code: 4101)

\section*{Program Example}
(1) In the following program, the data stored in D100 to D103 are added to the data stored in R0 to R3 when X20 turns ON, and the results are stored to D200 and the following devices.
[Structured ladder/FBD]

[Operation]
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline \multicolumn{2}{|l|}{b15---------b0} & \multirow{5}{*}{+} & \multirow[b]{2}{*}{R0} & \multirow[t]{2}{*}{\[
\begin{aligned}
& \text { b15-- } 1234(\mathrm{BIN}) \\
& \hline
\end{aligned}
\]} & \multirow{5}{*}{\[
\square
\]} & \multirow[b]{2}{*}{D200} & \multicolumn{2}{|l|}{b15---------b0} \\
\hline D100 & 6789 (BIN) & & & & & & 8023 & (BIN) \\
\hline D101 & 7821 (BIN) & & R1 & 2032 (BIN) & & D201 & 9853 & (BIN) \\
\hline D102 & 5432 (BIN) & & R2 & -3252 (BIN) & & D202 & 2180 & (BIN) \\
\hline D103 & 3520 (BIN) & & R3 & -1000 (BIN) & & D203 & 2520 & (BIN) \\
\hline
\end{tabular}
(2) In the following program, the constant 8765 is subtracted from the data in D100 to D102 when X1C is turned ON, and the results are stored to R0 and the following devices.
[Structured ladder/FBD]

[Operation]


\subsection*{6.2.14 32-bit BIN block data addition and subtraction}

DBK+, DBK-

- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported


\section*{3 Function}

\section*{DBK+}
(1) Adds 32-bit BIN data \(n\) points from the device specified for © \(\mathrm{S}_{1}\) and 32-bit BIN data \(n\) points from the device specified for \((22\) ) or a constant, and stores the results to (d) and the following devices.
- When a device is specified for \(\S_{2}\)
\begin{tabular}{ll} 
(51 \()+1\), & (s1) \\
(51) +3, & (s1) +2 \\
(51 \()+5\), & (51) +4 \\
(51) \(+(2 n-1)\), & (51) \(+(2 n-2)\)
\end{tabular}

(22) +2
(s2) +2
(s2) +4
(s2) \(+(2 n-2)\)
b31 b0
\begin{tabular}{|ll}
\hline \(50000(\mathrm{BIN})\) \\
\hline \(20000(\mathrm{BIN})\) \\
\hline\(-10000(\mathrm{BIN})\) \\
\hline\(-20000(\mathrm{BIN})\) & \begin{tabular}{l} 
(d) +1, \\
(d) +3, \\
\hline
\end{tabular} \\
(d) +5, \\
(d) \(+(2 n-1)\),
\end{tabular}
(d)
d +2
(d) \(+(2 n-2)\)

- When a constant is specified for (s2)
\begin{tabular}{ll} 
S1 +1, & S1) \\
S1 +3, & S1 +2 \\
(s1 +5, & (51) +4
\end{tabular}
(s1) \(+(2 n-1)\),

(d) +1 ,
(d) +3 ,
(d)
d +2
b31
\begin{tabular}{|rr|}
\hline 20000 & \((\mathrm{BIN})\) \\
\hline 90000 & \((\mathrm{BIN})\) \\
\hline 0 & \((\mathrm{BIN})\) \\
\hline 110000 & \((\mathrm{BIN})\) \\
\hline
\end{tabular}
(2) Block addition is performed in units of 32 bits.
(3) The constant can be specified between -2147483648 and 2147483647 (BIN 32 bits) for (22).
(4) No processing is performed if the value specified for n is 0 .
(5) The following is the result when an overflow is generated by the operation.

The carry flag (SM700) in this case does not turn ON.
```

. K2147483647+K2\longrightarrow K-2147483647
(7FFFFFFFFH) (00000002H)(80000001H)

```

K-2147483647 +K-2 \(\longrightarrow\) K2147483647 (80000001н) (FFFFFFFEн) (7FFFFFFFн)

\section*{DBK-}
(1) Subtracts 32-bit BIN data \(n\) points from the device specified for \(\Omega_{2}\), or a constant from 32-bit BIN data \(n\) points from the device specified for (51), and stores the results to (d) and the following devices.
- When a device is specified for \({ }_{\text {s }}\)

- When a constant is specified for (s2)

(2) Block subtraction is performed in units of 32 bits.
(3) The constant can be specified between -2147483648 and 2147483647 (BIN 32 bits) for \(\Omega_{2}\).
(4) No processing is performed if the value specified for n is 0 .
(5) A device other than the device ranges of \(n\) points from \((51)\) and \(n\) points from \(\varsigma_{2}\) is specified for (d). Note that a same device can be specified for (51) and (®2).
(6) The following is the result when an overflow is generated by the operation. The carry flag (SM700) in this case does not turn ON.
```

. K2147483647 -K-2 \longrightarrowK-2147483647
(7FFFFFFFFн) (00000002н) (80000001н)
. K-2147483647 -K2 W2147483647
(80000001н) (FFFFFFFFEн)(7FFFFFFFH)

```

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- A negative value is specified for \(n\).
(Error code: 4100)
- The range of \(n\) points from the device specified for (s1), (®2) or (d) exceeds the corresponding device.
(Error code: 4101)
- The device ranges of \(n\) points from (51) and \(n\) points from (d) overlap. (Except when the same device is assigned to (51) and (d))
(Error code: 4101)
- The device ranges of \(n\) points from \({ }^{2} 2\) and \(n\) points from © overlap. (Error code: 4101)

\section*{\(\triangle\) Program Example}
(1) In the following program, the data stored in R0 to R5 are added to the constant when M0 turns ON, and the results are stored to D30 to D35.
[Structured ladder/FBD]

[Operation]
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multicolumn{2}{|c|}{b31} & b0 & & \multicolumn{2}{|c|}{b31} \\
\hline R1,R0 & 600000 & & & D31,D30 & 723456 \\
\hline R3,R2 & -800000 & + & 123456 & \(\square\) D33,D32 & -676544 \\
\hline R5,R4 & -123456 & & & D35,D34 & 0 \\
\hline
\end{tabular}
(2) In the following program, the data stored in D50 to D59 are subtracted from the data in D100 to D109 when M0 turns ON, and the results are stored to R100 to R109.
[Structured ladder/FBD]

[Operation]
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multicolumn{2}{|c|}{b31} & \multicolumn{2}{|c|}{b31} & \multicolumn{2}{|c|}{b31} \\
\hline D101,D100 & 12345 & D51,D50 & 11111 & R101,R100 & 1234 \\
\hline D103,D102 & 54321 & D53,D52 & -11111 & \(\square \mathrm{R} 103, \mathrm{R} 102\) & 65432 \\
\hline D105,D104 & -12345 & D55,D54 & 22222 & R105,R104 & -34567 \\
\hline D107,D106 & -54321 & D57,D56 & -22222 & R107,R106 & -32099 \\
\hline D109,D108 & 99999 & D59,D58 & 33333 & R109,R108 & 66666 \\
\hline
\end{tabular}

\subsection*{6.2.15 Character string data concatenation}


Input argument,
EN: Executing condition
Bit
s1: Connection data, or start number of the device that stores
String connection data
s2: \(\quad\) Data to be connected, or start number of the device that stores: String
data to be connected
Output argument,
ENO: Execution result
: Bit
\(\mathrm{d}: \quad\) Start number of the device that stores the operation result
String
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J......} & \multirow[b]{2}{*}{U...an!} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant \$} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (1) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (2) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Connects the character string data specified for ©2) after the character string data specified for (s1) and stores the result to the variable specified for (d).

\begin{tabular}{|cc:cc|}
\hline 46 H & (F) & 48 H & \((\mathrm{H})\) \\
\hline 2D & \((-)\) & \(41_{\mathrm{H}}\) & (A) \\
\hline \multicolumn{3}{c|}{\(00_{\mathrm{H}}\)} \\
\hline
\end{tabular}
\(+\)
(s2)

(d)
\begin{tabular}{|c|c|}
\hline 46н (F) & 48H (H) \\
\hline 2Dн (-) & 41н (A) \\
\hline 35н (5) & 31н (1) \\
\hline 39н (9) & 33 \({ }_{\text {H }}\) (3) \\
\hline 00H & 41н (A) \\
\hline
\end{tabular}
d
\(\square\)


\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The entire character strings cannot be stored to the variable specified for © .
(Error code: 4101)
- The variables specified for \((51)\) and \(\Im_{2}\) overlap.
(Error code: 4101)
- The variables specified for (s2) and (d) overlap.
(Error code: 4101)
- The character string of (s1), (32) and (d) exceeds 16383 characters.
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the character string "ABCD" is connected to the character string stored in Var_D10 when X0 turns ON, and the results are stored to Var_D100 and the following devices.
[Structured ladder/FBD]


\subsection*{6.2.16 16-bit BIN data increment and decrement}

INC(P)
DEC(P)
P: Executing condition


Input argument,
EN: Executing condition
: Bit
Output argument,
ENO: Execution result
: Bit
d: Start number of the device (16-bit data) to be incremented or : ANY16
decremented by 1
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & Inter & vice & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J\%} & \multirow[t]{2}{*}{U} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (d) & \multicolumn{7}{|c|}{\(\bigcirc\)} & \multicolumn{2}{|c|}{-} \\
\hline
\end{tabular}

INC(P)
(1) Adds 1 to the variable (16-bit data) specified for (d).


\begin{abstract}

\end{abstract}


-
U-in
\(\square\) 1 \(\qquad\)

(2) When the instruction is executed with the variable specified for © , which value is 32768 , the value -32768 is stored to the variable specified for (d).
DEC(P)
(1) Subtracts 1 from the variable (16-bit data) specified for © .

(2) When the instruction is executed with the variable specified for (d), which value is -32768 , the value 32767 is stored to the device specified for © .

\section*{Operation Error}

No operation error occurs in the execution of the INC(P) and DEC(P) instructions.

\section*{\(\triangle\) Program Example}
(1) In the following program, the current values from the counter C 0 to C 20 are output to Y 30 to Y3F in BCD data each time X8 turns ON. (When current value is less than 9999)
[Structured ladder/FBD]

[ST]
BCDP(X8,C0Z1,K4Y30);
INCP(X8,Z1);
IF Z1=21 OR X7 THEN
RST(TRUE,Z1);
END_IF;
(2) The following is a down-counter program.
[Structured ladder/FBD]

[ST]
MOVP(X7,100,D8);
IF X8 AND NOT(M38) THEN
DECP(TRUE,D8);
END_IF;
OUT(D8=0,M38);

\subsection*{6.2.17 32-bit BIN data increment and decrement}


\section*{Function}

\section*{DINC(P)}
(1) Adds 1 to the variable (32-bit data) specified for © \({ }^{(d)}\).

(2) When the instruction is executed with the variable specified for © , which value is 2147483647 , the value -2147483648 is stored to the variable specified for (d).

\section*{DDEC(P)}
(1) Subtracts -1 from the variable (32-bit data) specified for © (.

(2) When the instruction is executed with the variable specified for © , which value is 0 , the value -1 is stored to the variable specified for (d).

\section*{Operation Error}

No operation error occurs in the execution of the \(\operatorname{DINC}(P)\) and \(\operatorname{DDEC}(P)\) instructions.

\section*{\(\triangle\) Program Example}
(1) In the following program, 1 is added to the data in Var_D0 when X0 turns ON.
[Structured ladder/FBD]

[ST]
DINCP(X0,Var_D0);
(2) In the following program, 1 is added to the data set in Var_D0 when X0 turns ON, and the result is stored to Var_D3.
[Structured ladder/FBD]

[ST]
DMOVP(X0,Var_D0,Var_D3);
DINCP(X0,Var_D3);
(3) In the following program, 1 is subtracted from the data in Var_D0 when X0 turns ON.
[Structured ladder/FBD]

[ST]
DDECP(X0,Var_D0);
(4) In the following program, 1 is subtracted from the data set in Var_D0 when X0 turns ON, and the result is stored to Var_D3.
[Structured ladder/FBD]

[ST]
DMOVP(X0,Var_D0,Var_D3);
DDECP(X0,Var_D3);

\subsection*{6.3 Data Conversion Instructions}

\subsection*{6.3.1 BIN data to 4-/8-digit BCD data conversion}

BCD, DBCD

Process
Redundant
Universal
LCPU
\(B C D(P)\)
DBCD(P)

> P: Executing condition

5

\(\begin{array}{lll}\text { Input argument, } & \text { EN: } & \text { Executing condition } \\ & \mathrm{s}: & \text { Start number of the device that stores BIN data } \\ \text { Output argument, } & \text { ENO: } & \text { Execution result } \\ & \mathrm{d}: & \text { Start number of the device that stores converted BIN data }\end{array}\)
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...al} & \multirow[b]{2}{*}{U:...icain} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant
\[
\mathrm{K}, \mathrm{H}
\]} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & \multicolumn{7}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) & \multicolumn{7}{|c|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{T Function}

\section*{\(B C D(P)\)}

Converts BIN data (0 to 9999) in the device specified for (s) to BCD data, and stores the result to the device specified for (d).


\section*{DBCD(P)}

Converts BIN data (0 to 99999999) in the device specified for © to BCD data, and stores the result to the device specified for (d).
(s)
(S) BIN 99999999

 Must always be "0" (upper 5 digits). BCD conversion

(d) BCD 99999999

(d)

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The data of ©s is other than 0 to 9999 at the execution of the \(\mathrm{BCD}(\mathrm{P})\) instruction.
(Error code: 4100)
- The data of ©s is other than 0 to 99999999 at the execution of the \(\operatorname{DBCD}(\mathrm{P})\) instruction. (Error code: 4100)

\section*{\(\triangle\) Program Example}
(1) In the following program, the current value of C 4 from Y20 to Y2F is output to the BCD display device.


7-segment display unit
[Structured ladder/FBD]
[ST]
BCDP(SM400,CN4,K4Y20);
(2) In the following program, 32-bit data from D0 to D1 are output to devices from Y40 to Y67.

[Structured ladder/FBD]

[ST]
IF SM400 THEN
Var_D2[0]:=Var_D0/10000;
DBCDP(TRUE,Var_D2[1],K6Y50);
DBCD(TRUE,Var_D2[0],K4Y40);
END_IF;

\subsection*{6.3.2 4-/8-digit BCD data to BIN data conversion}

BIN, DBIN



\section*{Function}

\section*{BIN(P)}

Converts the BCD data (0 to 9999) in the device specified for ©s to the BIN data, and stores the result to the device specified for (d).
(s) BCD 9999


Thousands place Hundreds place \(\square_{\text {TIN conversion }}^{\text {Tens place }}\) Units place
(d) BIN 9999


\section*{DBIN(P)}

Converts the BCD data (0 to 99999999) in the device specified for ©s to the BIN data, and stores the result to the device specified for (d).


\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- Values other than 0 to 9 are specified for each digit of (s).
(Error code: 4100)
The error above can be suppressed by turning ON SM722.
However, the instruction is not executed regardless of whether SM722 is turned ON or OFF if the specified value is out of the available range.
For the BINP and DBINP instruction, the next operation will not be performed until the command (executing condition) is turned from OFF to ON regardless of the presence or absence of an error.

\section*{\(\square\) Program Example}
(1) In the following program, the BCD data in X10 to X1B are converted to the BIN data when X8 turns ON, and the result is stored to Var_D8.

[Structured ladder/FBD]

[ST]
BINP(X8,K3X10,Var_D8);
(2) In the following program, the BCD data in X 10 to X 37 are converted to BIN data when X 8 turns ON, and the result is stored to Var_D0.
(Addition of the BCD data in X20 to X37 converted to BIN data and the BCD data in X10 to X1F converted to BIN data.)

[Structured ladder/FBD]

[ST]
IF X8 THEN
DBINP(TRUE,K6X20,Var_D9);
Var_D5[0]:=Var_D9*10000;
BIN(TRUE,K4X10,Var_D3);
INT_TO_DINT_E(TRUE,Var_D3,Var_D4);
Var_D0:=Var_D4+Var_D5[0];
END_IF;
If a BCD value which exceeds 2147483647 is specified for X 10 to X 37 , the value of Var_D0 becomes a negative value, because it exceeds the range of numeric values that can be handled by a 32-bit device.

\subsection*{6.3.3 16-/32-bit BIN data to floating-point data conversion (single precision)}

FLT, DFLT
- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later

FLT(P) DFLT(P)

P: Executing condition

位



Input argument,
EN: Executing condition
s: Integer data to be converted to 32-bit floating-point data, or start number of the device that stores integer data
Output argument, ENO: Execution result
d: \(\quad\) Start number of the device that stores converted 32-bit
floating-point data
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..ala} & \multirow{2}{*}{U...igan} & \multirow[t]{2}{*}{Zn} & & \multirow[t]{2}{*}{Constant K, H} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & & \\
\hline (s) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & & \(\bigcirc\) & & \(\bigcirc\) & & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & \(\bigcirc\) & & - & & - \\
\hline
\end{tabular}

\section*{Function}

\section*{FLT(P)}
(1) Converts 16-bit BIN data specified for (s) to the 32-bit floating-point real number, and stores the result to the device specified for (d).

(2) BIN values between -32768 and 32767 can be specified for ©s.

\section*{DFLT(P)}
(1) Converts 32-bit BIN data specified for ©s to the 32-bit floating-point real number, and stores the result to the device specified for (d).

(2) BIN values between -2147483648 and 2147483647 can be specified for © \(^{\text {(2). }}\)
(3) Due to the fact that 32-bit floating-point real numbers are processed by 32-bit single precision, the number of significant figures is 24 bits if the display is binary and approximately 7 digits if the display is decimal.

For this reason, if the integer value exceeds the range of -16777216 to 16777215 (24-bit BIN value), errors can be generated in the conversion value.
As for the conversion result, the 25th bit from the highest bit of the integer value is rounded off and the 26th bit and later are truncated.


\section*{O Operation Error}

No operation error occurs in the execution of the FLT(P) and DFLT(P) instructions.

\section*{\(\square\) Program Example}
(1) In the following program, the 16-bit BIN value in Var_D20 is converted to the 32-bit floatingpoint real number, and the result is stored to Var_D0.
[Structured ladder/FBD]


\section*{[ST] \\ FLTP(SM400,Var_D20,Var_D0);}
[Operation]
\(\underset{\)\begin{tabular}{c}
\text { BIN value }
\end{tabular}\(}{\text { Var_D20 }}\)\begin{tabular}{l} 
Integer \\
conversion
\end{tabular}\(\overbrace{\)\begin{tabular}{l}
\text { 32-bit floating-point } \\
\text { real number }
\end{tabular}}\(^{\text {Var_D0 }}\)
(2) In the following program, the 32-bit BIN value in Var_D20 is converted to the 32-bit floatingpoint real number, and the result is stored to Var_D0.
[Structured ladder/FBD]

[ST]
DFLTP(SM400,Var_D20L,Var_D0);
[Operation]
\begin{tabular}{|c|c|c|}
\hline Var_D20 & Integer conversion & Var_D0 \\
\hline 16543521 &  & 16543521 \\
\hline BIN value
Var_D20 & \multirow[t]{5}{*}{Integer conversion} & \begin{tabular}{l}
32-bit floating-point real number \\
An error is generated in operation results since the number of
\end{tabular} \\
\hline 173963112 & & significant figures is "7" (The integer value exceeded the range \\
\hline \multirow[t]{3}{*}{BIN value} & & \begin{tabular}{l}
of -16777216 to 16777215 (24-bit BIN value)). \\
Var_D0
\end{tabular} \\
\hline & & 173963120 \\
\hline & & 32-bit floating-point real number \\
\hline
\end{tabular}

\subsection*{6.3.4 16-/32-bit BIN data to floating-point data conversion (double precision)}

FLTD, DFLTD


FLTD (P)
DFLTD (P)
\(\left.\begin{array}{ll}\text { FLTD }(P) \\ \text { DFLTD }(P) & : \simeq\end{array}\right)\)


Input argument,

EN: Executing condition
s: Integer data to be converted to 64-bit floating-point data, or start number of the device that stores integer data
ENO: Execution result
d: \(\quad\) Start number of the device that stores converted 64-bit floating-point data
\begin{tabular}{|ll|}
\hline FLTD & FLTDP \\
instructions. & DFLTDP \\
\\
\\
\\
\\
\\
\\
\hline
\end{tabular}
: Bit
Word [signed]/double word [signed]
: Bit
Double-precision real
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...al} & \multirow[t]{2}{*}{U等:} & \multirow[t]{2}{*}{Zn} & & \multirow[t]{2}{*}{Constant K, H} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{3}{|c|}{-} & \multicolumn{3}{|c|}{\(\bigcirc\)} & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{3}{|c|}{-} & \multicolumn{3}{|c|}{-} & - \\
\hline
\end{tabular}

\section*{\(\sqrt{3}\) Function}

\section*{FLTD(P)}
(1) Converts 16-bit BIN data specified for ©s to a 64-bit floating-point real number, and stores the result to © .

(2) BIN values between -32768 and 32767 can be specified for (s).

\section*{DFLTD(P)}
(1) Converts 32-bit BIN data specified for ©s to a 64-bit floating-point real number, and stores the result to (d).

(2) BIN values between -2147483648 and 2147483647 can be specified for ©s, © +1 .

\section*{O Operation Error}

No operation error occurs in the execution of the FLTD(P) and DFLTD \((P)\) instructions.

\section*{\(\square\) Program Example}
(1) In the following program, the 16 -bit BIN value in Var_D20 is converted to the 64-bit floatingpoint real number, and the result is stored to Var_D0.
[Structured ladder/FBD]

[ST]
FLTDP(SM400,Var_D20,Var_D0);
[Operation]

(2) In the following program, the 32-bit BIN value in Var_D20 is converted to the 64 -bit floatingpoint real number, and the result is stored to Var_D0.
[Structured ladder/FBD]

[ST]
DFLTDP(SM400,Var_D20L,Var_D0);
[Operation]

6.3.5 Floating-point data to 16-/32-bit BIN data conversion (single precision)

INT, DINT
- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later

\section*{INT(P)}

DINT(P)
P: Executing condition



\section*{Function}

\section*{INT(P)}
(1) Converts the 32-bit floating-point real number specified for ©s to 16-bit BIN data and stores the result to the device specified for (d).

(2) 32-bit floating-point real numbers between -32768 and 32767 can be specified for (5).
(3) The integer value stored in © is stored as 16 -bit BIN values.
(4) After conversion, the first digit from the decimal point of real number is rounded off.
(5) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{DINT(P)}
(1) Converts the 32-bit floating-point real number specified for ©s to 32-bit BIN data, and stores the result to the device specified for (d).

(2) 32-bit floating-point real numbers between -2147483648 and 2147483647 can be specified for (s).
(3) The integer value stored in (d) is stored as BIN 32 bits.
(4) After conversion, the first digit from the decimal point of real number is rounded off.
(5) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the specified device is outside the following range. (For Universal model QCPU and LCPU)
\(0,2^{-126} \leqq \mid\) value of the specified device \(\mid<2^{128}\)
(Error code: 4140)
- The value of the specified device is -0 , unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)
- The 32-bit floating-point data specified for (s) is outside the range of -32768 to 32767 for the INT instruction.
(Error code: 4100)
- The 32-bit floating-point data specified for ©s is outside the range of -2147483648 to 2147483647 for the DINT instruction.
(Error code: 4100)

\section*{\(\square\) Program Example}
(1) In the following program, the 32-bit floating-point real number in Var_D20 is converted to the 16 -bit BIN value, and the result is stored to Var_D0.
[Structured ladder/FBD]

[ST]
INTP(SM400,Var_D20,Var_D0);
[Operation]

(2) In the following program, the 32-bit floating-point real number in Var_D20 is converted to the 32-bit BIN value, and the result is stored to Var_D0.
[Structured ladder/FBD]

[ST]
DINTP(SM400,Var_D20,Var_D0);
[Operation]


\subsection*{6.3.6 Floating-point data to 16 -bit/32-bit BIN data conversion (double precision)}

INTD, DINTD


INTD(P) DINTD (P)


Input argument,
\begin{tabular}{|l|l|}
\hline & indicates any of the following \\
instructions. & \\
INTD & INTDP \\
DINTD & \\
& \\
\\
& \\
\hline
\end{tabular}
: Bit
: Double-precision real

Bit
: ANY16/32
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J:.1.:3} & \multirow[t]{2}{*}{} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{3}{|c|}{-} & - & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{3}{|c|}{-} & \(\bigcirc\) & - & - \\
\hline
\end{tabular}

\section*{Function}

\section*{INTD(P)}
(1) Converts the 64-bit floating-point real number specified for ©s to 16-bit BIN data and stores the result to the device specified for (d).


64-bit floating-point real number
(2) 64-bit floating-point real numbers between -32768.0 and 32767.0 can be specified for \(\left(s^{+}\right.\) 3 , ⓢ +2 , (s) +1 or (s).
(3) The integer value stored in (d) is stored as 16 -bit BIN values.
(4) After conversion, the first digit from the decimal point of the 64-bit floating-point real number is rounded off.
(5) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{DINTD(P)}
(1) Converts the 64-bit floating-point real number specified for ©s to 32-bit BIN data, and stores the result to (d).

(2) 64-bit floating-point real numbers between -2147483648.0 and 2147483647.0 can be specified for (s) +3 , (s) +2 , (s) +1 or (s).
(3) The integer value stored in (d) +1 and (d) is stored as BIN 32 bits.
(4) After conversion, the first digit from the decimal point of the 64-bit floating-point real number is rounded off.
(5) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{0 Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the specified device is not within the following range. (Error code: 4140) \(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 , unnormalized number, nonnumeric, or \(\pm \infty\) (For Universal model QCPU and LCPU)
(Error code: 4140)
- The 64-bit floating-point data set for (s) is outside the range of -32768.0 to 32767.0 for the INTD instruction.
(Error code: 4140)
- The 64-bit floating-point data set for ©s is outside the range of -2147483648.0 to 2147483647.0 for the DINTD instruction.
(Error code: 4140)

\section*{\(\square\) Program Example}
(1) In the following program, the 64-bit floating-point real number in Var_D20 is converted to the 16-bit BIN value, and the result is stored to Var_D0.
[Structured ladder/FBD]

[ST]
INTDP(SM400,Var_D20,Var_D0);
[Operation]

(2) In the following program, the 64-bit floating-point real number in Var_D20 is converted to the 32-bit BIN value, and the result is stored to Var_D0.
[Structured ladder/FBD]

[ST]
DINTDP(SM400,Var_D20,Var_D0);
[Operation]


\subsection*{6.3.7 16-bit BIN data to 32-bit BIN data conversion}

DBL


Function

Converts 16-bit BIN data in the device specified for (s) to 32-bit BIN data with a sign, and stores the result to the device specified for (d).


\section*{O Operation Error}

No operation error occurs in the execution of the DBL(P) instruction.

\section*{/Program Example}

In the following program, the 16 -bit BIN value in Var_D100 is converted to the 32 -bit BIN value when X20 turns ON, and the result is stored to Var_R100.
[Structured ladder/FBD]

[ST]
DBLP(X20,Var_D100,Var_R100);
[Operation]
\(\xrightarrow[\substack{\text { VB2Eн } \\(-1234)}]{\substack{\text { Var_D100 }}} \stackrel{\text { Var_R100 }}{\substack{\text { FFFFFB2Eн } \\(-1234)}}\)

\subsection*{6.3.8 32-bit BIN data to 16 -bit BIN data conversion}


Function
Converts 32-bit BIN data in the device specified for (s) to 16 -bit BIN data with a sign, and stores the result to the device specified for (d).

Devices can be specified in the range from -32768 to 32767 .


\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the device specified for (s) is outside the range of -32768 to 32767.
(Error code: 4100)

\section*{/Program Example}

In the following program, the 32 -bit BIN value in Var_R100 is converted to the 16 -bit BIN value when X20 turns ON, and the result is stored to Var_D100.
[Structured ladder/FBD]

[ST]
WORDP(X20,Var_R100,Var_D100);
[Operation]


\subsection*{6.3.9 16-/32-bit BIN data to Gray code conversion}


Function

\section*{GRY(P)}

Converts 16-bit BIN data in the device specified for ©s to Gray code, and stores the result to the device specified for (d).

Negative values cannot be specified for (s).


\section*{DGRY(P)}

Converts 32-bit BIN data in the device specified for (s) to Gray code, and stores the result to the device specified for (a).

Negative values cannot be specified for (s).


\section*{Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of (s) is a negative number.

\section*{\(\triangle\) Program Example}
(1) In the following program, the BIN value in Var_D100 is converted to Gray code when X10 turns ON, and the result is stored to Var_D200.
[Structured ladder/FBD]

[ST]
GRYP(X10,Var_D100,Var_D200);
(2) In the following program, the BIN value in Var_D10 is converted to Gray code when X1C turns ON, and the result is stored to Var_D100.
[Structured ladder/FBD]

[ST]
DGRYP(X1C,Var_D10,Var_D100);

\subsection*{6.3.10 Gray code to 16 -/32-bit BIN data conversion}

GBIN, DGBIN

Basic \(\begin{gathered}\text { High } \\ \text { pertomance }\end{gathered}\)
Process
Redundan
Universal
LCPU


\section*{IT Function}

\section*{GBIN(P)}

Converts Gray code data in the device specified for (s) to 16-bit BIN data and stores the result to the device specified for (d).
(S) Gray code 1234
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multicolumn{17}{|c|}{16 bit} \\
\hline \multicolumn{17}{|l|}{} \\
\hline 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & & & 0 & 1 & \\
\hline
\end{tabular}
(d) BIN
\[
1234 \begin{array}{|l|l|l|l|l|l|l|l|l|l|l|l|l|l|l|l|}
\hline \mathrm{b} 15 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
\hline
\end{array}
\]

\section*{DGBIN(P)}

Converts Gray code data in the device specified for ©s to 32-bit BIN data and stores the result to the device specified for (d).


\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of © s is outside the range of 0 to 32767 for the \(\operatorname{GBIN}(P)\) instruction.
(Error code: 4100)
- The value of ©s is outside the range of 0 to 2147483647 for the \(\operatorname{DGBIN}(P)\) instruction.
(Error code: 4100)

\section*{\(\triangle\) Program Example}
(1) In the following program, the Gray code in Var_D100 is converted to the BIN value when X10 turns ON, and the result is stored to Var_D200.
[Structured ladder/FBD]

[ST]
GBINP(X10,Var_D100,Var_D200);
(2) In the following program, the Gray code in Var_D10 is converted to the BIN value when X1C turns ON, and the result is stored to Var_D0.
[Structured ladder/FBD]

[ST]
DGBINP(X1C,Var_D10,Var_D0);

\subsection*{6.3.11 Two's complement of 16-/32-bit BIN data (sign inversion)}

NEG, DNEG

\section*{Basic \\ High
performanc \\ Process \\ Redundant \\ Universal \\ LCPU}

NEG(P)
DNEG(P)



Input argument, Output argument,

EN: Executing condition
ENO: Execution result
d: \(\quad\) Start number of the device that stores data to be converted by : ANY16/32 the two's complement


\section*{Function}

\section*{NEG(P)}
(1) Inverts the sign of the 16 -bit device specified for (d) and stores the result to the device specified for (d).

(2) This instruction is used when inverting positive and negative signs.

\section*{DNEG(P)}
(1) Inverts the sign of the 32-bit device specified for (d) and stores the result to the device specified for (d).

(2) This instruction is used when inverting positive and negative signs.

\section*{Operation Error}

No operation error occurs in the execution of the \(N E G(P)\) and \(\operatorname{DNEG}(P)\) instructions.

\section*{\(\triangle\) Program Example}

In the following program, the value in g_int2 is subtracted from the value in g_int1 when g_bool1 turns ON, and if the result is negative, the absolute value is defined.
[Structured ladder/FBD]

[ST]
IF g_bool1 THEN
OUT(g_int1<g_int2,g_bool2);
g_int1:=g_int1-g_int2;
NEGP(g_bool2,g_int1);
END_IF;

\subsection*{6.3.12 Sign inversion of floating-point data (single precision)}
- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later

\section*{ENEG(P)}



Input argument, EN:
Output argument,

EN: Executing condition : Bit
ENO: Execution result : Bit
d: Floating-point data whose sign to be inverted : Single-precision real
Start number of the device that stores 32-bit floating-point data
whose sign to be inverted
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & Inte & vice & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...ala} & \multirow{2}{*}{U} & \multirow{2}{*}{Zn} & \multirow{2}{*}{Constant} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{3}{|c|}{-} \\
\hline
\end{tabular}

Function
(1) Inverts the sign of the 32-bit floating-point real number data specified for © , and stores the result to the device specified for (d).
(2) This instruction is used when inverting positive and negative signs.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value of the specified device is outside the following range.
(For Universal model QCPU and LCPU)
\(0,2^{-126} \leqq \mid\) value of the specified device \(\mid<2^{128}\)
(Error code: 4140)
- The value of the specified device is -0 , unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)

\section*{\(\square\) Program Example}

In the following program, the sign of the 32-bit floating-point real number in Var_D100 is inverted when X20 turns ON, and the result is stored to Var_D100.
[Structured ladder/FBD]


\section*{[ST]}

ENEGP(X20,Var_D100);

\section*{[Operation]}


\subsection*{6.3.13 Sign inversion of floating-point data (double precision)}

EDNEG


EDNEG(P)



Input argument,
Output argument,

EN:
ENO: Execution result
\(\mathrm{d}: \quad\) Start number of the device that stores 64-bit floating-point data : Double-precision real
whose sign to be inverted
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...ll} & \multirow[b]{2}{*}{U} & \multirow{2}{*}{Zn} & \multirow{2}{*}{Constant} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{6}{|c|}{-} \\
\hline
\end{tabular}

\section*{Function}
(1) Inverts the sign of the 64-bit floating-point real number specified for (d), and stores the result to the device specified for (d).
(2) This instruction is used when inverting positive and negative signs.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the specified device is not within the following range. (Error code: 4140) \(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 .
(Error code: 4140)

\section*{\(\square\) Program Example}

In the following program, the sign of the 64-bit floating-point real number in Var_D0 is inverted when X20 turns ON, and the result is stored to Var_D0.
[Structured ladder/FBD]

[ST]
ENEGP(X20,Var_D0);
[Operation]


\subsection*{6.3.14 16-bit BIN block data to 4-digit BCD block data conversion BKBCD}



Input argument,

Output argument,
\begin{tabular}{ll} 
EN: & Executing condition \\
\(\mathrm{s}:\) & Start number of the device that stores BIN data \\
\(\mathrm{n}:\) & Number of converted data \\
ENO: & Execution result \\
\(\mathrm{d}:\) & Start number of the device that stores converted BIN data
\end{tabular}

: Bit
: ANY16
: ANY16
: Bit
: ANY16
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...|ilil} & \multirow[b]{2}{*}{U:19:3} & \multirow[b]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{-} & - \\
\hline n & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{\(\bigcirc\)} & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{-} & - \\
\hline
\end{tabular}

\section*{\(\Sigma\) Function}
(1) Converts the BIN data (0 to 9999) n points from the device specified for ©s to the BCD data, and stores the results to the device specified for (d) and the following devices.

Must always be "0".


BCD conversion
(d) BCD 1234010101101011001011010

(d) +2 BCD \(15450: 00110: 1: 1: 0: 1: 000: 1010\)
(d) \(+(\mathrm{n}-2)\) BCD \(43210: 1: 000001: 10011000010\)
(d) \(+(\mathrm{n}-1)\) BCD \(555500: 101001: 0,1010,1 \mid 01010\)

\section*{OOperation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The range of \(n\) points from the device specified for (s), (d) exceeds the corresponding device.
(Error code: 4101)
- The data n points from the device specified for (s) is outside the range of 0 to 9999.
(Error code: 4100)
- The device ranges of (s) and (d) overlap.
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the BIN values in D100 to D102 are converted to the BCD values when X20 turns ON, and the results are stored to D200 and the following devices.
[Structured ladder/FBD]


\section*{[ST]}

Var_D0:=3;
BKBCDP(X20,D100,Var_D0,D200);
[Operation]
\begin{tabular}{|c|c|c|c|c|}
\hline & \multicolumn{4}{|l|}{\begin{tabular}{l}
NOOy \\
 \\

\end{tabular}} \\
\hline D100 BIN 5432 & OOO11 & 0110 & 001 & 1000 \\
\hline D101 BIN 4444 & O00 01 & 000 & 0110 & 1100 \\
\hline D102 BIN 3210 & 0000 & 1100 & 1000 & 1010 \\
\hline
\end{tabular}

BCD Var_D0 3

D200 BCD 543201101010000110010
D201 BCD \(44440110001: 00010001100\)
D202 BCD 3210001110010000110000

\subsection*{6.3.15 4-digit BCD block data to 16 -bit BIN block data conversion}

\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J)10,} & \multirow[t]{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{\begin{tabular}{l}
Constant \\
K, H
\end{tabular}} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & & - \\
\hline n & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & & - \\
\hline
\end{tabular}

\section*{F Function}
(1) Converts the BCD data (0 to 9999) n points from the device specified for © to the BIN data, and stores the results to the device specified for (d) and the following devices.

BIN conversion
NOMホNNom

(s) \(+1 \quad\) BIN \(5678 \quad 00010: 110001001110\)




\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The range of \(n\) points from the device specified for (s), (d) exceeds the range of the corresponding device.
(Error code: 4101)
- The data \(n\) points from the device specified for (s) is outside the range of 0 to 9999.
(Error code: 4100)
- The device ranges of (s) and (d) overlap.
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program，the BCD values in D100 to D102 are converted to the BIN values when X20 turns ON，and the results are stored to D200 and the following devices．
［Structured ladder／FBD］


\section*{［ST］}

Var＿D0：＝3；
BKBINP（X20，D100，Var＿D0，D200）；

\section*{［Operation］}
\％28882

 D101 BCD 76540 0，1： \(1: 1\) 1： \(1: 1: 00: 1: 0: 10110: 0\) D102 BCD 9999 100， 11110011100111001

BIN conversion （when Var＿D0＝3）

\section*{NOOサオNツ}


D201 BIN 765400001111001111110001110
D202 BIN \(99990001,0011,1,1000001,1,1,1\)

\subsection*{6.3.16 Single-precision to double-precision conversion}

\section*{ECON(P)}


\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J......} & \multirow[b]{2}{*}{Unic.an} & \multirow[b]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{3 Function}
(1) Converts 32-bit floating-point real number specified for (s) to 64-bit floating-point real number, and stores the conversion result to the device specified for (d).

(2) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the specified device is not within the following range. (Error code: 4140)
\(0,2^{-126} \leqq \mid\) value of the specified device \(\mid<2^{128}\)
- The value of the specified device is -0 , unnormalized number, nonnumeric or \(\pm \infty\).
(Error code: 4140)

\section*{\(\square\) Program Example}

In the following program, the 32-bit floating-point real number in Var_D10 is converted to 64-bit floating-point real number when X0 turns ON, and the result is output to Var_D0.
[Structured ladder/FBD]

[ST]
ECON(X0,Var_D10,Var_D0);

\subsection*{6.3.17 Double-precision to single-precision conversion}

:Bit
:Double-precision real
:Bit
:Single-precision real
\(\mathrm{d}: \quad\) Start number of the device that stores conversion data
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{)..aly} & \multirow[b]{2}{*}{Uninala} & \multirow[b]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{3}{|c|}{-} & - & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{3}{|c|}{-} & \(\bigcirc\) & - & \\
\hline
\end{tabular}

\section*{Function}
(1) Converts 64-bit floating-point real number specified for ©s to 32-bit floating-point real number, and stores the conversion result to the device specified for (d).

(2) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value of the specified device is not within the following range. (Error code: 4140) \(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 .
(Error code: 4140)
- The conversion result exceeds the following range (an overflow occurs). \(2^{128} \leqq \mid\) conversion result |
(Error code: 4141)

\section*{\(\triangle\) Program Example}

In the following program, the 64-bit floating-point real number in Var_D10 is converted to 32-bit floating-point real number when X0 turns ON, and outputs the conversion result to Var_D0.
[Structured ladder/FBD]

[ST]
EDCON(X0,Var_D10,Var_D0);

\subsection*{6.4 Data Transfer Instructions}

\subsection*{6.4.1 16-/32-bit data transfer}

MOV, DMOV

MOV(P)
DMOV(P)
P: Executing condition
\(\pm\)


instructions
MOV
DMOV

: Bit
: ANY16/32
: Bit
: ANY16/32


\section*{®POINT}

When BL, S, TR, BLIS, or BLITR is used, refer to SFC control instructions of the MELSEC-Q/L/QnA Programming Manual (SFC).

\section*{\(\mathcal{Y}\) Function}

\section*{MOV(P)}

Transfers the 16 -bit data in the device specified for © \({ }^{\text {s }}\) to the device specified for © .


Transfers the 32-bit data in the device specified for (s) to the device specified for © .
Before transfer
(s)

After transfer
(d)
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multicolumn{17}{|l|}{} \\
\hline 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & & 1 & 0 & 0 & 1 & 0 \\
\hline
\end{tabular}

\section*{O Operation Error}

No operation error occurs in the execution of the \(\operatorname{MOV}(\mathrm{P})\) and \(\mathrm{DMOV}(\mathrm{P})\) instruction.

\section*{\(\triangle\) Program Example}
(1) In the following program, the input data in the devices from \(X 0\) to \(X B\) are transferred to Var_D8.
[Structured ladder/FBD]

[ST]
MOVP(SM400,K3X0,Var_D8);
(2) In the following program, 155 is transferred to Var_D8 when X8 turns ON. [Structured ladder/FBD]

[ST]
MOVP(X8,155,Var_D8);
(3) In the following program, the data in Var_D0 are transferred to Var_D7. [Structured ladder/FBD]


\section*{[ST]}

DMOVP(SM400,Var_D0,Var_D7);
(4) In the following program, the data in the devices from X0 to X1F are transferred to Var_D0. [Structured ladder/FBD]


\section*{[ST]}

DMOVP(SM400,K8X0,Var_D0);

\subsection*{6.4.2 Floating-point data transfer (single precision)}
- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later

EMOV(P)
P: Executing condition


Input argument,
EN: Executing condition
\(\mathrm{s}: \quad\) Data to be transferred, or start number of the device that stores data to be transferred
Output argument,

ENO: Execution result
: Bit
d: Start number of the device that stores transfer destination of : Single-precision real data
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..1):} & \multirow[b]{2}{*}{U} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & & - & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & ) & - & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Transfers the 32-bit floating-point real number data in the device specified for ©s to the device specified for (d).

(2) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

No operation error occurs in the execution of the EMOV(P) instruction.

\section*{\(\triangle\) Program Example}
(1) In the following program, the real number in Var_D10 is transferred to Var_D0.
[Structured ladder/FBD]

[ST]
EMOVP(SM400,Var_D10,Var_D0);
[Operation]
\begin{tabular}{|c|c|} 
Var_D10 \\
\hline 36.475 \\
\hline
\end{tabular}
(2) In the following program, the real number -1.23 is transferred to Var_D10 when X8 turns ON.
[Structured ladder/FBD]

[ST]
EMOVP(X8,-1.23,Var_D10);
[Operation]
\(\square-1.23\).

\subsection*{6.4.3 Floating-point data transfer (double precision)}



Input argument,

Output argument,

EN: Executing condition
s: Data to be transferred, or start number of the device that stores data to be transferred
ENO: Execution result
\(\mathrm{d}: \quad\) Start number of the device that stores transfer destination of
\begin{tabular}{|ll|}
\hline EDMOV & EDMOVP \\
\\
\\
\\
\\
\\
\hline
\end{tabular}
: Bit
Double-precision real

Bit
: Double-precision real data
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J:.10:} & \multirow[b]{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{\(\sqrt{3}\) Function}
(1) Transfers the 64-bit floating-point real number specified for © \({ }^{\text {s }}\) to the device specified for © .

(2) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

No operation error occurs in the execution of the EDMOV(P) instruction.

\section*{\(\square\) Program Example}
(1) In the following program, the 64-bit floating-point real number in Var_D10 is transferred to Var_D0.
[Structured ladder/FBD]

[ST]
EDMOVP(SM400,Var_D10,Var_D0);

\section*{[Operation]}

(2) In the following program, the real number - 1.23 is transferred to Var_D10 when X8 turns ON .
[Structured ladder/FBD]

[ST]
EDMOVP(X8,-1.23,Var_D10);
[Operation]


\subsection*{6.4.4 Character string data transfer}

\begin{tabular}{|l|l|}
\hline & \\
\hline instructions. & indicates any of the following \\
SMOV & \\
\\
\\
\\
\\
\hline
\end{tabular}

Input argument,
EN: Executing condition
: Bit
s: \(\quad\) Character string data to be transferred (Maximum string
: String
length: 32 characters), or start number of the device that
stores character string data to be transferred
Output argument,
ENO: Execution result
: Bit
\(\mathrm{d}: \quad\) Start number of the device that stores character string data to : String
be transferred
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J等):} & \multirow[b]{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant \$} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Transfers the character string data specified for (s) to the device specified for (d) and the following devices.
Transfers the character string data with double quotation (") specified for (s) or the character string data from the specified device number to the device number that stores 00 H at once.

(2) Processing will be performed without error even in cases where the range for the devices storing the character string data to be transferred (s) to (s) \(+n\) ) overlaps with the range of the devices which will store the character string data after it has been transferred (© to © +n ). The following occurs when the character string data that had been stored in D10 to D13 is transferred to D11 to D14:
\begin{tabular}{|c|c|c|c|c|}
\hline \multicolumn{3}{|c|}{b15----b8b7---- b0} & 15---- b8b7---- & \multirow{6}{*}{Character string before transfer is remained.} \\
\hline D10 & 32н (2) \({ }^{\text {(2) }}\) (1) & D10 & 32н (2) 31 H (1) & \\
\hline D11 & 34н (4) 33 н (3) & D11 & 32н (2) 31н (1) & \\
\hline D12 &  & D12 & 34н (4) 33 H (3) & \\
\hline D13 & 00 H & D13 & 36 H (6) \({ }^{\text {a }}\) 35 H (5) & \\
\hline D14 & & -D14 & OOH & \\
\hline
\end{tabular}
(3) If the 00 H code is being stored to lower bytes of (s) \(+\mathrm{n}, 00 \mathrm{H}\) will be stored to both the upper bytes and the lower bytes of (d) \(+n\).



\section*{0 Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- There is no 00 H code stored between the device number specified for © 5 and the corresponding device.
(Error code: 4101)
- The entire character string linked from the device number specified for © to the final device number of the corresponding device cannot be stored.
(Error code: 4101)
- The character string of ©s exceeds 16383 characters.
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the character string data in Var_D10 are transferred to Var_D20 when X0 turns ON.
[Structured ladder/FBD]

[Operation]

(2) In the following program, the character string "ABCD" is transferred to Var_D20 when X0 turns ON.
[Structured ladder/FBD]


\subsection*{6.4.5 16-/32-bit data negation transfer}

CML, DCML

\begin{tabular}{|ll|}
\hline instructions. & indicates any of the following \\
CML & CMLP \\
DCML & DCMLP \\
& \\
\\
\\
\\
\\
\hline
\end{tabular}

Input argument,
Output argument,
Executing condition
: Bit
\(\mathrm{s}: \quad\) Data to be inverted, or start number of the device that stores
ANY16/32 data to be inverted
d:
: Bit
: ANY16/32
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{小..al} & \multirow[b]{2}{*}{U} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{\[
\begin{gathered}
\text { Constant } \\
\text { K, H }
\end{gathered}
\]} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & \multicolumn{7}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) & \multicolumn{7}{|c|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{Function}

\section*{CML(P)}

Inverts 16-bit data specified for ©s bit by bit, and transfers the result to the device specified for (d).


\section*{DCML(P)}

Inverts 32-bit data specified for (s) bit by bit, and transfers the result to the device specified for (d).

\section*{O Operation Error}

No operation error occurs in the execution of the \(\operatorname{CML}(P)\) and \(\operatorname{DCML}(P)\) instruction.

\section*{\(\triangle\) Program Example}
(1) In the following program, the data in the devices from \(X 0\) to \(X 7\) are inverted, and the results are transferred to Var_D0.
[Structured ladder/FBD]

[ST]
CML(SM402,K2X0,Var_D0);

\section*{[Operation]}

If the number of bits of (s) is smaller than the number of bits of (d)

(2) In the following program, the data in the devices from M16 to M23 are inverted, and the results are transferred to the devices from Y40 to Y47.
[Structured ladder/FBD]


\section*{[ST]}

CML(SM402,K2M16,K3Y40);

\section*{[Operation]}

If the number of bits of (s) is smaller than the number of bits of (d)

(3) In the following program, the data in Var_D0 are inverted when X3 turns ON, and the result is transferred to Var_D16.
[Structured ladder/FBD]

[ST]
CMLP(X3,Var_D0,Var_D16);
[Operation]

(4) In the following program, the data in the devices from X0 to X1F are inverted, and the results are transferred to Var_D0.
[Structured ladder/FBD]

[ST]
DCML(SM402,K8X0,Var_D0);

\section*{[Operation]}

If the number of bits of (s) is smaller than the number of bits of (d)

(5) In the following program, the data in the devices from M16 to M35 are inverted, and the results are transferred to the devices from Y40 to Y63.
[Structured ladder/FBD]

[ST]
DCML(SM402,K5M16,K6Y40);
[Operation]
If the number of bits of (s) is smaller than the number of bits of (d)

(6) In the following program, the data in Var_D0 are inverted when X3 turns ON, and the result is transferred to Var_D16.
[Structured ladder/FBD]

[ST]
DCMLP(X3,Var_D0,Var_D16);
[Operation]
b31------b24---- b8b7------b0 Var_D0 0:0,0:00:1:0:0 01111100011011100
b31-------b24-----b8b7-------b0
\begin{tabular}{|c|c|c|c|}
\hline & Var_D16 & 111111110111 & 10000 11101100111 \\
\hline
\end{tabular}

\subsection*{6.4.6 16-bit block data transfer}

\begin{tabular}{|l|l|}
\hline BMOV & \\
instructions. & BMOVP \\
\\
\\
\\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{3}{*}{Input argument,} & EN: & \multicolumn{5}{|l|}{Executing condition} & \multicolumn{4}{|l|}{} \\
\hline & s : & \multicolumn{5}{|l|}{Start number of the device that stores data to be transferred} & \multicolumn{4}{|l|}{: ANY_SIMPLE} \\
\hline & n : & \multicolumn{5}{|l|}{Number of transfers} & \multicolumn{4}{|l|}{: ANY16} \\
\hline \multirow[t]{7}{*}{Output argument,} & ENO: & \multicolumn{5}{|l|}{Execution result} & \multicolumn{4}{|l|}{: Bit} \\
\hline & d: & \multicolumn{5}{|l|}{Start number of the device at the transfer destination} & \multicolumn{4}{|l|}{: ANY_SIMPLE} \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...ala} & \multirow[t]{2}{*}{U:...icala} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (s) & \multicolumn{6}{|c|}{\(\bigcirc\)} & \multicolumn{2}{|c|}{-} & - \\
\hline & n & \multicolumn{6}{|c|}{\(\bigcirc\)} & \multicolumn{2}{|c|}{\(\bigcirc\)} & - \\
\hline & (d) & \multicolumn{6}{|c|}{\(\bigcirc\)} & \multicolumn{2}{|c|}{-} & - \\
\hline
\end{tabular}

\section*{®POINT}

When BL, S, TR, BLIS, or BLITR is used, refer to SFC control instructions of the MELSEC-Q/L/QnA Programming Manual (SFC).

\section*{\(\sum\) Function}
(1) Batch transfers the 16-bit data \(n\) points from the device specified for (s) to the \(n\) points of devices from the one specified for (d).

(2) Transfers can be accomplished even in cases where there is an overlap between the source and destination device.
In the case of transmission to the smaller device number, transmission is from (s); for transmission to the larger device number, transmission is from © \(+(n-1)\).
However, as shown in the example below, when transferring data from \(R\) to \(Z R\), or from ZR to \(R\), the range to be transferred and the range of destination must not overlap.
Transfer from R to R , or from ZR to ZR can be performed without any problem.
- ZR transfer range ((specified start number of \(Z R\) ) to (specified start number of \(Z R+\) the number of transfers -1))
- R transfer range ((specified start number of \(R+\) file register block number \(\times 32768\) ) to (specified start number of \(R+\) file register block number \(\times 32768+\) the number of transfers -1))

\section*{Example}

Transfer ranges of ZR and R overlap when transferring 10000 blocks of data from ZR30000 (source) to R10 (block No. 1 of the destination).
- ZR transfer range \(\rightarrow\) (30000) to \((30000+10000-1) \rightarrow(30000)\) to (39999)
- \(R\) transfer range \(\rightarrow(10+(1 \times 32768))\) to \((10+(1 \times 32768)+10000-1)\)
\(\rightarrow\) (32778) to (42777)
Therefore, the range 32778 to 39999 overlaps and the data are not correctly transferred.

(3) When © is a word device and (d) is a bit device, the target for the word device will be the number of bits specified for digit-specified bit device.
If K1Y30 has been specified for © \({ }^{(d)}\), the lower four bits of the word device specified for © (s) will become the object.

(4) If bit device has been specified for (s) and ©(d), then ©s and (d) should always have the same number of digits.
(5) When using a link direct device and an intelligent function module device for (s) and (d), only either of (s) or (d) can be used.
(6) Selecting the execution of the device range check

Whether to disable or enable the device range check at the execution of the BMOV instruction can be selected by the device range check disable flag (SM237). (Only when the subset condition is satisfied)
The device range check for the devices (s) to (s) \(+(n-1)\) and (d) to (d) \(+(n-1)\) is not performed when SM237 is ON.

\section*{Caution}

Do not perform the following accesses when SM 237 is ON.
- An access in which the index setting exceeds the device range.
- An access in which devices from © to © \(+(n-1)\) cross over the boundary of the device range. \({ }^{* 1}\)
- An access to file registers without setting file registers.
- An access to the area which does not contain multiple CPU high speed transmission area devices. (For QCPU (Q mode) only)
*1: Refer to the DFMOV instruction.

\section*{区POINT}

SM237 can be used for the following CPU modules only.
- Universal model QCPU with a serial number whose first five digits are '10012' or higher
- LCPU

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device range of \(n\) points from (s) or (d) exceeds the corresponding device range.
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the lower 4-bit data in the devices from D66 to D69 are transferred to the devices from Y30 to Y3F in units of 4 points.
[Structured ladder/FBD]


\section*{[ST]}

BMOVP(SM402,D66,4,K1Y30);
[Operation]
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multicolumn{5}{|r|}{Before execution (source of transfer)
b15- - - b4b3- - b0} & \multicolumn{5}{|l|}{After execution (destination of transfer)} \\
\hline D66 & 1 & 1 & 10 & & 1 & 1 & 0 & 1 & Y33 to Y30 \\
\hline D67 & 0 & 0 & 00 & & 0 & 0 & 0 & 0 & Y37 to Y34 \\
\hline D68 & 1 & 0 & 01 & 1 & 0 & 0 & 1 & 1 & Y3B to Y38 \\
\hline D69 & 0 & 1 & 10 & & 1 & 1 & 0 & 1 & Y3F to Y3C \\
\hline \multicolumn{5}{|c|}{Ignored} & & & & & \\
\hline
\end{tabular}
(2) In the following program, the data in the devices from X20 to X2F are transferred to the devices from D100 to D103 in units of 4 points.
[Structured ladder/FBD]


\section*{[ST]}

BMOVP(SM402,K1X20,4,D100);

\section*{[Operation]}


\subsection*{6.4.7 Identical 16-bit block data transfer}


\begin{tabular}{llll} 
Input argument, & EN: & Executing condition \\
\(\mathrm{s}:\) & \begin{tabular}{l} 
Data to be transferred, or start number of the device that \\
stores data to be transferred
\end{tabular} & \(:\) ANY16 \\
Output argument, & \(\mathrm{n}:\) & \begin{tabular}{l} 
Number of transfers
\end{tabular} \\
& \(\mathrm{ENO}:\) & Execution result \\
\(\mathrm{d}:\) & Start number of the device at the transfer destination & : ANY16 \\
: ANY
\end{tabular}


\section*{Function}
(1) Transfers the 16-bit data \(n\) points from the device specified for ©s to the \(n\) points of devices from the one specified for (d).

(2) In cases where (s) specifies a word device and (d) a bit device, the number of bits specified for digit-specified bit device will be the target bits for the word device.
If K1Y30 is specified for (d), the lower 4 bits of the word device specified for (s) becomes a target.

(3) If bit device is specified for (s) and (d), then (s) and (d) should always have the same number of digits.
(4) Selecting the execution of the device range check

Whether to disable or enable the device range check at the execution of the FMOV instruction can be selected by the device range check disable flag (SM237). (Only when the subset condition is satisfied)
The device range check for the devices (d) to (d) \(+(n-1)\) is not performed when SM237 is ON.

\section*{Caution}

Do not perform the following accesses when SM 237 is ON.
- An access in which the index setting exceeds the device range.
- An access in which devices from (d) to © \(+(n-1)\) cross over the boundary of the device range. \({ }^{*}\)
- An access to file registers without setting file registers.
- An access to the area which does not contain multiple CPU high speed transmission area devices. (For QCPU (Q mode) only)
*1: Refer to the DFMOV instruction.

\section*{®POINT}

SM237 can be used for the following CPU modules only.
- Universal model QCPU with a serial number whose first five digits are '10012' or higher.
- LCPU

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device range of \(n\) points from (d) exceeds the corresponding device range.
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the lower 4-bit data in D0 are transferred to the devices from Y10 to Y23 in units of 4 bits when X0A turns ON.
[Structured ladder/FBD]

[ST]
FMOVP(X0A,D0,5,K1Y10);
[Operation]

(2) In the following program, the data in the devices from X20 to X 23 are transferred to the devices from D100 to D103 when X0A turns ON.
[Structured ladder/FBD]

[ST]
FMOVP(X0A,K1X20,4,D100);
[Operation]

Before execution \(1: 0111 / 0,1: 1: 1\)
Ignored


Filled with 0s.

\subsection*{6.4.8 Identical 32-bit block data transfer}

- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported


Function
(1) Transfers the 32-bit data \(n\) points from the device specified for ©s to the \(n\) points of devices from the one specified for (d).
(S) +1 ,
(S) \({ }^{\text {b31 }} 1234567 \mathrm{H}\)
\(\stackrel{\text { Transfer }}{\square}\)
\(\begin{array}{ll}\text { (d) }+1 \text {, } & \text { (d) } \\ \text { (d) }+3, & \text { (d) }+2 \\ \text { (d) }+5, & \text { (d) }+4\end{array}\)
(d) \(+(2 n-1)\), (d
d \(+(2 n-2)\)
\begin{tabular}{l} 
b31 \\
\hline 1234567 H \\
\hline 1234567 H \\
\hline 1234567 H \\
\hline 12 \\
\hline 1234567 H \\
\hline
\end{tabular}
(2) When digits are specified for (s, only the data of digit specification are transferred. If K5Y0 is specified for ©s, the lower 20 bits ( 5 digits) of the word device specified for (s) become the targets.

(3) When digits are specified for (d), the data of digit specification for (d) are transferred. If K5Y0 is specified for © \({ }^{(d)}\), the lower 20 bits of the word device specified for ©s become the targets.
If digits are specified for both (s) and (a), the data of digit specification for (d) are transferred regardless of the number of specified digits.

(4) No processing is performed if the value specified for n is 0 .
(5) Whether to disable or enable the device range check at the execution of the DFMOV instruction can be selected by the device range check disable flag (SM237). (Only when the subset condition is satisfied)

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- A negative value is specified for \(n\).
(Error code: 4100)
- n points of data to be transferred exceed the device range of (d).
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the data in the devices from Y0 to X 13 (20 bits) are stored to the devices from D10 to D17 when M0 turns ON.
[Structured ladder/FBD]


\section*{[ST]}

DFMOV(M0,K5Y0,K4,D10);
[Operation]



\subsection*{6.4.9 16-/32-bit data exchange}

XCH, DXCH

\section*{Basic \\ Hishomance \\ Process \\ Redundant Universal \\ LCPU}

\begin{tabular}{|ll|}
\hline instructions. & XCHP \\
XCH & DXCHP \\
DXCH & \\
\\
& \\
\\
& \\
\hline
\end{tabular}
\begin{tabular}{llll} 
Input argument, & EN: & Executing condition & \(:\) Bit \\
Output argument, & ENO: & Execution result & \(:\) Bit \\
& \(\mathrm{d} 1, \mathrm{~d} 2:\) & Start number of the device that stores data to be exchanged & \(:\) ANY16/32
\end{tabular}


\section*{Function}

\section*{XCH(P)}

Exchanges the 16-bit data between (d1) and (d2).
\begin{tabular}{|c|c|}
\hline & (d1) \\
\hline & b15-----b8b7----- - b0 \\
\hline Before execution & 0:1111100000000000011111 \\
\hline
\end{tabular}


\section*{DXCH(P)}

Exchanges the 32-bit data between (d1) and (d2).


\section*{O Operation Error}

No operation error occurs in the execution of the \(\mathrm{XCH}(\mathrm{P})\) and \(\mathrm{DXCH}(\mathrm{P})\) instruction.

\section*{Program Example}
(1) In the following program, the current value of T0 and the value of Var_D0 are exchanged when X8 turns ON.
[Structured ladder/FBD]

[ST]
XCHP(X8,TN0,Var_D0);
(2) In the following program, the data in Var_D0 and the data in the devices from M16 to M31 are exchanged when X10 turns ON.
[Structured ladder/FBD]

[ST]
XCHP(X10,Var_D0,K4M16);
(3) In the following program, the data in Var_D0 and the data in the devices from M16 to M47 are exchanged when X10 turns ON.
[Structured ladder/FBD]


\section*{[ST]}

DXCHP(X10,Var_D0,K8M16);
(4) In the following program, the data in Var_D0 and the data in Var_D9 are exchanged when MO turns ON.
[Structured ladder/FBD]

[ST]
DXCHP(M0,Var_D0,Var_D9);

\subsection*{6.4.10 16-bit block data exchange}

\begin{tabular}{|l|l|}
\hline BXCH \\
instructions. \\
\\
\\
\\
\\
\\
\\
\end{tabular}

Input argument,
EN: Executing condition
: Bit
n : \(\quad\) Number of exchanges
: ANY16
Output argument,
ENO: Execution result
Bit
d1, d2: Start number of the device that stores data to be exchanged : ANY16
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{Jand} & \multirow[b]{2}{*}{U} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{\[
\begin{gathered}
\text { Constant } \\
\text { K, H }
\end{gathered}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline n & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{\(\bigcirc\)} & & - \\
\hline (d1) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & & - \\
\hline (d2) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & & - \\
\hline
\end{tabular}

\section*{\(\Sigma\) Function}

Exchanges the 16 -bit data \(n\) points from the device specified for (d1) and the 16 -bit data n points from the device specified for (d2).


\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device range of \(n\) points from the device specified for (d1), (d2) exceeds the corresponding device.
(Error code: 4101)
- The (d11) and (d2) devices overlap.
(Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, the data 3 points from D200 and the data 3 points from R0 are exchanged when X1C turns ON.
[Structured ladder/FBD]

[ST]
BXCHP(X1C,3,D200,R0);
[Operation]

D200 \(\begin{aligned} & \text { b/ } \\ & 01 \\ & 0\end{aligned}\)

D202
b15------b8b7-------b0

R1 \(0: 0111000: 1,10: 011100011\) R2 111,0,000000 1, 1,0,000000


D200 \(0 \cdot 1 \cdot 1: 1\)
D201 0:0, 1, 10:0, 1, 10:0, 1, 1 \(0,0,1,1\)
D202 1110:0000001110000000
b15------b8b7-------b0


R2 1001:0 10: \(1: 011001: 0110110\)

\subsection*{6.4.11 Upper and lower bytes exchange}

\section*{SWAP \\  \\ Hich \\ eatiomanee Process \\ Redundant Universal \\ LCPU}

\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{} \\
\hline instruct & \\
\hline SWAP & SWAPP \\
\hline
\end{tabular}
\begin{tabular}{llll} 
Input argument, & EN: & Executing condition & \(:\) Bit \\
Output argument, & \(\mathrm{S}:\) & SNO: & Execution result
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...1:} & \multirow{2}{*}{U"..'IGM:} & \multirow{2}{*}{Zn} & \multirow{2}{*}{Constant} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & \multicolumn{7}{|c|}{\(\bigcirc\)} & \multicolumn{2}{|c|}{-} \\
\hline
\end{tabular}

\section*{Function}

Exchanges the higher 8 bits and lower 8 bits of the device specified for \((s)\).


\section*{\(\square\) Program Example}

In the following program, the higher 8 bits and the lower 8 bits in R10 are exchanged when X10 turns ON.
[Structured ladder/FBD]

[ST]
SWAPP(X10,Var_R10);

\section*{[Operation]}


\subsection*{6.5 Program Branch Instructions}

\subsection*{6.5.1 Pointer branch}

\author{
CJ, SCJ, JMP
}

Process
Redundan
Universal
LCPU

CJ
SCJ
JMP


\section*{CJ}
(1) Executes the program of the specified pointer number or ladder block label when the execution command is ON.
(2) When the execution command is OFF, the program at the next step is executed.


\section*{SCJ}
(1) Executes the program of the specified pointer number or ladder block label from the next scan after the execution command turns from OFF to ON.
(2) Executes the program in the next step when the execution command is OFF or turns from ON to OFF.


\section*{JMP}
(1) Unconditionally executes the program of the specified pointer number or ladder block label.

Note the following points when using the CJ, SCJ, and JMP instructions.
1. After the timer coil has turned ON, accurate measurements cannot be made if there is an attempt to jump the timer of a coil that has been turned ON using the CJ, SCJ or JMP instruction.
2. Scan time is shortened if the CJ, SCJ or JMP instruction is used to force a jump to the OUT instruction.
3. Scan time is shortened if the CJ, SCJ or JMP instruction is used to force a jump to the rear.
4. The CJ, SCJ, and JMP instructions can be used to jump to a step prior to the step currently being executed. However, it is necessary to consider methods to get out of the loop so that the watchdog timer does not time out in the process.

5. The device to which a jump has been made with the CJ, SCJ or JMP instruction does not change.


Jumps to label P19 when XB turns ON.
Y43 and Y49 remain unchanged regardless of whether XB and XC are turned ON/OFF during the execution of \(C J\) instruction.
6. The jumping ranges of \(\mathrm{CJ}, \mathrm{SCJ}\), and JMP instructions are different according to the pointer type specified for \(p\). The following table shows the jumping ranges of pointer number and ladder block label.
\begin{tabular}{l|l}
\multicolumn{1}{c|}{ Pointer type } & \multicolumn{1}{c}{ Jumping range } \\
\hline Pointer number & Pointer numbers within the same program file \\
\hline Ladder block label & Ladder block labels within the same POU \\
\hline
\end{tabular}

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The pointer number which is not used as a label in the same program is specified.
(Error code: 4210)
- The common pointer number in another program is specified.
(Error code: 4210)

\section*{\(\triangle\) Program Example}
(1) The following program jumps to P 10 when g_bool1 turns ON.
[Structured ladder/FBD]

(2) The following program jumps to P 10 from the next scan after g_bool1 is turned ON. [Structured ladder/FBD]

(3) The following program jumps to Label3 when g_bool1 turns ON.
[Structured ladder/FBD]


\section*{Caution}
(1) For Universal model QCPU and LCPU, AND AM400 needs to be inserted directly prior to the SCJ instruction.
[Program example 1]
[Structured ladder/FBD]

[Program example 2]
[Structured ladder/FBD]


\subsection*{6.5.2 Jump to END processing}

\section*{GOEND}


Function
Jumps to the FEND instruction or END processing in the same program file.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The GOEND instruction has been executed after the execution of the CALL instruction prior to the execution of the RET instruction.
(Error code: 4211)
- The GOEND instruction has been executed after the execution of the FOR instruction, and prior to the execution of the NEXT instruction.
(Error code: 4200)
- The GOEND instruction has been executed during an interrupt program prior to the execution of the IRET instruction.
(Error code: 4221)
- The GOEND instruction was executed within the CHKCIR to CHKEND instruction loop.
(Error code: 4230)
- The GOEND instruction was executed within the IX to IXEND instruction loop.
(Error code: 4231)

\section*{/Program Example}

Execution of the program file is terminated if Var_D0 holds a negative number.
[Structured ladder/FBD]

[ST]
GOEND (Var_D0<0);

\subsection*{6.6 Program Execution Control Instructions}

\subsection*{6.6.1 Interrupt disable/enable, interrupt program mask}

\author{
DI, EI, IMASK
}

DI
IMASK
El


Input argument,
EN: Executing condition
: Bit
s: Interrupt mask data, or start number or the array of the device : Array of ANY16 (0..15)
that stores interrupt mask data
(when using the IMASK instruction)
Output argument, ENO: Execution result
: Bit
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J成成:} & \multirow[b]{2}{*}{U} & \multirow{2}{*}{Zn} & \multirow{2}{*}{Constant} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{6}{|c|}{-} \\
\hline
\end{tabular}

1 For Basic model QCPU

\section*{DI}
(1) Disables the execution of an interrupt program until the El instruction has been executed, even if a start cause for the interrupt program occurs.
(2) A DI status is entered when power is turned ON or when the CPU module is reset.

\section*{El}

Releases the disable interrupt status of the DI instruction, and enables the execution of the interrupt program with the interrupt pointer number which is allowed by the IMASK instruction.
When the IMASK instruction is not executed, I32 to I47 are disabled.


\section*{IMASK}
(1) Enables/disables the execution of the interrupt program marked by the specified interrupt pointer by using the bit pattern of 8 points from the device specified for (s).
- 1(ON)...... Interrupt program execution enabled
- O(OFF).... Interrupt program execution disabled
(2) The interrupt pointer numbers corresponding to each bit are as shown below.

(3) When the power is turned ON or when the CPU module has been reset, the execution of interrupt programs 10 to \(I 31\), 148 to \(I 127\) is enabled, and the execution of interrupt programs 132 to 147 is disabled.
(4) The status of devices © [0] to © [7] are stored to SD715 to SD717 and SD781 to SD785 (storage area for the IMASK instruction mask pattern).
(5) Although the special registers are separated as SD715 to SD717 and SD781 to SD785, device numbers should be specified as (s) [0] to (s) [7] successively.
1. For information on interrupt conditions, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used
2. The DI status (interrupt disabled) is active during the execution of an interrupt program. Do not insert the El instructions in interrupt programs to attempt the execution of multiple interrupts, with interrupt programs running inside interrupt programs.
3. If there are the El and DI instructions within a master control, these instructions will be executed regardless of the execution or non-execution status of the MC instruction.

No operation error occurs in the execution of the DI, EI, and IMASK instructions.

\section*{\(\triangle\) Program Example}

In the following program, the interrupt programs with the interrupt pointer number \(I 1\) and \(I 3\) are set in the enable execution status while XO is ON .
[Structured ladder/FBD]
- Task_01 [Always] ... interrupt enable program

The FEND instruction is not required.

- Task_Interrupt No1 [Event] ... I1 interrupt program

The IRET instruction is not required.

- Task_Interrupt No3 [Event] ... I3 interrupt program

The IRET instruction is not required.

```

[ST]
\bullet Task_01 [Always] ... interrupt enable program
IF (X0=FALSE) THEN
DI(TRUE);
END_IF;
IF (X1=FALSE) THEN
IF XO THEN
MOVP(TRUE,H0A,Var_D10[0]);
FMOVP(TRUE,0,7,Var_D10[1]));
END_IF;
IMASK(TRUE,Var_D10);
El(TRUE);
END_IF;
OUT(M0,Y20);

- Task_Interrupt No1 [Event] .. I1 interrupt program
MOVP(M10,10,Var_D100);
\bullet Task_Interrupt No3 [Event] ... I3 interrupt program
IF M11 THEN
Var_D200 := Var_D100 + Var_D200;
END_IF;

```

\section*{XPOINT}

For interrupt programs, create multiple tasks to register interrupt enable programs and interrupt programs.

+ Task_01 \(\qquad\) Interrupt enable program
+ Task_InterruptNo1 Interrupt I1 program
+ Task_InterruptNo3 Interrupt I3 program

Task_Interrupt No1 [Always] to [Event] ... Change to a interrupt program.
Task_Interrupt No3 [Always] to [Event] ... Change to a interrupt program.
[Property setting]


Enter the device I1 or I3.

\section*{\(\sqrt{3}\) Function}

\section*{DI}
(1) Disables the execution of an interrupt program until the El instruction has been executed, even if a start cause for the interrupt program occurs.
(2) A DI status is entered when power is turned ON or when the CPU module is reset.

\section*{El}

Releases the disable interrupt status of the DI instruction, and enables the execution of the interrupt program with the interrupt pointer number which is allowed by the IMASK instruction and the fixed scan execution type program.
When the IMASK instruction is not executed, I32 to I47 are disabled.


\section*{IMASK}
(1) Enables/disables the execution of the interrupt program marked by the specified interrupt pointer by using the bit pattern of 16 points from the device specified for © .
- 1(ON)...... Interrupt program execution enabled
- 0(OFF).... Interrupt program execution disabled
(2) The interrupt pointer numbers corresponding to each bit are as shown below.

(3) The following are the results when the power is turned ON or when the CPU module has been reset.
(a) For High Performance model QCPU

The execution of interrupt programs 10 to \(131, I 48\) to \(I 255\) is enabled, and the execution of interrupt programs I 32 to I 47 is disabled.
(b) For Universal model QCPU and LCPU

The execution of interrupt programs 10 to \(\mathrm{I} 31, \mathrm{I} 48\) to I 255 is enabled, and the execution of interrupt programs I 32 to I 44 is disabled.
(4) The status of devices © [0] to © [15] are stored to SD715 to SD717 and SD781 to SD793 (storage area for the IMASK instruction mask pattern).
(5) Although the special registers are separated as SD715 to SD717 and SD781 to SD793, device numbers should be specified as © \([0]\) to © \([15]\) successively.
®POINT
1. For information on interrupt conditions, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.
2. The DI status (interrupt disabled) is active during the execution of an interrupt program. Do not insert the El instructions in interrupt programs to attempt the execution of multiple interrupts, with interrupt programs running inside interrupt programs.
3. If there are the El and DI instructions within a master control, these instructions will be executed regardless of the execution or non-execution status of the MC instruction.

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The device specified for (s) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, the interrupt programs with the interrupt pointer number I 1 and I 3 are set in the enable execution status while XO is ON .
[Structured ladder/FBD]
- Task_01 [Always] ... interrupt enable program The FEND instruction is not required.

- Task_Interrupt No1 [Event] ... I1 interrupt program The IRET instruction is not required.

- Task_Interrupt No3 [Event] ... I3 interrupt program The IRET instruction is not required.

```

[ST]
\bullet Task_01 [Always] ... interrupt enable program
IF (X0=FALSE) THEN
DI(TRUE);
END_IF;
IF (X0=FALSE) THEN
IF XO THEN
MOVP(TRUE,H0A,Var_D10[0]);
FMOVP(TRUE,0,7,Var_D10[1]));
END_IF;
IMASK(TRUE,Var_D10);
El(TRUE);
END_IF;
OUT(M0,Y20);

- Task_Interrupt No1 [Event] .. I1 interrupt program
MOVP(M10,10,Var_D100);
\bullet Task_Interrupt No3 [Event] ... I3 interrupt program
IF M11 THEN
Var_D200 := Var_D100 + Var_D200;
END_IF;

```

\section*{XPOINT}

For interrupt programs, create multiple tasks to register interrupt enable programs and interrupt programs.

\begin{tabular}{|c|c|}
\hline Task_01 & Interrupt enable program \\
\hline Task_InterruptNo1 & Interrupt I1 program \\
\hline Task_InterruptNo3 & Interrupt l3 program \\
\hline
\end{tabular}

Task_Interrupt No1 [Always] to [Event] ... Change to a interrupt program. Task_Interrupt No3 [Always] to [Event] ... Change to a interrupt program.
[Property setting]


\footnotetext{
Enter the device I1 or I3
}

\subsection*{6.6.2 Recovery from interrupt programs}

\section*{Basic}

High
pefformanc
Process
Redundan Universal

LCPU

\section*{IRET}


: Bit


Function
Used to end the interrupt program processing forcedly.
Do not use when execute the interrupt program processing completely.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- A pointer corresponds to the interrupt number does not exist.
(Error code: 4220)
- The IRET instruction was executed before executing the interrupt program.
(Error code: 4223)
- The FEND, GOEND or STOP instruction was executed after the interruption and before the IRET instruction execution.
(Error code: 4221)
- The IRET instruction was executed in the fixed scan execution type program. (For Universal model QCPU and LCPU)
(Error code: 4223)

\section*{\(\square\) Program Example}

In the following program, 1 is subtracted from Var_D0 if g_bool1 is ON, and 1 is added to Var_D0 if \(g_{-}\)bool 1 is OFF when the interrupt of No. 3 occurs.
[Structured ladder/FBD]
- Task_01 [Always] ... interrupt program

The FEND instruction is not required.

- Task_Interrupt No3 [Event] ... interrupt program


\section*{®POINT}

For interrupt programs, create multiple tasks to register interrupt enable programs and interrupt programs.
+ Task_01
Interrupt enable program
+ Task_InterruptNo3 Interrupt program

Task_Interrupt No3 [Always] to [Event] ... Change to a interrupt program. [Property setting]


\subsection*{6.7 I/O Refresh Instructions}

\subsection*{6.7.1 I/O refresh}

\section*{RFS(P)}

> P: Executing condition

\(\begin{array}{llll}\text { Input argument, } & \mathrm{EN}: & \text { Executing condition } & : \text { Bit } \\ & \mathrm{s}: & \text { Start number of the device that executes the refresh function } & : \text { Bit } \\ & \mathrm{n}: & \text { Number of refreshes } & \text { : ANY16 } \\ \text { Output argument, } & \text { ENO: } & \text { Execution result } & \text { : Bit }\end{array}\)


\section*{Function}
(1) Refreshes only the device being scanned during a scan, and functions to fetch external input or to output data to an output module.
(2) Fetching of input from or sending output to an external source is conducted in batch only after the END processing in the program is executed, so it is not possible to output a pulse signal to an outside source during the execution of a scan. When the I/O refresh instruction is executed, the inputs \((X)\) or outputs \((Y)\) of the corresponding device numbers are refreshed forcibly midway through program execution. Therefore, a pulse signal can be output to an external source during a scan.
(3) Use direct access inputs (DX) or direct access outputs (DY) to refresh inputs (X) or outputs \((\mathrm{Y})\) in units of 1 point.


\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The range of \(n\) points from the device specified for (s) exceeds the proximate l/O range.
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the devices from X100 to X11F, and from Y200 to Y23F are refreshed when MO turns ON.
[Structured ladder/FBD]


\section*{[ST]}

RFSP(M0,X100,H20);
RFSP(M0,Y200,H40);

\subsection*{6.8 Other Convenient Instructions}

\subsection*{6.8.1 Single-phase input up/down counter}

\section*{UDCNT1}


\section*{UDCNT1}


Input argument,

Output argument,
\(\mathrm{EN}:\)
\(\mathrm{s}:\)
\(\mathrm{s}: \quad \mathrm{s}[0]:\) Input number for the counter input \(\mathrm{s}[1]\) :For setting the up/down counter
- OFF:Up-counter
- ON:Down-counter
\begin{tabular}{ll}
\(\mathrm{n}:\) & Set value \\
ENO: & Execution result
\end{tabular}
\(\mathrm{d}: \quad\) Counter number that starts counting by the UDCNT1
instruction
\begin{tabular}{|l|l|}
\hline UDCNT1 \\
\\
\\
\\
\\
\\
\hline
\end{tabular}
: Bit
: Array of bit (1..2)
: ANY16
: Bit
: ANY16
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{小..ala} & \multirow[b]{2}{*}{U} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & \(\bigcirc(X \text { only })^{* 1}\) & - & - & & & - & & & - \\
\hline n & \(\triangle^{* 2}\) & \(\triangle^{* 2}\) & \(\triangle^{*}{ }^{2}\) & & & \(\bigcirc\) & & & - \\
\hline (d) & - & \(\Delta^{*}\) (Conly) & - & & & - & & & - \\
\hline
\end{tabular}
*1: Specify the array in which X is set as a device for global label. X devices can be used only in the range of I/O points (accessible points to the actual I/O module).
*2: Local devices and file registers per program cannot be used as setting data.

\section*{F Function}
(1) When the input specified for © \((3]\) turns from OFF to ON, the current value of the counter specified for (d) will be updated.
(2) The direction of the count is determined by the ON/OFF status of the input specified for (s) [1].
- OFF : Up counter (counts by adding to the current value)
- ON : Down counter (counts by subtracting from the current value)
(3) Count processing is conducted as described below.
- When the count is going up, the counter contact specified for (d) turns ON when the current value becomes identical with the setting value specified for \(n\). However, the current value count will continue even when the contact of the counter specified for (d) turns ON. (Refer to Program Example)
- When the count is going down, the counter for the contact specified for (d) turns OFF when the current value reaches the setting value - 1. (Refer to Program Example)
- The counter specified for (d) is a ring counter. If it is counting up when the current value is 32767 , the current value will become -32768 . Further, if it is counting down when the current value is -32768 , the current value will become 32767 . The count processing performed on the current value is as shown below.

(4) The UDCNT1 instruction triggers counting when the execution command is turned from OFF to ON and suspends counting when the execution command is turned from ON to OFF. When the execution command is turned from OFF to ON again, the counting resumes from the suspended value.
(5) The RST instruction clears the current value of the counter specified for (d) and turns the contact OFF.

\section*{区POINT}
1. With the UDCNT1 instruction, the argument device data are registered in the work area of the CPU module and counting operation is processed as a system interrupt. (The device data registered in the work area is cleared by turning the execution command OFF, or turning the STOP/RUN switch from STOP to RUN.) For this reason, the pulses that can be counted must have longer ON and OFF times than the interrupt interval of the CPU module. The interrupt interval of each module is shown below
\begin{tabular}{l|c|}
\hline \multicolumn{1}{c|}{ CPU module model } & Interrupt interval \\
\hline \begin{tabular}{l} 
High Performance model QCPU, Process CPU, Universal model \\
QCPU, LCPU
\end{tabular} & 1 ms \\
\hline
\end{tabular}
2. The setting value cannot be changed during counting directed by the UDCNT1 instruction (while the execution command is ON). To change the setting value, turn OFF the execution command.
3. Counters which have been specified by the UDCNT1 instruction cannot be used by other instructions. If they are used by other instructions, they will not be capable of returning an accurate count.
4. The UDCNT1 instruction can be used as many as 6 times within all the programs being executed. The seventh and the subsequent UDCNT1 instructions are not processed.

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device specified for ©s exceeds the corresponding device range. (Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, the number of turns from OFF to ON in Var_X0 are counted using C0 (up/down counter) after X20 has turned ON.
[Structured ladder/FBD]


\section*{[ST]}

UDCNT1(X20,Var_X0,5,C0);
[Operation]


\subsection*{6.8.2 Two-phase input up/down counter}

\section*{UDCNT2}

\section*{UDCNT2}


Input argument,

Output argument

EN: Executing condition
\(\mathrm{s}: \quad \mathrm{s}[0]\) :Input number for the counter input (A-phase pulse) \(\mathrm{s}[1]\) :Input number for the counter input (B-phase pulse) n : Set value
ENO: Execution result
\(\mathrm{d}: \quad\) Counter number that starts counting by the UDCNT2
instruction
\begin{tabular}{|l|l|}
\hline UDCNT2 \\
\\
\\
\\
\\
\\
\hline
\end{tabular}
: Bit
Array of bit (1..2)
: ANY16
: Bit
: ANY16
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{} & \multirow[b]{2}{*}{U...igata} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant
\[
\mathrm{K}, \mathrm{H}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & \(\mathrm{O}(\mathrm{Xonly})^{* 1}\) & - & - & & & - & & & - \\
\hline n & \(\triangle^{* 2}\) & \(\triangle^{*}{ }^{2}\) & \(\triangle^{*}{ }^{2}\) & & & \(\bigcirc\) & & & - \\
\hline (d) & - & \(\Delta^{4}(\) Conly \()\) & - & & & - & & & - \\
\hline
\end{tabular}
*1: Specify the array in which X is set as a device for global label. X devices can be used only in the range of I/O points (accessible points to the actual I/O module).
*2: Local devices and file registers per program cannot be used as setting data.

\section*{\(\sqrt{2}\) Function}
(1) The current value of the counter specified for (d) is updated depending on the status of the input specified for © [5] (A-phase pulse) and the status of the input specified for © [s [1] (Bphase pulse).
(2) Direction of the count is determined in the following manner.
- When © \({ }^{[ }[0]\) is ON, if © \((1]\) turns from OFF to ON, count-up operation is performed. (Values are added to the current value of the counter.)
 (Values are subtracted from the current value of the counter.)
- No count operation is performed if © \([0]\) is OFF.
(3) Count processing is conducted as described below.
- When the count is going up, the counter contact specified for (d) turns ON when the current value becomes identical with the setting value specified for \(n\). However, the current value count will continue even when the contact of the counter specified for (d) turns ON. (Refer to Program Example)
- When the count is going down, the counter for the contact specified for (d) turns OFF when the current value reaches the setting value -1. (Refer to Program Example)
- The counter specified for (d) is a ring counter. If it is counting up when the current value is 32767, the current value will become - 32768. Further, if it is counting down when the current value is -32768 , the current value will become 32767 . The count processing performed on the current value is as shown below.

(4) Count processing conducted according to the UDCNT2 instruction begins when the execution command is turned from OFF to ON, and is suspended when it is turned from ON to OFF.
When the execution command is turned from OFF to ON again, the counting resumes from the suspended value.
(5) The RST instruction clears the current value of the counter specified for (d) and turns the contact OFF.

\section*{XPOINT}
1. 1. With the UDCNT2 instruction, the argument device data are registered in the work area of the CPU module and counting operation is processed as a system interrupt. (The device data registered in the work area is cleared by turning the execution command OFF, or turning the STOP/RUN switch from STOP to RUN.) For this reason, the pulses that can be counted must have longer ON and OFF times than the interrupt interval of the CPU module. The interrupt interval of each module is shown below:
\begin{tabular}{l|c|}
\hline \multicolumn{1}{c|}{ CPU module model } & Interrupt interval \\
\hline High Performance model QCPU, Process CPU, Universal model & 1 ms \\
QCPU, LCPU & \\
\hline
\end{tabular}
2. The setting value cannot be changed during counting directed by the UDCNT2 instruction (while the execution command is ON). To change the setting value, turn OFF the execution command.
3. Counters specified by the UDCNT2 instruction cannot be used by any other instructions. If they are used by other instructions, they will not be capable of returning an accurate count.
4. The UDCNT2 instruction can be used as many as 5 times within all the programs being executed. The sixth and the subsequent UDCNT2 instructions are not processed.

\section*{Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device specified for © exceeds the corresponding device range. (Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the status of Var_X0[0] and Var_X0[1] are counted by C0 (up/down counter) after X20 has turned ON.
[Structured ladder/FBD]

[ST]
UDCNT2(X20,Var_X0,3,CN0);
[Operation]
X20
Var_Xo[0]
Var_XO[1]
COM current value

CO contact


\subsection*{6.8.3 Teaching timer}

TTMR

\section*{TTMR}


Input argument,

Output argument,

EN: Executing condition
n : \(\quad\) Multiplier of the measurement value
ENO: Execution result
\(\mathrm{d}: \quad \mathrm{d}[0]:\) Measurement value storage device
\(\mathrm{d}[1]\) :For CPU module system use
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{小成:} & \multirow[b]{2}{*}{U"..tial} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline n & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{\(\bigcirc\)} & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{-} & - \\
\hline
\end{tabular}

\section*{IT Function}
(1) Measures the time while the execution command is ON in units of seconds, multiplies by the multiplier specified for n , and stores the result to the device specified for (d).
(2) Clears the device specified for (d) [0] or (d) [1] when the execution command is turned from OFF to ON.
(3) The multipliers that can be specified for n are as shown below.
\begin{tabular}{c|c}
n & Multiplier \\
\hline 0 & 1 \\
\hline 1 & 10 \\
\hline 2 & 100 \\
\hline
\end{tabular}

\section*{®POINT}
1. Time measurements are conducted when the TTMR instruction is executed. Using the JMP or similar instruction to jump the TTMR instruction will make it impossible to get an accurate measurement.
2. Do not change the multiplier specified for \(n\) while the TTMR instruction is being executed. Changing this multiplier will result in an inaccurate value being returned.
3. The TTMR instruction can also be used in low-speed execution type programs.
4. The device specified for (d) [1] is used by the system of the CPU module, so users should not change its value. If users change this value, the value stored to the device specified for © [0] will no longer be accurate.
(4) No processing is performed when the value specified for \(n\) is other than 0 to 2 .

\section*{Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device specified for © exceeds the corresponding device range.
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, amount of time turning X0 ON is stored to Var_D0.
[Structured ladder/FBD]

[ST]
TTMR(X0,0,Var_D0);

\subsection*{6.8.4 Special function timer}

STMR

\section*{STMR}


Input argument,

Output argument,

\section*{EN: Executing condition}
: Bit
s: Timer number
: ANY16
n : Set value : ANY16
ENO: Execution result
: Bit
d: d[0]:Off-delay timer output
: Array of bit (1..4)
d[1]:One-shot timer output after OFF
d[2]:One-shot timer output after ON
\(\mathrm{d}[3]\) :On-delay and off-delay timer output
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J.alal} & \multirow[b]{2}{*}{} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \(\triangle^{* 1}\) & - & & & - & & & - \\
\hline n & \(\bigcirc\) & \(\bigcirc\) & \(\bigcirc\) & & & \(\bigcirc\) & & & - \\
\hline (d) & \(\bigcirc\) & - & - & & & - & & & - \\
\hline
\end{tabular}
*1: Only for timers (T).

\section*{Function}
(1) The STMR instruction uses the 4 points from the device specified for © to perform four types of timer output.
- OFF-delay timer output (@[0])

Turns ON at the rising edge command of the STMR instruction, and after the falling edge command, turns OFF when the amount of time specified for \(n\) has passed.
- One-shot timer output after OFF (© [1])

Turns ON at the falling edge command of the STMR instruction, and turns OFF when the amount of time specified for \(n\) has passed.
- One-shot timer output after ON (© [2])

Turns ON at the rising edge the command for the STMR instruction, and turns OFF either when the amount of time specified for \(n\) has passed, or when the command of the STMR instruction turns OFF.
- ON-delay and OFF-delay timer output (©[3])

Turns ON at the falling edge of the timer coil, and after the falling edge command of the STMR instruction, turns OFF when the amount of time specified for \(n\) has passed.
(2) The timer coil specified for (s) turns ON at the rising edge and falling edge command of the STMR instruction, and starts measurement of the current value.
- The timer coil measures to the point where the value reaches the setting value specified for \(n\), then enters a time up status and turns OFF.
- If the command of the STMR instruction turns OFF before the timer coil reaches the time up status, it will remain ON. Timer measurement is continued at this time.
When the STMR instruction command turns ON once again, the current value will be cleared to 0 and measurement will begin once again.
(3) The timer contact turns ON at the rising edge command of the STMR instruction, and after the falling edge is reached, the timer coil turns OFF at the falling edge command of the STMR instruction.
The timer contact is used by the CPU module system, and cannot be used by the user.

(4) Measurement of the current value of the timer specified in the STMR instruction is conducted regardless of ON/OFF status of the STMR instruction.
If the STMR instruction is jumped with the JMP or similar instruction, it will not be possible to get accurate measurement.
(5) Measurement unit for the timer specified for (d) is identical to the low-speed timer.
(6) Values between 0 and 32767 can be set for \(n\). No processing is performed when the value specified for n is other than 0 to 32767.
(7) The timer specified for (s) cannot be used in the OUT instruction.

If the STMR instruction and the OUT instruction use the same timer number, accurate operation will not be conducted.

\section*{Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.

> - The device specified for © exceeds the corresponding device range.

\section*{\(\triangle\) Program Example}

In the following program, Y 0 and Y 1 are turned ON and OFF every second (flicker) when X 20 turns ON.
(Uses 100ms timer)
[Structured ladder/FBD]

[ST]
STMR(X20 AND NOT(Var_M0[3]),T0,10,Var_M0);
OUT(Var_M0[1],Y0);
OUT(Var_M0[2],Y1);
[Timing chart]


\section*{Caution}

Note that the STMR instruction operates when the instruction is used within the range written data by the online program change.

For details, refer to the User's Manual (Function Explanation, Program Fundamentals) for the CPU module used.

\subsection*{6.8.5 Rotary table shortest direction control}

ROTC


\section*{ROTC}


\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{6}{*}{Input argument,} & EN: & \multicolumn{5}{|l|}{Executing conditio} & \multicolumn{4}{|l|}{: Bit} \\
\hline & s: & \multicolumn{9}{|l|}{\(\mathrm{s}+0\) : For measuring the number of table rotations (for system) : Array of ANY16 (1..3)} \\
\hline & & \multicolumn{9}{|l|}{\(\mathrm{s}+1\) : Call station number} \\
\hline & & \multicolumn{9}{|l|}{\(\mathrm{s}+2\) : Call item number} \\
\hline & \(\mathrm{n} 1:\) & \multicolumn{5}{|l|}{Number of table divisions (2 to 32767)} & \multicolumn{4}{|l|}{: ANY16} \\
\hline & n2: & \multicolumn{5}{|l|}{Number of low-speed sections (value from 0 to n 1 )} & \multicolumn{3}{|l|}{: ANY16} & \\
\hline \multirow[t]{15}{*}{Output argument,} & ENO: & \multicolumn{5}{|l|}{Execution result} & \multicolumn{3}{|l|}{: Bit} & \\
\hline & d : & \multicolumn{5}{|l|}{\(\mathrm{d}[0]\) :A-phase input signal} & \multicolumn{3}{|l|}{: Array of bit (1..8)} & \\
\hline & & \multicolumn{9}{|l|}{\(\mathrm{d}[1]\) :B-phase input signal} \\
\hline & & \multicolumn{9}{|l|}{\(\mathrm{d}[2]: 0\) point detection input signal} \\
\hline & & \multicolumn{9}{|l|}{\(\mathrm{d}[3]\) :High-speed forward rotation output signal (for system)} \\
\hline & & \multicolumn{9}{|l|}{\(\mathrm{d}[4]\) :Low-speed forward rotation output signal (for system)} \\
\hline & & \multicolumn{9}{|l|}{\(\mathrm{d}[5]\) :Stop output signal (for system)} \\
\hline & & \multicolumn{9}{|l|}{\(\mathrm{d}[6]\) :Low-speed reverse rotation output signal (for system)} \\
\hline & & \multicolumn{9}{|l|}{d[7]:High-speed reverse rotation output signal (for system)} \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & & & \multirow[t]{2}{*}{U:..ing} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (s) & - & & & & & - & & & - \\
\hline & n1 & \(\bigcirc\) & & & & & \(\bigcirc\) & & & - \\
\hline & n2 & \(\bigcirc\) & & & & & \(\bigcirc\) & & & - \\
\hline & (d) & \(\bigcirc\) & & & & & - & & & - \\
\hline
\end{tabular}

\section*{\(\sqrt{2}\) Function}
(1) This control functions to enable shortest direction control of the rotary table to the position of the station number specified for (s[1] in order to remove or deposit an item whose number has been specified for (s)[2] on a rotary table with equal divisions of the value specified for n1.
(2) The item number and station number are controlled as items allocated by counterclockwise rotation.
(3) The system uses © [5] as a counter to instruct it as to what item is at which number counting from station number 0 . Do not rewrite the sequence program data.
Accurate controls will not be possible in cases where users have rewritten the data.
(4) The value of \(n 2\) should be less than the number of table divisions that were specified for \(n 1\).
(5) (d) [0] and © [1] are A and B phase input signals that are used to detect whether the direction of the rotary table rotation is forward or reverse.
The direction of rotation is judged by whether the B-phase pulse is at its rising or falling pulse when the A-phase pulse is ON.
- When the B-phase is at the rising edge: Forward rotation (clockwise rotation)
- When the B-phase is at the falling edge: Reverse rotation (counterclockwise rotation)
(6) © [2] is the 0 point detection output signal that turns ON when item number 0 has arrived at the station No. 0.
When the device specified for © [2] turns ON while the ROTC instruction is being executed, (s)[0] is cleared.

It is best to perform this clear operation first, then to begin shortest direction control with the ROTC instruction.
(7) The data from (d) [3] to © [ [7] consists of output signals needed to control the table's operation.
The output signal of one of the devices from (d) [3] to (d) [7] will turn ON in response to the execution results of the ROTC instruction.
(8) If the command for the ROTC instruction is OFF, clears all © [3] to © [7] without performing shortest direction control.
(9) The ROTC instruction can be used only one time in all programs where it is executed. Attempts to use it more than one time will result in inaccurate operations.
(10) No processing is performed when the value of ©s [0] to ©s[2], or the value of \(n 2\) is greater than n 1 .

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The device specified for (s) or (d) exceeds the corresponding device range.
(Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, item at section Var_D0[2] on a 10-division rotary table is taken in and out at section Var_D0[1], and the rotation direction and the control speed of the motor are determined by two sections ahead and behind of the item when the table is rotated at low speed.

\section*{[Structured ladder/FBD]}

[ST]
OUT(X0,Var_MO[0]);
OUT(X1,Var_MO[1]);
OUT(X2,Var_M0[2]);
ROTC(X10,Var_D0,10,2,Var_M0);


\subsection*{6.8.6 Ramp signal}

RAMP

RAMP


Input argument,
\begin{tabular}{ll} 
EN: & Executing condition \\
\(\mathrm{n} 1:\) & Initial value \\
n2: & Final value \\
n3: & Number of transitions \\
ENO: & Execution result \\
d1: & d1[0]:Current value \\
& d1[1]:Number of executions \\
d2: & d2[0]:Completion device \\
& d2[1]:Bit for selecting data retainment at completion
\end{tabular}
\begin{tabular}{|l|l|}
\hline RAMP \\
\\
\\
\\
\\
\\
\\
\hline
\end{tabular}
: Bit
: ANY16
: ANY16
: ANY16
: Bit
Array of ANY16 (1..2)
: Array of bit (1..2)


\section*{Function}
(1) When the execution command is ON, the following processing is executed.
- Shifts from the value specified for \(n 1\) to the value specified for \(n 2\) in the number of times specified for n3.
- For n3, specify the number of scans (number of shifts) required for shift from n1 to n2. No processing is performed when the value specified for n 3 is other than 0 to 32768.
- The system uses (d1][1] to store the number of times the instruction has been executed.
- The value of one variation (one scan) is obtained by the expression below.
\[
\text { Value of one variation (one scan })=\frac{(\text { Value specified by } n 2)-(\text { Value specified by } \mathrm{n} 1)}{(\text { Value specified by } \mathrm{n} 3)}
\]

Example 0 is varied to 350 in seven scans as shown below.


When the calculated one variation is indivisible, compensation is made to achieve the value specified for n 2 by the number of shifts specified for n 3 .
Hence, a linear ramp may not be made.
(2) If the scan is performed for the number of moves specified for \(n 3\), the completion device specified for (d2][0] will turn ON.
The ON/OFF status of the completion device and the value of © \({ }^{(11}\) [0] are determined by the ON/OFF status of the device specified for (d2[1].
- When (12][1] is OFF, ©22[0] will turn OFF at the next scan, and the RAMP instruction will begin a new transition operation from the initial value.
- When (d2)[1] is ON, (d2 [0] will remain ON, and the value of (d1][0] will not change.
(3) When the command is turned OFF during the execution of this instruction, the value of © [11] \([0]\) will not change following this.
When the command turns ON again, the RAMP instruction will begin a new transition from the initial value.
(4) Do not change the values of n 1 and n 2 before the completion device specified for (d2[2] turns ON.
Since the same expression is used every scan to calculate the value stored in ©11[1], changing \(\mathrm{n} 1 / \mathrm{n} 2\) may cause a sudden variation.

\section*{Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The device specified for (d1) or (d2) exceeds the corresponding device range.
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the value of Var_D0 is changed from 10 to 100 in a total of 6 scans, and the value of Var_D0 is saved when the transition has been completed.
[Structured ladder/FBD]


\section*{[ST]}

SET(X0,Var_MO[1]);
RAMP(X0,10,100,6,Var_D0,Var_M0);
[Timing chart]


\subsection*{6.8.7 Pulse density measurement}


\section*{SPD}

\begin{tabular}{|l|l|}
\hline SPD & \\
\\
\\
\\
\\
\\
\hline
\end{tabular}
\begin{tabular}{llll} 
Input argument, & EN: & Executing condition & \(:\) Bit \\
& \(\mathrm{s}:\) & Pulse input & \(:\) Bit \\
& \(\mathrm{n}:\) & Measurement time (unit: ms) & : ANY16 \\
Output argument, & ENO: & Execution result & Bit \\
& \(\mathrm{d}:\) & Start number of the device that stores the measurement result: ANY16
\end{tabular}


\section*{Function}
(1) The number of turning OFF \(\rightarrow\) ON input from the device specified for © © is counted for just the amount of time specified for n , and results of the count are stored to the device specified for (d).

(2) When the measurement directed by the SPD instruction has been completed, the measurement is executed again from 0.
Turn OFF the execution command to stop the measurement directed by the SPD instruction.

\section*{XPOINT}
1. With the SPD instruction, the argument device data are registered in the work area of the CPU module and counting operation is processed as a system interrupt. (The device data registered in the work area is cleared by turning the execution command OFF, or turning the STOP/RUN switch from STOP to RUN.) For this reason, the pulses that can be counted must have longer ON and OFF times than the interrupt interval of the CPU module. The interrupt interval of each module is shown below.
\begin{tabular}{l|c|}
\hline \multicolumn{1}{c|}{ CPU module model } & Interrupt interval \\
\hline High Performance model QCPU, Process CPU, Universal model & 1 ms \\
QCPU, LCPU & \\
\hline
\end{tabular}
2. When using the High Performance model QCPU

The instruction is not processed when \(n=0\).
3. The SPD instruction can be used as many as 6 times within all the programs being executed. The seventh and the subsequent SPD instructions are not processed.
4. While the measurement is in execution (while the command input is ON) by the SPD instruction, the setting value cannot be changed. Turn OFF the command input before changing the setting value.

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device specified for (s) exceeds the corresponding device range.
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the pulses input to X0 are measured for a period of 500 ms when X10 turns ON, and the result is stored to Var_D0.
[Structured ladder/FBD]

[ST]
SPD(X10,X0,500,Var_D0);

\subsection*{6.8.8 Fixed cycle pulse output}

PLSY

\begin{tabular}{|c|c|}
\hline PLSY & \\
\\
\\
\\
\\
\\
\end{tabular}
\begin{tabular}{llll} 
Input argument, & EN: & Executing condition & : Bit \\
& \(\mathrm{n} 1:\) & Frequency, or device number that stores frequency & : ANY16 \\
& \(\mathrm{n} 2:\) & Output counts, or device number that stores output counts & : ANY16 \\
Output argument, & ENO: & Execution result & : Bit \\
& \(\mathrm{d}:\) & Device number that generates pulse outputs & : Bit
\end{tabular}

(1) Outputs a pulse at a frequency specified for n 1 the number of times specified for n 2 , to the output module with the output signal (Y) specified for (d).
(2) Frequency of \(n 1\) can be set between 1 and 100 Hz .

If \(n 1\) is other than 1 to 100 Hz , the PLSY instruction will not be executed.
(3) The number of outputs that can be specified for n 2 is between 0 and \(65535(0000 \mathrm{H}\) to 0FFFFH).
If n 2 is set to 0 , pulses are continuously output.
(4) Only an output number corresponding to the output module can be specified for pulse output (d).
(5) Pulse output commences with the rising edge command of the PLSY instruction.

Pulse output is suspended when the PLSY instruction command turns OFF.
1. With the PLSY instruction, the argument device data are registered in the work area of the CPU module and counting operation is processed as a system interrupt. (The device data registered in the work area is cleared by turning the execution command OFF, or turning the STOP/RUN switch from STOP to RUN.) For this reason, the pulses that can be output must have longer ON and OFF times than the interrupt interval of the CPU module. The interrupt interval of each module is shown below.
\begin{tabular}{|c|c|}
\hline CPU module model & Interrupt interval \\
\hline High Performance model QCPU, Universal model QCPU, LCPU & 1 ms \\
\hline
\end{tabular}
2. Do not change the argument for the PLSY instruction during pulse output directed by the PLSY instruction (while the execution command is ON). To change the argument, turn OFF the execution command.
3. The PLSY instruction can be used only once in all programs executed by the CPU module. The second and the subsequent PLSY instructions are not processed.

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The device specified for (d) exceeds the corresponding device range.
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, 10Hz pulse is output for 5 times to Var_Y20 when X0 turns ON.
[Structured ladder/FBD]

[ST]
PLSY(X0,10,5,Var_Y20);

\subsection*{6.8.9 Pulse width modulation}

PWM

\begin{tabular}{|l|l|}
\hline PWM & \\
\\
\\
\\
\\
\\
\hline
\end{tabular}
\begin{tabular}{llll} 
Input argument, & EN: & Executing condition & : Bit \\
& \(\mathrm{n} 1:\) & ON-time, or device number that stores ON-times & : ANY16 \\
& \(\mathrm{n} 2:\) & Cycle, or device number that stores cycles & : ANY16 \\
Output argument, & ENO: & Execution result & : Bit \\
& d: & Device number that generates pulse outputs & : Bit
\end{tabular}


\section*{\(\mathcal{Y}\) Function}
(1) Outputs the ON-time specified for n 1 and the pulse of the cycle specified for n 2 are output to module specified for (d).

(2) The setting ranges for n 1 and n 2 are shown below.
\begin{tabular}{l|c}
\multicolumn{1}{c|}{ CPU module model } & Setting range for n1 and n2 [ms] *1 \\
\hline \begin{tabular}{l} 
High Performance model QCPU, Process CPU, Universal \\
model QCPU, LCPU
\end{tabular} & 1 to 65535 (0001 to 0FFFFF) \\
\hline
\end{tabular}
*1 : The value specified for n 1 should be less than the value specified for n 2 .
1. 1. With the PWM instruction, the argument device data are registered in the work area of the CPU module and counting operation is processed as a system interrupt. (The device data registered in the work area is cleared by turning the execution command OFF, or turning the STOP/RUN switch from STOP to RUN.) The interrupt interval of each module is shown below.
\begin{tabular}{|c|c|}
\hline CPU module model & Interrupt interval of n1, n2 \\
\hline High Performance model QCPU, Universal model QCPU, LCPU & 1 ms \\
\hline
\end{tabular}

For this reason, the PWM instruction can be used only once within all the programs being executed by the CPU module.
2. The instruction is not processed in any of the following cases.
- When both n 1 and n 2 are 0
- When n1 \(\geqq \mathrm{n} 2\)
- When the PWM instruction is executed twice or more.
3. Do not change the argument for the PWM instruction during pulse output directed by the PWM instruction (while the execution command is ON). To change the argument, turn OFF the execution command.

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device specified for (d) exceeds the corresponding device range.

In the following program, 100ms pulse is output for every second to Var_Y20 when X0 turns ON.
[Structured ladder/FBD]

[ST]
PWM(X0,100,1000,Var_Y20);

\subsection*{6.8.10 Matrix input}

Process
Redundant Universal

LCPU

\section*{MTR}


\section*{\(\mathcal{F}\) Function}
(1) It reads the input from 16 points \(\times \mathrm{n}\) rows from the input number specified for (s), then stores fetched input data to the device specified for (12) and the following devices.
(2) One row (16 points) can be fetched in 1 scan.
(3) Fetching from the first to the nth row is repeated.
(4) The first through the 16th points store the first row of data and the next 16 points store the second row of data at the devices following the device specified for (12).
For this reason, the space of \(16 \times \mathrm{n}\) points from the device specified for (d2) are occupied by the MTR instruction.
(5) (d1) is the output needed to select the row which will be fetched, and the system automatically turns it ON and OFF.

It uses the n points from the device specified for (d1).
(6) Only device numbers divisible by 16 can be specified for (5), (d1) and (12).
(7) For n , a value in the range from 2 to 8 can be assigned.
(8) No processing is performed in any of the following cases.
- The device number specified for (s), (d1), or (d2) is not a multiple of 16.
- The device specified for \((5\) is outside of the actual input range.
- The device specified for (d1) is outside of the actual output range.
- The space \(16 \times \mathrm{n}\) points following the device specified for (d2) exceeds the corresponding device range.
- The value for n is not between 2 and 8 .

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The device other than the input \((X)\) was specified for (s).
(Error code: 4101)
- The device other than the output \((Y)\) was specified for @11.
(Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, the 16 points \(\times 3\) matrix from X10 and the following devices are fetched when X0 turns ON, and the matrix is stored to M0 and the following devices.
[Structured ladder/FBD]

[ST]
MTR(X0,X10,3,Y20,M0);
[Operation]

(1) Note that the MTR instruction directly operates on actual input and output.

The output (d1) that had been turned ON by the MTR instruction does not turn OFF when the MTR command turns OFF.
Turn OFF the specified output (d1) in the sequence program.
(2) The MTR instruction execution interval must be longer than the total of response time of input and output modules.
If the set interval is shorter than the value indicated above, an input cannot be read correctly. If the scan time in a sequence program is short, select the constant scan and set the scan time longer than the total of response time.
(3) When using the MTR instruction, do not branch a line from (d2).

If branched, the operation result is not output to the devices or labels of branch destination correctly.
For details of branch point of line from destination, refer to Section 3.2.

MEMO

\section*{APPLICATION INSTRUCTIONS}
7.1 Logical Operation Instructions ..... 7-2
7.2 Rotation Instructions ..... 7-27
7.3 Shift Instructions ..... 7-41
7.4 Bit Processing Instructions ..... 7-56
7.5 Data Processing Instructions ..... 7-64
7.6 Structured Instructions ..... 7-112
7.7 Data Table Operation Instructions ..... 7-132
7.8 Buffer Memory Access Instructions ..... 7-144
7.9 Display Instructions ..... 7-152
7.10 Debug/Error Diagnostics Instructions ..... 7-163
7.11 String Processing Instructions ..... 7-171
7.12 Special Function Instructions ..... 7-253
7.13 Data Control Instructions ..... 7-332
7.14 File Register Switching Instructions ..... 7-351
7.15 Clock Instructions ..... 7-360
7.16 Extended Clock Instructions ..... 7-386
7.17 Program Control Instructions ..... 7-395
7.18 Other Instructions ..... 7-406

\subsection*{7.1 Logical Operation Instructions}
(1) The logical operation instructions perform logical OR, logical AND or other logical operations in unit of 1 bit.
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multirow[b]{2}{*}{Operation} & \multirow[b]{2}{*}{Description} & \multirow[t]{2}{*}{Formula for Operation} & \multicolumn{3}{|c|}{Example} \\
\hline & & & A & B & Y \\
\hline \multirow{4}{*}{Logical AND (AND)} & \multirow{4}{*}{A bit is set to 1 only when both input \(A\) and input \(B\) are 1 , otherwise set to 0} & \multirow{4}{*}{\(\mathrm{Y}=\mathrm{A} / \mathrm{B}\)} & 0 & 0 & 0 \\
\hline & & & 0 & 1 & 0 \\
\hline & & & 1 & 0 & 0 \\
\hline & & & 1 & 1 & 1 \\
\hline \multirow{4}{*}{\[
\begin{gathered}
\text { Logical OR } \\
\text { (OR) }
\end{gathered}
\]} & \multirow{4}{*}{A bit is set to 0 only when both input A and input \(B\) are 0 , otherwise set to 1} & \multirow{4}{*}{\(\mathrm{Y}=\mathrm{A}+\mathrm{B}\)} & 0 & 0 & 0 \\
\hline & & & 0 & 1 & 1 \\
\hline & & & 1 & 0 & 1 \\
\hline & & & 1 & 1 & 1 \\
\hline \multirow{4}{*}{Exclusive OR (XOR)} & \multirow{4}{*}{\begin{tabular}{l}
A bit is set to 0 when input \(A\) and input \\
\(B\) are equal, otherwise set to 1
\end{tabular}} & \multirow{4}{*}{\(Y=\bar{A} B+A \bar{B}\)} & 0 & 0 & 0 \\
\hline & & & 0 & 1 & 1 \\
\hline & & & 1 & 0 & 1 \\
\hline & & & 1 & 1 & 0 \\
\hline \multirow{4}{*}{Exclusive NOR (XNR)} & \multirow{4}{*}{A bit is set to 1 when input \(A\) and input \(B\) are equal, otherwise set to 0} & \multirow{4}{*}{\(Y=(\bar{A}+B)(A+\bar{B})\)} & 0 & 0 & 1 \\
\hline & & & 0 & 1 & 0 \\
\hline & & & 1 & 0 & 0 \\
\hline & & & 1 & 1 & 1 \\
\hline
\end{tabular}

\subsection*{7.1.1 Logical AND operation on 16-/32-bit data}


WAND(P)
DAND (P)
P: Executing condition


Input argument,
EN: Executing condition

:Bit
s1, s2: Data on which the logical AND operation is performed, or start :ANY16/32 number of the device that stores data
Output argument, ENO: Execution result
:Bit
\(\mathrm{d}: \quad\) Start number of the device that stores operation result
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..n)} & \multirow[b]{2}{*}{U...igat} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K,H} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s1) & \multicolumn{7}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (s2) & \multicolumn{7}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) & \multicolumn{7}{|c|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{Function}

\section*{WAND(P)}
(1) Performs a logical AND operation on each bit of the 16-bit data of the device specified for (51) and the 16 -bit data of the device specified for \({ }_{(\Omega 2}\), and the results are stored to the device specified for (d).
(5)

(2) For bit devices, the operation is performed as 0 s are stored to the bit devices which follow the digit-specified bit devices. (Refer to Program Examples (1) and (2))

\section*{DAND(P)}
(1) Performs a logical AND operation on each bit of the 32-bit data for the device specified for (s1) and the 32-bit data for the device specified for © \(\mathrm{S}_{2}\), and stores the results to the device specified for (d).

(d)

(2) For bit devices, the operation is performed as 0 s are stored to the bit devices which follow the digit-specified bit devices. (Refer to Program Example (3))

\section*{OO Operation Error}

No operation error occurs in the execution of the WAND \((P)\) and DAND \((P)\) instructions.

\section*{\(\triangle\) Program Example}
(1) In the following program, the logical AND operation is performed on the data from X10 to X1B and the data in Var_D33 when X0A turns ON, and the results are stored to Var_D40. [Structured ladder/FBD]

[ST]
WANDP(X0A,K3X10,Var_D33,Var_D40);
[Operation]


(2) In the following program, the logical AND operation is performed on the data in Var_D10 and Var_D20 when X1C turns ON, and the results are stored to the devices from M0 to M11.
[Structured ladder/FBD]

[ST]
WANDP(X1C,Var_D10,Var_D20,K3M0);
[Operation]



Not changed
(3) In the following program, the digit in the hundred-thousands place of the 8-digit BCD value in Var_D10 (sixth digit from the end) is masked to 0 when XOA turns ON, and the results are stored to the devices from Y10 to Y2B.
[Structured ladder/FBD]

[ST]
DANDP(X0A,Var_D10,H0FF0FFFFF,K7Y10);
[Operation]





Y2F-Y2C Y2B-Y28 Y27-Y24 Y23-Y20 Y1F-Y1CY1B-Y18 Y17-Y14 Y13-Y10 Y2B to Y10 \(1,1,1,00,0,1,0,0,0,0,1,0,0,0,1,0,1,0,1,0,1,1,1,1,0,0,0\)

Not changed

\subsection*{7.1.2 Logical AND operation on block data}

*1 :Same device numbers can be specified for (51) and (d), or (22) and (d).
(1) Performs a logical AND operation on the data \(n\) points from the device specified for (51), and the data \(n\) points from the device specified for \({ }_{(22}\), and stores the results to the device specified for (d) and the following devices.

\begin{tabular}{|c|c|}
\hline & \[
\vartheta
\] \\
\hline (d) &  \\
\hline (d) +1 & 1111110000000000000000 \\
\hline (d) +2 &  \\
\hline & \(\}\) \\
\hline (d) \(+(\mathrm{n}-2)\) & 0110010110010011001001001 \\
\hline (d) \(+(\mathrm{n}-1)\) & 1111110000000000000000 \\
\hline
\end{tabular}
(2) The constant can be specified for \(\Omega_{2}\) between -32768 and 32767 (16-bit BIN data).
\begin{tabular}{|c|c|c|}
\hline (51) &  & \\
\hline (51) +1 & 1/111110000:0000:0111111 & \\
\hline (51) +2 & \(000000000011.11,1.1111111\) & b15---------b8b7--------- b0 \\
\hline  & ) & BKAND (2) \(1,1 / 1,11111 / 1,1100000111111\) \\
\hline (51) \(+(\mathrm{n}-2)\) &  & \\
\hline (51) \(+(n-1)\) & 111/11100000 11, 1110000'0, & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline (d) & \[
\begin{aligned}
& \text { b15---------b8b7--------- b0 } \\
& \hline 0: 01110011100000010111
\end{aligned}
\] \\
\hline (d) +1 & 11111100000000:0111111 \\
\hline (d) +2 & 0000000000000001111111 \\
\hline & ) \\
\hline d \(+(\mathrm{n}-2)\) & \(0 \cdot 10010110: 1000000110.1\) \\
\hline (d) \(+(\mathrm{n}-1)\) & 11111110ioioioloioionoioio \\
\hline
\end{tabular}

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The range of \(n\) points from the device specified for (s1), (®2) or (d) exceeds the corresponding device.
(Error code: 4101)
- The device range of \(n\) points from the device specified for ©s) overlaps with the device range of \(n\) points from the device specified for (d).
(Except when the same device is specified for (51) and (d)) (Error code: 4101)
- The device range of \(n\) points from the device specified for ©s2 overlaps with the device range of \(n\) points from the device specified for (d).
(Except when the same device is specified for © and ©) (Error code: 4101)

\section*{Program Example}

In the following program, the logical AND operation is performed on the data in the devices from D100 to D102 and the data in the devices from R0 to R2 when X20 turns ON, and the results are stored to D200 and the following devices.
[Structured ladder/FBD]


\section*{[ST]}

Var_D0:=K3
BKANDP(X20,D100,R0,Var_D0,D200);
[Operation]


\subsection*{7.1.3 Logical OR operation on \(16-/ 32\)-bit data}


\section*{Function}

\section*{WOR(P)}
(1) Performs a logical OR operation on each bit of the 16-bit data of the device specified for (51) and the 16 -bit data of the device specified for \(\Omega_{2}\), and stores the results to the device specified for © \({ }^{(11}\).
(51)



(2) For bit devices, the operation is performed as 0 s are stored to the bit devices which follow the digit-specified bit devices. (Refer to Program Example (1))

\section*{DOR(P)}
(1) Performs a logical OR operation on each bit of the 32-bit data of the device specified for (11) and the 32 -bit data of the device specified for \(®_{2}\), and stores the results to the device specified for (d).
(s1) +1 (s1)

(2)

(d) +1
(d)

(2) For bit devices, the operation is performed as 0s are stored to the bit devices which follow the digit-specified bit devices. (Refer to Program Example (2))

\section*{O Operation Error}

No operation error occurs in the execution of the \(\operatorname{WOR}(P)\) and \(\operatorname{DOR}(P)\) instructions.

\section*{\(\triangle\) Program Example}
(1) In the following program, the logical OR operation is performed on the data from X 10 to X 1 B , and the data in Var_D33 when X0A turns ON, and the results are output to Y30 to Y3B.
[Structured ladder/FBD]


\section*{[ST]}

WORP(X0A,K3X10,Var_D33,K3Y30);

\section*{[Operation]}

(2) In the following program, the logical OR operation is performed on the 32-bit data in Var_D0, and the 24-bit data from X20 to X37 when M8 turns ON, and the results are stored to Var_D23.
[Structured ladder/FBD]


\section*{[ST]}

DORP(M8,Var_D0,K6X20,Var_D23);

\section*{[Operation]}


(d) +1
(d)


\subsection*{7.1.4 Logical OR operation on block data}

BKOR

Basic
High
Process
Resumandan
Universal
LCPU

*1 :Same device numbers can be specified for (51) and (d), or (22) and (©).
(1) Performs a logical OR operation on the data \(n\) points from the device specified for (s1), and the data \(n\) points from the device specified for \(\Omega_{2}\), and stores the results to the device specified for (d) and the following devices.

(2) The constant can be specified for ©2 between -32768 and 32767 (16-bit BIN data).


\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The range of \(n\) points from the device specified for (s1), (®2) or (d) exceeds the corresponding device.
(Error code: 4101)
- The device range of \(n\) points from the device specified for ©s1 overlaps with the device range of \(n\) points from the device specified for (d). (Except when the same device is specified for (51) and (d))
(Error code: 4101)
- The device range of \(n\) points from the device specified for \({ }_{〔 2}\) overlaps with the device range of \(n\) points from the device specified for (d). (Except when the same device is specified for \((22)\) and (d))
(Error code: 4101)

\section*{Program Example}

In the following program, the logical OR operation is performed on the data in the devices from D100 to D102 and the data in the devices from R0 to R2 when X20 turns ON, and the result are stored to D200 and the following devices.
[Structured ladder/FBD]


\section*{[ST]}

Var_D0:=K3;
BKORP(X20,D100,R0,Var_D0,D200);

\section*{[Operation]}

\subsection*{7.1.5 Exclusive OR operation on 16-/32-bit data}

WXOR(P)
DXOR(P)


Input argument,

Output argument

EN: Executing condition
s1, s2: Data on which the exclusive OR operation is performed, or start number of the device that stores data
ENO: Execution result
d1: Start number of the device that stores exclusive OR operation :ANY16/32
result


\section*{Function}

\section*{WXOR(P)}
(1) Performs an exclusive OR operation on each bit of the 16-bit data of the device specified for (s1) and the 16-bit data of the device specified for (s2), and stores the results to the device specified for © \({ }^{(11)}\).

(s)

(d)

(2) For bit devices, the operation is performed as 0 s are stored to the bit devices which follow the digit-specified bit devices. (Refer to Program Example (1))

\section*{DXOR(P)}
(1) Performs an exclusive OR operation on each bit of the 32-bit data of the device specified for (51) and the 32-bit data of the device specified for (s2), and stores the results to the device specified for (d1).
(s1)

XOR
(s2)
(s2)

(d)

(2) For bit devices, the operation is performed as 0 s are stored to the bit devices which follow the digit-specified bit devices.

\section*{\(\bigcirc\) Operation Error}

No operation error occurs in the execution of the \(\mathrm{WXOR}(\mathrm{P})\) and \(\mathrm{DXOR}(\mathrm{P})\) instructions.

\section*{\(\triangle\) Program Example}
(1) In the following program, the exclusive OR operation is performed on the data from X 10 to X1B and the data in Var_D33 when X10 turns ON, and the results are output to Y30 to Y3B. [Structured ladder/FBD]

[ST]
WXORP(X10,K3X10,Var_D33,K3Y30);
[Operation]

Regarded as 0s.
XOR
Var_D33

(2) In the following program, the exclusive OR operation is performed on the data in Var_D20 and the data in Var_D30 when X10 turns ON, and the results are stored to Var_D40.
[Structured ladder/FBD]

[ST]
DXORP(X10,Var_D20,Var_D30,Var_D40);
[Operation]


\subsection*{7.1.6 Exclusive OR operation on block data}

BKXOR

BKXOR(P)



Input argument,
EN: Executing condition
s1: Start number of the device that stores data on which the
exclusive OR operation is performed
s2: Exclusive OR operation data, or start number of the device that stores data
n: Number of operation data :ANY16
Output argument,

Bit
:ANY16
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..flat} & \multirow[b]{2}{*}{U:...icai..} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant
K,H} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) *1 & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & - & - \\
\hline (s2) *1 & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & \(\bigcirc\) & - \\
\hline n & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) *1 & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & - & - \\
\hline
\end{tabular}
*1 :Same device numbers can be specified for (51) and (d), or (22) and (d).
(1) Performs an exclusive OR operation on the data \(n\) points from the device specified for (91), and the data n points from the device specified for \(\Omega_{2}\), and stores the results to the device specified for (d) and the following devices.

(d) b15---------b8b7---------b0
(d) +1 1111110:0:000 1110:0 00:0:0
(d) +2 1,1,1,10,0:000000 \(1,1,1,1\)
(d) \(+(n-2)\)

(d) \(+(\mathrm{n}-1) \quad 00000: 0,0: 0,0\) 1:1,1,1
(2) The constant can be specified for ©2 between -32768 and 32767 (16-bit BIN data).


\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The range of \(n\) points from the device specified for (s1), (2) or (d) exceeds the corresponding device.
(Error code: 4101)
- The device range of \(n\) points from the device specified for (sl) overlaps with the device range of \(n\) points from the device specified for (d). (Except when the same device is specified for (51) and (d))
(Error code: 4101)
- The device range of \(n\) points from the device specified for \(\left.{ }_{(S 2}\right)\) overlaps with the device range of \(n\) points from the device specified for (d). (Except when the same device is specified for (®2) and (d))
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the exclusive OR operation is performed on the data in the devices from D100 to D102 and the data in the devices from R0 to R2 when X20 turns ON, and the results are stored to D200 and the following devices.
[Structured ladder/FBD]

[ST]
Var_D0:=3;
BKXORP(X20,D100,R0,Var_D0,D200);
[Operation]



\subsection*{7.1.7 Exclusive NOR operation on 16-/32-bit data}


\section*{\(\sqrt{3}\) Function}

\section*{WXNR(P)}
(1) Performs an exclusive NOR operation on the 16-bit data of the device specified for (51) and the 16-bit data of the device specified for (s2), and stores the results to the device specified for (d1).

(2) For bit devices, the operation is performed as 0s are stored to the bit devices which follow the digit-specified bit devices.

\section*{DXNR(P)}
(1) Performs an exclusive NOR operation on the 32-bit data of the device specified for (s1) and the 32-bit data of the device specified for \(\S_{2}\), and stores the results to the device specified for (d).
(s1)

(s2)

(d)

(2) For bit devices, the operation is performed as 0s are stored to the bit devices which follow the digit-specified bit devices.

\section*{O Operation Error}

No operation error occurs in the execution of the \(W X N R(P)\) and \(\operatorname{DXNR}(P)\) instructions.

\section*{\(\square\) Program Example}
(1) In the following program, the exclusive NOR operation is performed on the 16-bit data from X30 to X3F and the data in Var_D99 when X0 turns ON, and the results are stored to Var_D7.
[Structured ladder/FBD]

[ST]
WXNRP(X0,K4X30,Var_D99,Var_D7);
[Operation]
\begin{tabular}{|c|c|c|c|c|}
\hline \multicolumn{5}{|r|}{X3F---- X3C X3B----- X38 X37 - - - - X34 X33 - - - - X30} \\
\hline \multirow[t]{2}{*}{X3F ~X30} & \(\begin{array}{l:l:l:l}0 & 1 & 0 & 1\end{array}\) & \(\begin{array}{l:l:l:l}0 & 1 & 0 & 1\end{array}\) & \(\begin{array}{l:l:l:l}0 & 1 & 0 & 1\end{array}\) & \begin{tabular}{l:l:l:l}
0 & 1 & 0 & 1 \\
\hline
\end{tabular} \\
\hline & \multicolumn{4}{|c|}{XNR} \\
\hline & \multicolumn{4}{|l|}{} \\
\hline \multirow[t]{3}{*}{Var_D99} & \begin{tabular}{l:l:l:l}
1 & 1 & 0 & 0 \\
\hline
\end{tabular} & \begin{tabular}{l:l:l:l}
1 & 0 & 0 & 1 \\
\hline
\end{tabular} & \begin{tabular}{l:l:l:l}
1 & 1 & 1 & 1 \\
\hline
\end{tabular} & \begin{tabular}{l:l:l:l|}
1 & 1 & 0 & 0 \\
\hline
\end{tabular} \\
\hline & & & & \\
\hline & \multicolumn{4}{|l|}{} \\
\hline Var_D7 & \(\begin{array}{l:l:l:l}0 & 1 & 1 & 0\end{array}\) & \(\begin{array}{l:l:l:l}0 & 0 & 1 & 1\end{array}\) & \(\begin{array}{l:l:l:l}0 & 1 & 0 & 1\end{array}\) & \begin{tabular}{l:l:l:l}
0 & 1 & 1 & 0 \\
\hline
\end{tabular} \\
\hline
\end{tabular}
(2) In the following program, the exclusive NOR operation is performed on the 32-bit data in Var_D20 and the data in Var_D10 when X10 turns ON, and the results are stored to Var_D40.
[Structured ladder/FBD]

[ST]
DXNRP(X10,Var_D20,Var_D10,Var_D40);

\section*{[Operation]}
(s1)

(s2)

勺
(d)


\subsection*{7.1.8 Exclusive NOR operation on block data}

BKXNR

\section*{Basic}

*1 :Same device numbers can be specified for (51) and (d), or (22) and (d).

\section*{\(\sqrt{3}\) Function}
(1) Performs an exclusive NOR operation on the data \(n\) points from the device specified for (s1), and the data n points from the device specified for \(\Im_{2}\), and stores the results to the device specified for (d) and the following devices.

(2) The constant can be specified for \({ }_{\Omega 2}\) between -32768 and 32767 (16-bit BIN data).


\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The range of \(n\) points from the device specified for (s1), (22) or (d) exceeds the corresponding device.
(Error code: 4101)
- The device range of \(n\) points from the device specified for (51) overlaps with the device range of \(n\) points from the device specified for (d). (Except when the same device is specified for (s1) and (d))
(Error code: 4101)
- The device range of n points from the device specified for ©2 overlaps with the device range of \(n\) points from the device specified for (d). (Except when the same device is specified for (s2) and (d))
(Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, the exclusive NOR operation is executed on the data in the devices from D100 to D102 and the data in the devices from R0 to R2 when X20 turns ON, and the results are stored to D200 and the following devices.
[Structured ladder/FBD]

[ST]
BKXNRP(X20,D100,R0,Var_D0,D200);

\section*{[Operation]}


\subsection*{7.2 Rotation Instructions}

\subsection*{7.2.1 Right rotation of 16-bit data}

\author{
ROR, RCR
}

\section*{Basic}

High
Process
Redundant Universa
LCPU

ROR(P)
RCR(P)
P: Executing condition
\(\approx\)

\(\begin{array}{lll}\text { Input argument, } & \text { EN: } & \text { Executing condition } \\ & \mathrm{n}: & \text { Number of rotations (0 to 15) } \\ \text { Output argument, } & \text { ENO: } & \text { Execution result } \\ & \mathrm{d}: & \text { Start number of the device to be rotated }\end{array}\)

:Bit
:ANY16
:Bit
:ANY16


\section*{ROR(P)}
(1) Rotates 16-bit data of the device specified for © , not including the carry flag, \(n\) bits to the right.
The carry flag is ON or OFF depending on the status prior to the execution of the ROR instruction.

(2) When a bit device is specified for © , a rotation is performed within the device range set by digit specification.
The number of bits that rotate is the remainder of \(n\) divided by 'number of bits specified by digit specification'.
For example, when \(\mathrm{n}=15\) and 'number of bits specified by digit specification' = 12 bits, the remainder of \(15 / 12=1\) is 3 , thus the data are rotated 3 bits.
(3) Specify any of 0 to 15 for \(n\).

If the value specified for n is 16 or higher, the remainder of n divided by 16 is used for rotation.

For example, when \(n=18\), the remainder of \(18 / 16=1\) is 2 , thus the data are rotated 2 bits to the right.

\section*{RCR(P)}
(1) Rotates 16-bit data of the device specified for (d), including the carry flag, \(n\) bits to the right. The carry flag is ON or OFF depending on the status prior to the execution of the RCR instruction.

(2) When a bit device is specified for (d), a rotation is performed within the device range set by digit specification.
The number of bits that rotate is the remainder of \(n\) divided by 'number of bits specified by digit specification'.
For example, when \(\mathrm{n}=15\) and 'number of bits specified by digit specification' = 12 bits, the remainder of \(15 / 12=1\) is 3 , thus the data are rotated 3 bits.
(3) Specify any of 0 to 15 for \(n\).

If the value specified for n is 16 or higher, the remainder of n divided by 16 is used for rotation.
For example, when \(n=18\), the remainder of \(18 / 16=1\) is 2 , thus the data are rotated 2 bits to the right.

\section*{Operation Error}

No operation error occurs in the execution of the \(\operatorname{ROR}(P)\) and \(R C R(P)\) instructions.

\section*{Program Example}
(1) In the following program, the data in Var_D0 are rotated, not including the carry flag, 3 bits to the right, when XOC turns ON.
[Structured ladder/FBD]

[ST]
RORP(X0C,3,Var_D0);
[Operation]

(2) In the following program, the data in Var_D0 are rotated, including the carry flag, 3 bits to the right, when XOC turns ON.
[Structured ladder/FBD]


\section*{[ST]}

RCRP(X0C,3,Var_D0);
[Operation]


\subsection*{7.2.2 Left rotation of 16 -bit data}

\section*{ROL, RCL \\ }


\section*{I Function}

\section*{ROL(P)}
(1) Rotates 16-bit data of the device specified for © , not including the carry flag, \(n\) bits to the left.
The carry flag turns ON or OFF depending on its status prior to the execution of the ROL instruction.

(2) When a bit device is specified for (d), a rotation is performed within the device range of points by digit specification.
The number of bits that rotate is the remainder of \(n\) divided by 'number of bits specified by digit specification'.
For example, when \(\mathrm{n}=15\) and 'number of bits specified by digit specification' = 12 bits, the remainder of \(15 / 12=1\) is 3 , thus the data are rotated 3 bits.
(3) Specify any of 0 to 15 for \(n\).

If the value specified for n is 16 or higher, the remainder of n divided by 16 is used for rotation.
For example, when \(n=18\), the remainder of \(18 / 16=1\) is 2 , thus the data are rotated 2 bits to the left.

\section*{RCL(P)}
(1) Rotates 16-bit data of the device specified for (d), including the carry flag, \(n\) bits to the left. The carry flag turns ON or OFF depending on its status prior to the execution of the RCL instruction.

(2) When a bit device is specified for (d), a rotation is performed within the device range of points by digit specification.
The number of bits that rotate is the remainder of \(n\) divided by 'number of bits specified by digit specification'.
For example, when \(\mathrm{n}=15\) and 'number of bits specified by digit specification' \(=12\) bits, the remainder of \(15 / 12=1\) is 3 , thus the data are rotated 3 bits.
(3) Specify any of 0 to 15 for \(n\).

If the value specified for n is 16 or higher, the remainder of n divided by 16 is used for rotation.
For example, when \(n=18\), the remainder of \(18 / 16=1\) is 2 , thus the data are rotated 2 bits to the left.

\section*{Operation Error}

No operation error occurs in the execution of the \(\operatorname{ROL}(P)\) and \(R C L(P)\) instructions.

\section*{Program Example}
(1) In the following program, the data in Var_D0 are rotated, not including the carry flag, 3 bits to the left, when X0C turns ON.
[Structured ladder/FBD]

[ST]
ROLP(X0C,3,Var_D0);

\section*{[Operation]}

(2) In the following program, the data in Var_D0 are rotated, including the carry flag, 3 bits to the left, when XOC turns ON.
[Structured ladder/FBD]

[ST]
RCLP(X0C,3,Var_D0);

\section*{[Operation]}

*ON/OFF status of the carry flag depends on its status before the execution of RCL instruction.

\subsection*{7.2.3 Right rotation of 32-bit data}

\section*{DROR, DRCR \\ }

DROR(P)
DRCR(P)
P: Executing condition


:Bit
:ANY16
:Bit
:ANY32
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...1)} & \multirow{2}{*}{U..1g:...} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{\begin{tabular}{l}
Constant \\
K,H
\end{tabular}} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline n & \multicolumn{7}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) & \multicolumn{7}{|c|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{\(\{\) Function}

\section*{DROR(P)}
(1) Rotates 32-bit data of the device specified for (d), not including the carry flag, \(n\) bits to the right.
The carry flag turns ON or OFF depending on its status prior to the execution of the DROR instruction.

(2) When a bit device is specified for (d), a rotation is performed within the device range of points by digit specification.
The number of bits that rotate is the remainder of \(n\) divided by 'number of bits specified by digit specification'.
For example, when \(\mathrm{n}=31\) and 'number of bits specified by digit specification' \(=24\) bits, the remainder of \(31 / 24=1\) is 7 , thus the data are rotated 7 bits.
(3) Specify any of 0 to 31 for \(n\).

If the value specified for n is 32 or higher, the remainder of n divided by 32 is used for rotation.
For example, when \(n=34\), the remainder of \(34 / 32=1\) is 2 , thus the data are rotated 2 bits to the right.

\section*{DRCR(P)}
(1) Rotates 32-bit data of the device specified for (d), including carry flag, \(n\) bits to the right. The carry flag turns ON or OFF depending on its status prior to the execution of the DRCR instruction.

(2) When a bit device is specified for (d), a rotation is performed within the device range of points by digit specification. The number of bits that rotate is the remainder of \(n\) divided by 'number of bits specified by digit specification'.
For example, when \(n=31\) and 'number of bits specified by digit specification' \(=24\) bits, the remainder of \(31 / 24=1\) is 7 , thus the data are rotated 7 bits.
(3) Specify any of 0 to 31 for \(n\).

If the value specified for n is 32 or higher, the remainder of n divided by 32 is used for rotation. For example, when \(n=34\), the remainder of 34 / \(32=1\) is 2 , thus the data are rotated 2 bits to the right.

No operation error occurs in the execution of the \(\operatorname{DROR}(P)\) and \(\operatorname{DRCR}(P)\) instructions.

\section*{\(\triangle\) Program Example}
(1) In the following program, the data of Var_D0 are rotated, not including the carry flag, 4 bits to the right, when XOC turns ON.
[Structured ladder/FBD]

[ST]
DRORP(XOC, 4, Var_DO \() ;\)
[Operation]

(2) In the following program, the data of Var_D0 are rotated, including the carry flag, 4 bits to the right, when XOC turns ON.
[Structured ladder/FBD]

[ST]
DRCRP(X0C,4,Var_D0);
[Operation]
\begin{tabular}{|c|c|}
\hline  & Carry flag (SM700)
\(\square\) \\
\hline b31--b28b27--b24 b23--b20b19--b16 b15--b12b11--b8 b7--b4 b3 - - b0 & Carry flag (SM700) \\
\hline  & 1 \\
\hline \begin{tabular}{ll}
\begin{tabular}{l} 
Content of b2 to b0 \\
before execution
\end{tabular}\(\quad\)\begin{tabular}{l} 
Content of carry flag \\
SM700 before execution
\end{tabular} & Before execution
\end{tabular} & Content of b3 before execution \\
\hline
\end{tabular}
* : ON/OFF status of the carry flag depends on its status before the execution of DRCR instruction.

\subsection*{7.2.4 Left rotation of 32-bit data}

\section*{DROL, DRCL}


\section*{Function}

\section*{DROL(P)}
(1) Rotates 32-bit data of the device specified for (d), not including the carry flag, \(n\) bits to the left. The carry flag turns ON or OFF depending on its status prior to the execution of the DROL instruction.

(2) When a bit device is specified for (d), a rotation is performed within the device range set by digit specification. The number of bits that rotate is the remainder of \(n\) divided by 'number of bits specified by digit specification'.
For example, when \(n=31\) and 'number of bits specified by digit specification' \(=24\) bits, the remainder of \(31 / 24=1\) is 7 , thus the data are rotated 7 bits.
(3) Specify any of 0 to 31 for \(n\).

If the value specified for n is 32 or higher, the remainder of n divided by 32 is used for rotation. For example, when \(n=34\), the remainder of \(34 / 32=1\) is 2 , thus the data are rotated 2 bits to the left.

\section*{DRCL(P)}
(1) Rotates 32-bit data of the device specified for © , including the carry flag, n bits to the left. The carry flag turns ON or OFF depending on its status prior to the execution of the DRCL instruction.

(2) When a bit device is specified for (d), a rotation is performed within the device range of points by digit specification. The number of bits that rotate is the remainder of \(n\) divided by 'number of bits specified by digit specification'.
For example, when \(\mathrm{n}=31\) and 'number of bits specified by digit specification' \(=24\) bits, the remainder of \(31 / 24=1\) is 7 , thus the data are rotated 7 bits.
(3) Specify any of 0 to 31 for \(n\).

If the value specified for n is 32 or higher, the remainder of n divided by 32 is used for rotation. For example, when \(n=34\), the remainder of \(34 / 32=1\) is 2 , the data are rotated 2 bits to the left.

\section*{Operation Error}

No operation error occurs in the execution of the \(\operatorname{DROL}(P)\) and \(\operatorname{DRCL}(P)\) instructions.

\section*{\(\square\) Program Example}
(1) In the following program, the data in Var_D0 are rotated, not including the carry flag, 4 bits to the left, when XOC turns ON.
[Structured ladder/FBD]

[ST]
DROLP(X0C,4,Var_D0);
[Operation]
b31--b28b27--b24b23--b20b19--b16b15--b12b11-b8 b7--b4 b3 --b0

Content of b27 to b0 Content of b31 to b28 before execution before execution
(2) In the following program, the data in Var_D0 are rotated, including the carry flag, 4 bits to the left, when XOC turns ON.
[Structured ladder/FBD]


\section*{[ST]}

DRCLP(X0C,4,Var_D0);

\section*{[Operation]}

* : ON/OFF status of the carry flag depends on its status before the execution of DRCL instruction.

\subsection*{7.3 Shift Instructions}

\section*{7.3 .1 n-bit right/left shift of 16-bit data}


\section*{Input argument,}

Output argument,

:Bit
:ANY16
:Bit
:ANY16


\section*{Function}

\section*{SFR(P)}
(1) Shifts to the right by \(n\) bits of the 16-bit data in the device specified for (d). The n bits from the most significant bit become 0 .

(2) When a bit device is specified for (d), a right shift is performed within the device range of points by digit specification.


The number of bits that shift is the remainder of \(n\) divided by 'number of bits specified by digit specification'.
For example, when \(\mathrm{n}=15\) and 'number of bits specified by digit specification' \(=8\) bits, the remainder of \(15 / 8=1\) is 7 , thus the data are shifted 7 bits.
(3) Specify any of 0 to 15 for \(n\).

If the value specified for n is 16 or higher, the remainder of n divided by 16 is used for right shift.
For example, when \(n=18\), the remainder of \(18 / 16=1\) is 2 , thus the data are shifted 2 bits to the right.

\section*{SFL(P)}
(1) Shifts 16-bit data in the device specified for (d) n bits to the left.

The n bits from the lowest bit become 0 .

(2) When a bit device is specified for (d), a left shift is performed within the device range set by digit specification.


The number of bits that shift is the remainder of \(n\) divided by 'number of bits specified by digit specification'. For example, when \(\mathrm{n}=15\) and 'number of bits specified by digit specification' \(=8\) bits, the remainder of \(15 / 8=1\) is 7 , thus the data are shifted 7 bits.
(3) Specify any of 0 to 15 for \(n\).

If the value specified for n is 16 or higher, the remainder of n divided by 16 is used for left shift.
For example, when \(n=18\), the remainder of \(18 / 16=1\) is 2 , thus the data are shifted 2 bits to the left.

\section*{Operation Error}

No operation error occurs in the execution of the \(\operatorname{SFR}(P)\) and \(\operatorname{SFL}(P)\) instructions.

\section*{\(\square\) Program Example}
(1) In the following program, the data in Var_D0 are shifted to the right by the number of bits specified for Var_D100 when X20 turns ON.
[Structured ladder/FBD]

[ST]
Var_D0:=3;
SFRP(X20,Var_D0,Var_D100);
[Operation]

(2) In the following program, the data in the devices from X 10 to X 17 are shifted 3 bits to the left when X1C turns ON.
[Structured ladder/FBD]

[ST]
SFLP(X1C,3,K2Y10);
[Operation]


\subsection*{7.3.2 1-bit right/left shift of \(n\)-bit data}


\section*{3 Function}

\section*{BSFR(P)}
(1) Shifts the data \(n\) points from the device specified for © to the right by 1 bit.

(2) The device specified for © \(+(n-1)\) becomes 0 .

\section*{BSFL(P)}
(1) Shifts the data \(n\) points from the device specified for © to the left by 1 bit.

(2) The device specified for (a) becomes 0 .

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The range of \(n\) points from the device specified for (d) exceeds the corresponding device.
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the data in the devices from M668 to M676 are shifted to the right when X8F turns ON.
[Structured ladder/FBD]

[ST]
BSFRP(X8F,9,M668);
[Operation]


Becomes 0
(2) In the following program, the data in the devices from Y60 to Y6F are shifted to the left when X4 turns ON.
[Structured ladder/FBD]

[ST]
BSFLP(X4,16,Y60);
[Operation]


\section*{Caution}

When using the BSFR/BSFL instruction, do not branch a line from (d).
If branched, the operation result is not output to the devices or labels of branch destination correctly.

For details of branch point of line from destination, refer to Section 3.2.

\subsection*{7.3.3 n-bit right/left shift of \(n\)-bit data}

SFTBR, SFTBL

- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported


\section*{Function}

\section*{SFTBR(P)}
(1) Shifts the n 1 -bit data from the device specified for (d) to the right by n 2 bit. When \(\mathrm{n} 1=10, \mathrm{n} 2=4\)

(2) n 1 and n 2 are specified as \(\mathrm{n} 1>\mathrm{n} 2\). When \(\mathrm{n} 1 \leqq 2\), bits are shifted by the amount of the remainder obtained from dividing n 2 by n 1 .
However, no processing is performed if the value of the remainder obtained from dividing n2 by n 1 is 0 .
(3) n 1 is specified within the range from 1 to 64 .
(4) The \(n 2\) bits from the most significant bit become 0 . When \(n 1<n 2\), the bits equal to the amount of the remainder obtained from dividing n 2 by n 1 become 0 .
(5) No processing is performed if 0 is specified for n 1 or n 2 .

\section*{SFTBL(P)}
(1) Shifts the n 1 -bit data from the device specified for © to the left by n 2 bit. When \(n 1=10, n 2=4\)

(2) n 1 and n 2 are specified as \(\mathrm{n} 1>\mathrm{n} 2\). When \(\mathrm{n} 1 \leqq \mathrm{n} 2\), bits are shifted by the amount of the remainder obtained from dividing n 2 by n 1 .
However, no processing is performed if the value of the remainder obtained from dividing n2 by n 1 is 0 .
(3) n 1 is specified within the range from 1 to 64 .
(4) The n 2 bits from the lowest bit become 0 . When \(\mathrm{n} 1<\mathrm{n} 2\), the bits equal to the amount of the remainder obtained from dividing n 2 by n 1 become 0 .
(5) No processing is performed if 0 is specified for n 1 or n 2 .

\section*{Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value specified for n 1 is not within 0 to 64.
(Error code: 4100)
- A negative value is specified for n 2 .
(Error code: 4100)
- The device points specified for n1 exceed the device range specified for (d).
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the data in the devices from Y10 to Y17 (8 bits) specified for (d) are shifted to the right by 2 bits ( n 2 ) when M0 turns ON.
[Structured ladder/FBD]

[ST]
SFTBR(M0,K8,K2,Y10);
[Operation]
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline Y17 & Y16 & Y15 & Y14 & Y13 & Y12 & Y11 & Y10 & \\
\hline 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & \\
\hline Y17 & Y16 & Y15 & Y14 & Y13 & Y12 & Y11 & Y10 & (SM700) \\
\hline 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
\hline
\end{tabular}
(2) In the following program, the data in the devices from Y21 to Y2C (12 bits) specified for (d) are shifted to the left by 5 bits ( n 2 ) when M0 turns ON.
[Structured ladder/FBD]

[ST]
SFTBL(M0,K12,K5,Y21);
[Operation]
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\multicolumn{1}{c}{ Y2C } & Y2B & Y2A & Y29 & Y28 & Y27 & Y26 & Y25 & Y24 & Y23 & Y22 & Y21 \\
\hline 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
\hline
\end{tabular}

Carry flag
(SM700)
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline Y2C & Y2B & Y2A & Y29 & Y28 & Y27 & Y26 & Y25 & Y24 & Y23 & Y22 & Y21 \\
\hline 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\hline
\end{tabular}

\section*{Caution}

When using the SFTBR/SFTBL instruction, do not branch a line from © .
If branched, the operation result is not output to the devices or labels of branch destination correctly.

For details of branch point of line from destination, refer to Section 3.2.

\subsection*{7.3.4 1-word right/left shift of n-word data}

DSFR, DSFL
Basic
Hioh
Process
Redundant
Univers
LCPU


\section*{3 Function}

\section*{DSFR(P)}
(1) Shifts the data \(n\) points from the device specified for (d) to the right by 1 word.

(2) The device specified for (d) \(+(n-1)\) becomes 0 .

\section*{DSFL(P)}
(1) Shifts the data \(n\) points from the device specified for (d) to the left by 1 word.

(2) The device specified for (d) becomes 0 .

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The range of \(n\) points from the device specified for (d) exceeds the corresponding device.
(Error code: 4101)

\section*{Program Example}
(1) In the following program, the data from D683 to D689 are shifted to the right when X0B turns ON.
[Structured ladder/FBD]

[ST]
DSFRP(X0B,7,D683);
[Operation]

(2) In the following program, the data from D683 to D689 are shifted to the left when X0B turns ON.
[Structured ladder/FBD]


\section*{[ST]}

DSFLP(X0B,7,D683);
[Operation]


\section*{Caution}

When using the DSFR/DSFL instruction, do not branch a line from (d).
If branched, the operation result is not output to the devices or labels of branch destination correctly.
For details of branch point of line from destination, refer to Section 3.2.

\subsection*{7.3.5 n-word right/left shift of n-word data}

- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported


Input argument,

Output argument,
\begin{tabular}{ll} 
EN: & Executing condition \\
\(\mathrm{n} 1:\) & Number of words to be shifted \\
\(\mathrm{n} 2:\) & Number of words \\
ENO: & Execution result \\
\(\mathrm{d}:\) & Start number of the device to be shifted
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...)} & \multirow[b]{2}{*}{U"ila} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{\begin{tabular}{l}
Constant \\
K, H
\end{tabular}} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline n1 & - & \(\bigcirc\) & \(\bigcirc\) & \multicolumn{5}{|c|}{\(\bigcirc\)} & - \\
\hline n2 & - & \(\bigcirc\) & \(\bigcirc\) & \multicolumn{5}{|c|}{\(\bigcirc\)} & - \\
\hline (d) & - & \(\bigcirc\) & \(\bigcirc\) & \multicolumn{5}{|c|}{-} & - \\
\hline
\end{tabular}

\section*{Function}

\section*{SFTWR(P)}
(1) Shifts the n1-word data from the device specified for (d) to the right by n 2 word. When \(\mathrm{n} 1=9, \mathrm{n} 2=4\)



Become OH
(2) The n 2 words from the most significant word become 0 .
(3) No processing is performed if 0 is specified for n 1 or n 2 .
(4) When \(\mathrm{n} 1 \leqq \mathrm{n} 2\), all n1-word data from the device specified for (a) become 0 .

\section*{SFTWL(P)}
(1) Shifts the n1-word data from the device specified for © to the right by n 2 word.

When \(n 1=9, n 2=4\)
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline\(\overbrace{\text { (d) }+8}\) & (d) +7 & (d) +6 & (d) +5 & (d) +4 & (d) +3 & (d) +2 & (d) +1 & (d) \\
\hline \(1 \mathrm{FFH}_{\mathrm{H}}\) & 10 H & \(0 H\) & \(7 \mathrm{FFH}_{\mathrm{H}}\) & \(3 \mathrm{~A}_{\mathrm{H}}\) & \(1 \mathrm{FH}_{\mathrm{H}}\) & 30 H & 0 H & \(\mathrm{FFH}^{2}\) \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline (d) +8 & (d) +7 & (d) +6 & (d) +5 & (d) +4 & (d) +3 & (d) +2 & d) +1 & (d) \\
\hline \(3 \mathrm{~A}_{\mathrm{H}}\) & 1FH & 30 H & OH & \(\mathrm{FFH}_{\mathrm{H}}\) & OH & OH & OH & OH \\
\hline
\end{tabular}

Become \(\mathrm{OH}_{\mathrm{H}}\)
(2) The n 2 words from the lowest word becomes 0.
(3) No processing is performed if 0 is specified for n 1 or n 2 .
(4) When \(\mathrm{n} 1 \leqq \mathrm{n} 2\), all n1-word data from the device specified for (d) become 0 .

\section*{0 Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- A negative value is specified for n 1 or n 2 .
(Error code: 4100)
- The device points specified for n 1 exceed the device range specified for (d).
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the 8-word data ( n 1 ) from D10 specified for (d) are shifted to the right by 2 words (n2) when M0 turns ON.
[Structured ladder/FBD]

[ST]
SFTWR(M0,K8,K2,D10);
[Operation]

(2) In the following program, the 12-word data (n1) from D21 specified for (d) are shifted to the left by 5 words ( n 2 ) when M0 turns ON.
[Structured ladder/FBD]

[ST]
SFTWL(M0,K12,K5,D21);
[Operation]


\section*{Caution}

When using the SFTWR/SFTWL instruction, do not branch a line from (d).
If branched, the operation result is not output to the devices or labels of branch destination correctly.

For details of branch point of line from destination, refer to Section 3.2.

\subsection*{7.4 Bit Processing Instructions}

\subsection*{7.4.1 Bit set and reset of word devices}

BSET, BRST

BSET(P)
BRST(P)

> P: Executing condition


Input argument,
Output argument,
\begin{tabular}{ll} 
EN: & Executing condition \\
\(\mathrm{n}:\) & Bit number to be set or reset (0 to 15) \\
ENO: & Execution result \\
\(\mathrm{d}:\) & Device number whose bit is set or rese
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J等)} & \multirow[b]{2}{*}{U...ing} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline n & \multicolumn{7}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) & \multicolumn{7}{|c|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{Function}

\section*{BSET(P)}
(1) Sets (to 1) the nth bit of the word device specified for (d).
(2) If n exceeds 15 , bit set/reset is performed with the lower 4 bits of the data.


\section*{BRST(P)}
(1) Resets (to 0 ) the nth bit of the word device specified for (d).
(2) If \(n\) exceeds 15 , bit set/reset is performed with the lower 4 bits of the data.


\section*{Operation Error}

No operation error occurs in the execution of the BSET \((P)\) and \(\operatorname{BRST}(P)\) instructions.

\section*{Program Example}

In the following program, the 8th bit (b8) of Var_D8 is reset to 0 when XOB turns OFF, and the 3rd bit (b3) of Var_D8 is set to 1 when X0B turns ON.
[Structured ladder/FBD]

[ST]
BRSTP(NOT(XOB),8,Var_D8);
BSETP(X0B,3,Var_D8);
[Operation]


Bit set or reset of word devices can also be performed by bit specification of word device.
For details on bit specification of word device, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used. The processing of the program example would be performed as shown below with the bit specification of word device.


\subsection*{7.4.2 Bit test}

\section*{TEST, DTEST \\ }


Input argument,
\begin{tabular}{ll} 
EN: & Executing condition \\
s1: & Start device number of the device that stores bit data to be
\end{tabular} extracted
s2: Position of bit data to be extracted :ANY16 ( 0 to 15 (TEST)/0 to 31 (DTEST))

Output argument,
ENO: Execution result
d: Bit device number that stores extracted bit data
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J:.10:} & \multirow[b]{2}{*}{U...igata} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & \multicolumn{6}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - & - \\
\hline (s2) & \multicolumn{6}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & \(\bigcirc\) & - \\
\hline (d) & \multicolumn{6}{|c|}{\(\bigcirc\)} & - & - & - \\
\hline
\end{tabular}

\section*{\(\sqrt{3}\) Function}

\section*{TEST(P)}
(1) Fetches bit data from the location specified for \((2)\) within the word device specified for (s1), and writes it to the bit device specified for (d).
(2) The bit device specified for (d) is turned OFF when the corresponding bit is 0 and ON when it is 1 .
\begin{tabular}{|ll|}
\hline & \\
\hline instructions. & indicates any of the following \\
TEST & TESTP \\
DTEST & DTESTP \\
\\
\\
\\
\\
\hline
\end{tabular}
:Bit
:ANY16/32
:Bit
:Bit
 Others
\(\qquad\)
(3) The position specified for (s2) specifies the position of each bit in a 1-word data block ( 0 to 15). When 16 or higher value is specified for \(\S_{2}\), the target is the bit data at the position indicated by the remainder of \(n\) divided by 16 . For example, when \(n=18\), the remainder of \(18 / 16=1\) is 2 , thus the target is the data of b2.


\section*{DTEST(P)}
(1) Fetches bit data from the location specified for \(\circledR_{2}\) ) within the 2-word device specified for (s1), and writes it to the bit device specified for (d).
(2) The bit device specified for (d) is turned OFF when the corresponding bit is 0 and ON when it is 1 .
(3) The position specified for ©2 specifies the position of each bit in a 2-word data block ( 0 to 31). When 32 or higher value is specified for \(\Omega_{2}\), the target is the bit data at the position indicated by the remainder of \(n\) divided by 32 . For example, when \(n=34\), the remainder of \(34 / 32=1\) is 2 , thus the target is the data of b2.


\section*{O Operation Error}

No operation error occurs in the execution of the TEST(P) and DTEST(P) instructions.

\section*{\(\triangle\) Program Example}
(1) In the following program, Var_M0 is turned ON or OFF depending on the status of the 10th bit of the 1-word data block (Var_D0).
[Structured ladder/FBD]


\section*{[ST]}

TESTP(SM400,Var_D0,10,Var_M0);
[Operation]


(2) In the following program, Var_M0 is turned ON or OFF, depending on the status of the 19th bit of the 2-word data (Var_W0).
[Structured ladder/FBD]

[ST]
DTESTP(SM400,Var_W0,19,Var_M0);
[Operation]


\section*{Remark}

Programs using the bit test instruction can be rewritten as programs using bit specification of word device.
If the program were changed to use bit specification of word device, it would appear as follows.


M0 turns ON/OFF depending on the ON/OFF status of b10 of D0 (D0.A).

\subsection*{7.4.3 Batch reset of bit devices}

BKRST

\section*{Basic}

High
Process
Resundant
Universs
LCPU


\section*{5 Function}
(1) Resets bit devices \(n\) points from the bit device specified for (s).
\begin{tabular}{|c|c|}
\hline Device & Status \\
\hline Annunciator (F) & \begin{tabular}{l}
- Turns OFF bit devices n points from annunciator (F) number specified for (s). \\
- Deletes annunciator number turned OFF from SD64 to SD79 and compresses remaining data forward. \\
- Stores annunciators stored in SD64 to SD79, to SD63.
\end{tabular} \\
\hline \[
\begin{gathered}
\text { Timer (T) } \\
\text { Counter (C) }
\end{gathered}
\] & - Sets 0 for the current value \(n\) points from the timer ( \(T\) ) or counter (C) number specified for (S), and turns the coil or contact OFF. \\
\hline Bit devices other than the above & - Turns OFF coil or contact \(n\) points from the device specified for © S . \\
\hline
\end{tabular}
(2) If the specified device is OFF, the device status will not change.

\section*{Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The range of \(n\) points from the device specified for (s) exceeds the corresponding device.
(Error code: 4101)

\section*{\(\triangle\) Program Example}
(1) In the following program, the devices from M 0 to M 7 are turned OFF when X 0 turns ON . [Structured ladder/FBD]


\section*{[ST]}

BKRSTP(X0,M0,8);
[Operation]
M9M8M7---M4M3- --M0
1:1 1100:1 1:1100
1100
Not changed
(2) In the following program, the data from 2nd bit (b2) of D10 to 1 st bit (b1) of D11 are set to 0 when X20 turns ON.
[Structured ladder/FBD]

[ST]
BKRSTP(X20,D10.2,16);
[Operation]





\subsection*{7.5 Data Processing Instructions}

\subsection*{7.5.1 16-/32-bit data search}

SER, DSER

SER(P)
DSER(P)

> P: Executing condition
\(\pm\)


instructions.
SER SERP

Input argument,

Output argument,
\begin{tabular}{ll} 
EN: & Executing condition \\
s1: & Search data \\
s2: & Data to be searched \\
\(\mathrm{n}:\) & Number of searches \\
ENO: & Execution result \\
\(\mathrm{d}:\) & Start number of the device that stores the search result
\end{tabular}
:Bit
:ANY16/32
:ANY16/32
:ANY16
:Bit
:Array of ANY16 (0..1)
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{} & \multirow[t]{2}{*}{U...igal} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{\begin{tabular}{l}
Constant \\
K,H
\end{tabular}} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & \(\bigcirc\) & & & \(\bigcirc\) & & \(\bigcirc\) & & \(\bigcirc\) & - \\
\hline (32) & - & & & - & & - & & - & - \\
\hline n & \(\bigcirc\) & & & \(\bigcirc\) & & \(\bigcirc\) & & \(\bigcirc\) & - \\
\hline (d) & - & & & - & & \(\bigcirc\) & & - & - \\
\hline
\end{tabular}

\section*{\(\sqrt{3}\) Function}

\section*{SER(P)}
(1) Searches \(n\) points from the 16-bit data of the device specified for \({ }_{(\Omega 2}\), regarding 16 -bit data of the device specified for \((51)\) as a keyword. Then, the number of matches with the keyword is stored to the device specified for © [1], and the relative value from ©2) to the first matched device number is stored to the device specified for © [0].

(2) No processing is performed if n is 0 or a negative value.
(3) If no matches are found in the search, the device specified for (d) becomes 0 .

\section*{DSER(P)}
(1) Searches n points from the device specified for \({ }_{(22}\) in units of 32 bits ( \(2 \times \mathrm{n}\) points in units of 16 bits) regarding 32-bit data of the device specified for \((51)\) as a keyword. Then, the number of matches with the keyword is stored to the device specified for © [1], and the relative value from \({ }_{(22}\) to the first matched device number is stored to the device specified for © [0].

(2) No processing is performed if n is 0 or a negative value.
(3) If no matches are found in the search, the device specified for © becomes 0 .

\section*{XPOINT}

If the data to be searched using the \(\operatorname{SER}(\mathrm{P})\) or \(\operatorname{DSER}(\mathrm{P})\) instruction is sorted in the ascending order, searches can be accelerated by the use of the binary search method, which is activated by turning SM702 *1 ON. However, correct search results are not obtained if SM702 is turned ON when the data to be searched are not sorted in the ascending order.

\footnotetext{
*1: SM702 is the special relay for setting the search method.
}
- SM702 OFF: Linear search method (linear search method) (Comparison with the search data starts from the beginning of the data to be searched.)
- SM702 ON: Binary search method (Obtains the center value of the sorted array and decides if the obtained value is larger or smaller than the search value, then, chooses the area for search between the larger and smaller value divisions. By repeating this process, the area for search is narrowed down.)


\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The n points from the device \(\S_{2}\) exceeds the specified device range. (Error code: 4101)
- The device specified for (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\triangle\) Program Example}
(1) In the following program, the devices from D100 to D105 are searched by the data in Var_D0 when X20 turns ON, and the results are stored to Var_W0.
[Structured ladder/FBD]


\section*{[ST]}

Var_D0:=123;
SERP(X20,Var_D0,D100,6,Var_W0);
[Operation]
\begin{tabular}{cc} 
& \multicolumn{1}{c}{ Search data } \\
\cline { 2 - 3 } & 123 \\
\cline { 2 - 3 }
\end{tabular}

Data to be searched

(2) In the following program, the devices from D100 to D111 are searched by the data in Var_D10 when X20 turns ON, and the results are stored to Var_W0.
[Structured ladder/FBD]

[ST]
Var_D10:=56789051;
DSERP(X20,Var_D10,D100,6,Var_W0);
[Operation]


\subsection*{7.5.2 16-/32-bit data bit check}


\section*{Function}

\section*{SUM \((P)\)}

From the 16-bit data in the device specified for (s), stores the total number of bits set to 1 , to the device specified for (d).


Stores the total number of bits where 1 is set in BIN data.
(There are 8 bits where 1 is set in the example.)

\section*{DSUM(P)}

From the 32-bit data in the device specified for ©s, stores the total number of bits where 1 is set, to the device specified for (d).


Stores the total number of bits where 1 is set in BIN data.
(There are 16 bits where 1 is set in the example.)

\section*{0 Operation Error}

No operation error occurs in the execution of the \(\operatorname{SUM}(P)\) and \(\operatorname{DSUM}(P)\) instructions.

\section*{\(\triangle\) Program Example}
(1) In the following program, the number of bits which are turned ON in the devices from X 8 to X17 are stored to Var_D0 when X10 turns ON.
[Structured ladder/FBD]

[ST]
SUMP(X10,K4X8,D0);

\section*{[Operation]}

X17---------------------- X8

Stores the total number of bits where 1 is set to Var_D0.

(2) In the following program, the number of bits which are turned ON in Var_D100 are stored to Var_D0 when X10 turns ON.
[Structured ladder/FBD]

[ST]
DSUMP(X10,Var_D100,Var_D0);
[Operation]

Stores the total number of bits where 1 is set to DO.
Var_D0 \(\quad\)\begin{tabular}{c} 
D \\
\\
\\
\hline
\end{tabular}

\subsection*{7.5.3 Decoding from 8 to 256 bits}

DECO (P)
P: Executing condition

indicates any of the following instructions.
DECO DECOP
\begin{tabular}{lll} 
Input argument, & \(\mathrm{EN}:\) & \begin{tabular}{l} 
Executing condition \\
\(\mathrm{s}:\)
\end{tabular} \\
& \(\mathrm{n}:\) & \begin{tabular}{l} 
Decoding data, or start number of the device that stores \\
decoding data
\end{tabular} \\
Output arg bit length (1 to 8), no processing on 0
\end{tabular}
:Bit
:ANY_SIMPLE
:ANY16
:Bit
:ANY_SIMPLE
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J.....!} & \multirow[t]{2}{*}{U...ic...!} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{\begin{tabular}{l}
Constant \\
K, H
\end{tabular}} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & \multicolumn{3}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline n & \multicolumn{3}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) & \multicolumn{3}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Turns ON the bit position of © , which corresponds to the binary value specified by the lower \(n\) bits of (s).

(2) The value of n can be specified between 1 and 8 .
(3) If \(\mathrm{n}=0\), there will be no operation, and the content of (d) will not change.
(4) Bit devices are treated as 1 bit, and word devices as 16 bits.
(5) More than n bits are required for (s).

When using a label, specify the data type of \(n\) bits, or array type label of \(n\) bits.
(6) More than \(2^{n}\) bits are required for (d).

When using a label, specify the data type of \(2^{n}\) bits, or array type label of \(2^{n}\) bits.

\section*{Caution}

The devices set as array type labels specified for (s) and (d) are used for the operation regardless of whether the required device points are reserved or not.

Therefore, when the devices which are out of the range of device points set as array type labels are used for another operation, an operation error occurs and it may cause malfunction of the CPU module.

\section*{Example}


As shown above, the binary value including the devices of the 3rd bit (M12) is used because 3 bits are required for (s).

The next device and the following devices (M4 to M7) are used for the 4th bit to the 7th because 8 bits are required for (d).

When 6 is specified for the binary value used for (s) as shown below, g_bool 1 turns ON because M6 turns ON.


In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of n is outside the range of 0 to 8 .
- The range of \(2^{n}\) bits from (d) exceeds the corresponding device.

\section*{\(\square\) Program Example}
(1) In the following program, the 3 bits from g_bool2 (X0) are decoded when g_bool1 turns ON, and the results are stored to g_bool3 (M10) and the following devices.
[Structured ladder/FBD]

[ST]
DECOP(g_bool1,g_bool2,3,g_bool3);
[Operation]


If 3 bits are specified as significant bits, 8 points are occupied.
(2) In the following program, the lower 3 bits of g_int1 (D0) are decoded when g_bool1 turns ON, and the results are stored to g_int2 (D10) and the following devices.
[Structured ladder/FBD]


\section*{[ST]}

DECOP(g_bool1,g_int1,3,g_int2);
[Operation]

(3) In the following program, the 3 bits from g_boolArray1[0] are decoded when g_bool1 turns ON, and the results are stored to g_boolArray2[0] and the following devices.
The array label settings are as follows.
\begin{tabular}{|l|l|l|l|}
\hline Class & Label & Data type & Device \\
\hline VAR_GLOBAL & g_boolArray1 & Bit(0..2) & X0 \\
\hline VAR_GLOBAL & g_boolArray2 & Bit(0..7) & M10 \\
\hline
\end{tabular}
[Structured ladder/FBD]

[ST]
DECOP(g_bool1,g_boolArray1[0],3,g_boolArray2[0]);
[Operation]


\subsection*{7.5.4 Encoding from 256 to 8 bits}

ENCO

\(\mathrm{ENCO}(\mathrm{P}) \quad(\mathrm{P}:\) Executing condition \(\quad: \pm\),

\begin{tabular}{lll} 
Input argument, & EN: & Executing condition \\
& \(\mathrm{s}:\) & Start number of the device that stores encoding data \\
& \(\mathrm{n}:\) & Valid bit length (1 to 8), no processing on 0 \\
Output argument, & ENO: & Execution result \\
& \(\mathrm{d}:\) & Start number of the device that stores encoding result
\end{tabular}
:Bit
:ANY_SIMPLE
:ANY16
:Bit
:ANY16
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J\%1} & \multirow[b]{2}{*}{U} & \multirow[b]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & \multicolumn{3}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline n & \multicolumn{3}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) & \multicolumn{3}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{\(\widehat{3}\) Function}
(1) Stores the binary value corresponding to the bits set to 1 in the \(2^{n}\)-bit data of © to (d).

(2) The value of n can be specified between 1 and 8 .
(3) If \(\mathrm{n}=0\), there will be no operation, and the content of (d) will not change.
(4) Bit devices are treated as 1 bit, and word devices as 16 bits.
(5) If more than 1 bit is set to 1 , processing will be performed at the upper bit position.
(6) More than \(2^{n}\) bits are required for (s).

When using a label, use the data type that can reserve \(2^{n}\) bits, or reserve \(2^{n}\) bits or more for (d) or later devices by using array.

For data type that can be used for (s) and the number of array elements by the value of \(n\), refer to the following table.
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline & \multicolumn{9}{|c|}{Data types that can be specified for \(\boldsymbol{s}\) and number of array elements} \\
\hline \begin{tabular}{l}
Value specified \\
n
\end{tabular} & Bit & Word (signed) & Double word (signed) & Word (unsigned)/ 16-bit string & Double word (unsigned)/ 32-bit string & Singleprecision real & Doubleprecision real & String (32 characters) & Time \\
\hline 1 & 1 point & 1 point & 1 point & 1 point & 1 point & 1 point & 1 point & 1 point & 1 point \\
\hline 2 & 4 points & 1 point & 1 point & 1 point & 1 point & 1 point & 1 point & 1 point & 1 point \\
\hline 3 & 8 points & 1 point & 1 point & 1 point & 1 point & 1 point & 1 point & 1 point & 1 point \\
\hline 4 & 16 points & 1 point & 1 point & 1 point & 1 point & 1 point & 1 point & 1 point & 1 point \\
\hline 5 & 32 points & 2 points & 1 point & 2 points & 1 point & 1 point & 1 point & 1 point & 1 point \\
\hline 6 & 64 points & 4 points & 2 points & 4 points & 2 points & 2 points & 1 point & 1 point & 2 points \\
\hline 7 & 128 points & 8 points & 4 points & 8 points & 4 points & 4 points & 2 points & 1 point & 4 points \\
\hline 8 & 256 points & 16 points & 8 points & 16 points & 8 points & 8 points & 4 points & 2 points & 8 points \\
\hline
\end{tabular}

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of n is outside the range of 0 to 8 . (Error code: 4100)
- The range of \(2^{n}\) bits from (s) exceeds the corresponding device. (Error code: 4101)
- All data of \(2^{n}\) bits from (s) are 0 .
(Error code: 4100)

\section*{\(\triangle\) Program Example}
(1) In the following program, the 3 bits from M10 are encoded when X 20 turns ON , and the results are stored to Var_D8.
[Structured ladder/FBD]

[ST]
ENCOP(X20,M10,3,Var_D8);

\section*{[Operation]}


The location of the ON bit, counted from M10, is stored in BIN data.
(2) Example of the program using an array label

The 3 bits from L_ArrayENCO_S[0] are encoded when X20 turns ON, and the results are stored to L_ENCO_D.
[Structured ladder/FBD]

[ST]
ENCOP(X20,L_ArrayENCO_S[0],3,L_ENCO_D);
(3) Example of the program using a label

The lower 3 bits of L_WORD1 are encoded when X20 turns ON, and the results are stored to L_WORD2.
[Structured ladder/FBD]

[ST]
ENCOP(X20,L_WORD1,3,L_WORD2);

\subsection*{7.5.5 7-segment decode}

SEG


\section*{Function}
(1) Decodes the data from 0 to F specified for the lower 4 bits of © to 7 -segment display data, and stores to (d).
(2) If (d) is a bit device, indicates the start number of the devices storing the 7-segment display data, and if it is a word device, indicates the number of the device storing the data.


\section*{Operation Error}

No operation error occurs in the execution of the \(\operatorname{SEG}(P)\) instruction．
7－segment decode display
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multicolumn{2}{|r|}{©} & \multirow[b]{2}{*}{Configuration of 7 segments} & & & & & & & & & \multirow[b]{2}{*}{Display data} \\
\hline Hexade cimal & Bit pattern & & B7 & B6 & B5 & B4 & B3 & B2 & B1 & B0 & \\
\hline 0 & 0000 & \multirow[t]{5}{*}{} & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & ！ \\
\hline 1 & 0001 & & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & I \\
\hline 2 & 0010 & & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & こ \\
\hline 3 & 0011 & & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & \(\bar{\square}\) \\
\hline 4 & 0100 & & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
\hline 5 & 0101 & & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & E \\
\hline 6 & 0110 & \multirow[t]{10}{*}{} & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & E \\
\hline 7 & 0111 & & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
\hline 8 & 1000 & & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 晶 \\
\hline 9 & 1001 & & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 曰 \\
\hline A & 1010 & & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 晰 \\
\hline B & 1011 & & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & ！ \\
\hline c & 1100 & & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & \(\stackrel{1}{-}_{1}^{1}\) \\
\hline D & 1101 & & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & －1 \\
\hline E & 1110 & & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & E \\
\hline F & 1111 & & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & I－ \\
\hline
\end{tabular}

Start number of bit device
Lowest bit of word device

\section*{\(\square\) Program Example}

In the following program, the data from XC to XF are converted to 7-segment display data and the results are output to the devices from Y38 to Y3F when X0 turns ON.
[Structured ladder/FBD]


\section*{[ST]}

SEGP(X0,K1X0C,K2Y38);
[Timing chart]


\subsection*{7.5.6 4-bit separation of 16-bit data}

DIS

\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{} \\
\hline instr & \\
\hline DIS & DISP \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{3}{*}{Input argument,} & EN: & \multicolumn{5}{|l|}{Executing condition} & \multicolumn{4}{|l|}{:Bit} \\
\hline & s : & \multicolumn{5}{|l|}{Start number of the device that stores data to be separated} & \multicolumn{4}{|l|}{:ANY16} \\
\hline & n : & \multicolumn{5}{|l|}{Number of separations (1 to 4), no processing on 0} & \multicolumn{4}{|l|}{:ANY16} \\
\hline \multirow[t]{7}{*}{Output argument,} & ENO: & \multicolumn{5}{|l|}{Execution result} & \multicolumn{4}{|l|}{:Bit} \\
\hline & d: & \multicolumn{5}{|l|}{Start number of the device that stores separated data} & \multicolumn{4}{|l|}{:ANY16} \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...alat} & \multirow{2}{*}{U..igan:} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (5) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{\(\bigcirc\)} & & - \\
\hline & n & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{\(\bigcirc\)} & & - \\
\hline & (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & - & & & - \\
\hline
\end{tabular}

\section*{Function}
(1) Stores the lower \(n\) digits (1 digit is 4 bits) of the 16-bit data specified for ©s to the lower 4 bits \(n\) points from the device specified for (d).

(2) The upper 12 bits n points from the device specified for (a) become 0 .
(3) The value of n can be specified between 1 and 4.
(4) If \(\mathrm{n}=0\), there will be no processing, and the content of n points from (d) will not change.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The range of \(n\) points from the device specified for (d) exceeds the corresponding device.
(Error code: 4101)
- The value of n is outside the range of 0 to 4 .
(Error code: 4100)

\section*{\(\triangle\) Program Example}

In the following program, the 16-bit data in Var_D0 are separated into 4-bit groups, and the results are stored to the devices from D10 to D13 when X0 turns ON.
[Structured ladder/FBD]


\section*{[ST]}

DISP(X0,Var_D0,4,D10);
[Operation]


\section*{Caution}

When using the DIS instruction, do not branch a line from (d).
If branched, the operation result is not output to the devices or labels of branch destination correctly.

For details of branch point of line from destination, refer to Section 3.2.

\subsection*{7.5.7 4-bit connection of 16-bit data}


\section*{Function}
(1) Connects lower 4 bits of 16-bit data \(n\) points from device specified for ©s to 16 -bit device specified for (d).

(2) The bits of the upper \((4-n\).) digits of the device specified for (d) become 0.
(3) The value of \(n\) can be specified between 1 and 4.
(4) If \(\mathrm{n}=0\), there will be no processing, and the content of device (d) will not change.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The range of \(n\) points from the device specified for (s) exceeds the corresponding device.
(Error code: 4101)
- The value of \(n\) is outside the range of 0 to 4 .
(Error code: 4100)

\section*{\(\square\) Program Example}

In the following program, the lower 4 bits of the devices from D0 to D2 are connected when X0 turns ON, and they are stored to Var_D10.
[Structured ladder/FBD]


\section*{[ST]}

UNIP(X0,D0,3,Var_D10);

\section*{[Operation]}


\subsection*{7.5.8 Separation and connection of random data}

NDIS, NUNI

\begin{tabular}{lll} 
Input argument, & EN: & Executing condition \\
s1: & \begin{tabular}{l} 
Start number of the device that stores data to be separated or :ANY16 \\
connected
\end{tabular} \\
O2: & \begin{tabular}{l} 
Start number of the device that stores units of separations and :ANY16 \\
connections
\end{tabular} \\
Output argument, & ENO: & \begin{tabular}{l} 
Execution result \\
d:
\end{tabular} \\
& \begin{tabular}{l} 
Start number of the device that stores separated or connected :ANY16 \\
data
\end{tabular}
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{} & \multirow[b]{2}{*}{U......} & \multirow[t]{2}{*}{Zn} & \multirow[b]{2}{*}{Constant} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & - & & & & & & & & \\
\hline (2) & - & & & & & & & & \\
\hline (d) & - & & & & & & & & \\
\hline
\end{tabular}

\section*{I Function}

\section*{NDIS(P)}
(1) Separates each bit of data stored in the device number specified for (51) and the following devices in blocks whose numbers of bits are specified for (s2), and stores the data to the device number specified for (d) and the following devices.

(2) The number of separated bits specified for © \({ }^{2}\) can be specified within a range of 1 to 16 bits.
(3) Bits from the device number specified for \(\S_{2}\) to the device number where 0 is stored are processed as separated bits.
(4) Do not overlap the device range for data to be separated (①) to end range of (51) ) with the device range which stores the separated data (d) to end range of (d)). If overlapped, the correct operation result may not be obtained.
(5) Do not specify the same device number for (51), (22), and (d). If the same device is specified for (1), (s2), and (d), the operation does not perform correctly.

\section*{NUNI(P)}
(1) Connects each bit of data stored in the device number specified for (51) and the following devices in blocks whose numbers of bits are specified for \(\Omega_{2}\), and stores them to the device number specified for (d) and the following devices.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|r|}{Specification of the number of connected bits} \\
\hline (22) & 6 & \\
\hline (52) +1 & 8 & \\
\hline (32) +2 & 6 & \\
\hline (52) +3 & 4 & \\
\hline (32) +4 & 8 & \\
\hline (52) +5 & 10 & \\
\hline (22) +6 & 3 & \\
\hline (22) +7 & 0 & Specification of the end of setting \\
\hline
\end{tabular}

(2) The number of connected bits specified for ©s can be within a range of from 1 to 16 bits.
(3) Processing will be performed on the number of bits to be connected from the device number specified for © \({ }^{2}\) to the device number storing 0 .
(4) Do not overlap the device range for data to be connected (①) to end range of (①) ) with the device range which stores the connected data (d) to end range of (d)). If overlapped, the correct operation result may not be obtained.
(5) Do not specify the same device number for (51), (22), and (d). If the same device is specified for (①), (52), and (d), the operation does not perform correctly.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The number of separated or connected bits specified for \((2)\), or the device use range specified for (s1) or (d) exceeds the final device number of their respective devices.
(Error code: 4101)
- The number of separated or connected bits specified for \(\S_{2}\) is not set within the range from 1 to 16 bits.
(Error code: 4100)

\section*{\(\square\) Program Example}
(1) In the following program, the data of 4, 3, and 6 bits respectively from the lower bits of Var_D0 are separated, and they are stored to the devices from D10 to D12.
[Structured ladder/FBD]

```

[ST]
MOVP(SM400,4,D20);
MOVP(SM400,3,D21);
MOVP(SM400,6,D22);
MOVP(SM400,0,D23);
NDISP(SM400,Var_D0,D20,D10);

```
[Operation]

(2) In the following program, the lower 4 bits of data in D10, the lower 3 bits of data in D11, and the lower 6 bits of data in D12 are connected, and they are stored to Var_D0.
[Structured ladder/FBD]


\section*{[ST]}

MOVP(SM400,4,D20);
MOVP(SM400,3,D21);
MOVP(SM400,6,D22);
MOVP(SM400,0,D23);
NUNIP(SM400,D10,D20,Var_D0);

\section*{[Operation]}


\subsection*{7.5.9 Separation and connection of data in units of bytes}
WTOB, BTOW
Basic

\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{\multirow[t]{2}{*}{\(\square\) indicates any of the following instructions.}} \\
\hline & \\
\hline WTOB & WTOBP \\
\hline BTOW & BTOWP \\
\hline
\end{tabular}

Input argument,
\begin{tabular}{lll} 
EN: & Executing condition & \(:\) Bit \\
\(\mathrm{s}:\) & Start number of the device that stores data to be separated or :ANY16 \\
& connected in unit of bytes \\
\(\mathrm{n}:\) & Number of byte data to be separated or connected & :ANY16 \\
ENO: & Execution result & :Bit \\
\(\mathrm{d}:\) & \begin{tabular}{l} 
Start number of the device that stores the separation or \\
connection result in unit of bytes
\end{tabular} & \(:\) ANY16
\end{tabular}


\section*{\(\sqrt{3}\) Function}

\section*{WTOB(P)}
(1) Separates \(n\) bytes of the 16-bit data stored in the device specified for ©s and the following devices, and stores them to the device specified for (d) and the following devices.


For example, if \(n=5\), data through the lower 8 bits of (s) to \((\) (s) +2\()\) would be stored to (d) to (d) +4 ).

(2) Setting the number of bytes with \(n\) automatically determines the range of the 16-bit data specified for © and the range of the devices to store the byte data specified for © (
(3) No processing will be performed when the number of bytes specified for n is 0 .
(4) The 00 H code will automatically be stored to the upper 8 bits of the byte storage device specified for © .

(5) Separation is correctly processed even when the device range (© to ©s \(+(n-1)\) ) where the data to be separated is stored overlaps with the device range (d to © \(+\left(\frac{n}{2}-1\right)\) ) where the separated data will be stored.

\section*{BTOW(P)}
(1) Connects the lower 8 bits of the 16-bit data in \(n\) words stored in the device specified for ©s and the following devices in units of 1-word and stores it to the device specified for (d) and the following devices. The upper 8 bits of n-word data stored in the device specified for (s) and the following devices will be ignored. Further, if n is an odd number, 0 is stored to the upper 8 bits of the device where the nth byte data are stored.


For example, if \(\mathrm{n}=5\), the lower 8 -bit data from (s) to (ⓢ+4) are connected and stored to (d) to ( \((1+2)\).

(2) Setting the number of bytes with n automatically determines the range of the byte data specified for (s) and the range of the devices to store the connected data specified for (d).
(3) No processing will be performed when the number of bytes specified for n is 0 .
(4) The upper 8 bits of the byte data storage device specified for (s) are ignored, and the lower 8 bits are used.
(5) Connection is correctly processed even when the device range (© to ©s \(+(n-1)\) ) where the data to be connected is stored overlaps with the device range (d to © \(+\left(\frac{n}{2}-1\right)\) ) where the connected data will be stored.

For example, the following will take place in a case where the lower 8 bits of D11 to D16 are to be stored to D12 to D14.


\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The range of the number of bytes specified for \(n\) following the device number specified for (s) exceeds the corresponding device range.
(Error code: 4101)
- The range of the number of bytes specified for \(n\) following the device number specified for (d) exceeds the corresponding device range.
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the data from D10 to D12 are separated in units of bytes when X0 turns ON, and they are stored to the devices from D20 to D25.
[Structured ladder/FBD]

[ST]
WTOBP(X0,D10,6,D20);
[Operation]

(2) In the following program, the lower 8-bit data from D20 to D25 are connected when X0 turns ON, and they are stored to the devices from D10 to D12.
[Structured ladder/FBD]

[ST]
BTOWP(X0,D20,6,D10);
[Operation]


\subsection*{7.5.10 Maximum value search of \(16-/ 32\)-bit data}

\author{
MAX, DMAX \\ Basic \\ Process \\ Redundant \\ Univers \\ LCPU
}


\section*{Function}

\section*{MAX(P)}
(1) Searches for the maximum value from the \(n\) points of 16 -bit BIN data in the device specified for (s), and stores the searched maximum value to the device specified for (d). Searches from the device specified for (s) and counted from (s), stores the location of device number which is stored in the maximum value of the first search to © +1 and stores the number of maximum values to (d) +2 .


\section*{DMAX(P)}
(1) Searches for the maximum value in the \(n\) points of 32 -bit BIN data, from the device specified for (s), and stores the searched maximum value to the device specified for (d) and (d) +1 . Searches from the device specified for (s) and counted from (s), stores the location of device number which is stored in the maximum value of the first search to (d) +2 and stores the number of maximum values to (d) +3 .


\section*{OOperation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The range of \(n\) points from the device specified for ©s exceeds the corresponding device.
(Error code: 4101)
- The device specified for (d) exceeds the corresponding device range.
(For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\triangle\) Program Example}
(1) In the following program, the data from D100 to D103 are subtracted from the data from R0 to R3 when X1C turns ON, and the maximum value is searched in the subtraction results, then the search results are stored to the devices from D200 to D202.
[Structured ladder/FBD]


\section*{[ST]}

IF X1C THEN
DO:=4;
D150:=D100-R0;
D151:=D101-R1;
D152:=D102-R2;
D153:=D103-R3;
MAXP(TRUE,D150,D0,D200);
END_IF;
[Operation]

(2) In the following program, the maximum value is searched from 32-bit data in the data 4 points from Var_D0 and stores the result to Var_D100 when X20 turns ON.
[Structured ladder/FBD]

[ST]
DMAXP(X20,Var_D0,4,Var_D100);
[Operation]


\subsection*{7.5.11 Minimum value search of \(16-/ 32\)-bit data}
\(\operatorname{MIN}(P)\)
DMIN(P)
P: Executing condition

\begin{tabular}{|lc|}
\hline & \\
\hline & indicates any of the following \\
instructions. & I \\
\begin{tabular}{ll} 
MN \\
DMIN
\end{tabular} & \begin{tabular}{l} 
MINP \\
DMINP
\end{tabular} \\
\hline
\end{tabular}

Input argument,
\begin{tabular}{llll} 
EN: & Executing condition & \(:\) Bit \\
\(\mathrm{s}:\) & Start number of the device in which the minimum value is & \(:\) ANY16/32 \\
\(\mathrm{n}:\) & searched & Number of search data & \(:\) ANY16 \\
ENO: & Execution result \\
\(\mathrm{d}:\) & \begin{tabular}{l} 
Start number of the device that stores the search result of the \\
minimum value
\end{tabular} & \(:\) ANY16/32
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...1.} & \multirow[b]{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{-} & - \\
\hline n & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{\(\bigcirc\)} & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{-} & - \\
\hline
\end{tabular}

\section*{Function}

\section*{\(\operatorname{MIN}(P)\)}
(1) Searches for the minimum value in the \(n\) points of 16 -bit BIN data, from the device specified for © , and stores searched minimum value to the device specified for (d). Searches from the device specified for (s) and counted from (s), stores the location of device number which is stored in the minimum value of the first search to © +1 and stores the number of minimum values to (d) +2 .


\section*{DMIN(P)}
(1) Searches for the minimum value in the n points of 32 -bit BIN data, from the device specified for (s), and stores searched minimum value to the devices specified for (d) and © \({ }^{(d)}+1\). Searches from the device specified for (s) and counted from (s), stores the location of the device number which is stored in the minimum value of the first search to (d) +2 and stores the number of minimum values to (d) +3 .


\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The range of \(n\) points from the device specified for © exceeds the corresponding device.
(Error code: 4101)
- The device specified for (d) exceeds the corresponding device range.
(For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\triangle\) Program Example}
(1) In the following program, the data from D100 to D103 and the data from R0 to R3 are added when g_bool1 turns ON, and the minimum value is searched in the addition result, then the search results are stored to the devices from D200 to D202.
[Structured ladder/FBD]

[ST]
IF g_bool1THEN
g_int1:=4;
D150:=D100+R0;
D151:=D101+R1;
D152:=D102+R2;
D153:=D103+R3;
MINP(TRUE,D150,g_int1,D200);
END_IF;

\section*{[Operation]}


DO \(\qquad\)


D200 \(\square\) Minimum value Location
Quantity
(2) In the following program, the minimum value is searched from 32-bit data in the data 4 points from Var_D0 (D0) and stores the result to Var_D100 (D100) when X20 turns ON.
[Structured ladder/FBD]

[ST]
DMINP(X20,Var_D0,4,Var_D100);
[Operation]
Var_D0 \begin{tabular}{|r|}
\hline \(57020175(\mathrm{BIN})\) \\
\hline \(2070166(\mathrm{BIN})\) \\
\hline \(3596045(\mathrm{BIN})\) \\
\hline\(-69386(\mathrm{BIN})\) \\
\hline
\end{tabular}


\subsection*{7.5.12 Sorting 16-/32-bit data}

SORT, DSORT

SORT
DSORT


Input argument,

Output argument,
\begin{tabular}{ll} 
EN: & Executing condition \\
s1: & Start number of table to be sorted \\
\(\mathrm{n}:\) & Number of sort data \\
s2: & Number of data to be compared in one sort execution \\
ENO: & Execution result \\
d1: & Bit device number to be turned on at sort completion \\
d2: & Device for system use
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J\%):} & \multirow[t]{2}{*}{U...ici..:} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{\begin{tabular}{l}
Constant \\
K, H
\end{tabular}} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & & - \\
\hline n & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & & - \\
\hline (2) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & & - \\
\hline (11) & \(\bigcirc\) & \multicolumn{2}{|c|}{-} & \multicolumn{4}{|c|}{-} & & - \\
\hline (12) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & & - \\
\hline
\end{tabular}

\section*{SORT}
(1) Sorts 16-bit BIN data \(n\) points from \({ }^{(51)}\) in ascending or descending order. Sort order is specified by the ON or OFF status of SM703.
- When SM703 is OFF : Sorts in ascending order
- When SM703 is ON : Sorts in descending order

(2) Several scans are required for sorts performed by the SORT instruction. The number of scans executed until completion is the value obtained by dividing the maximum number of times executed until the completion of the sort by the number of data blocks compared at one execution specified for \(\mathrm{S}_{2}\). (Fractional part is rounded up.) When the value of \(\mathrm{S}_{2}\) ) is increased, the number of scans until completion of the sort is reduced, but the amount of time per scan is lengthened.
(3) The maximum number of executions until completion of the sort should be calculated according to the following equation

Maximum number of executions until completion \(=(n) \times(n-1) / 2\) [times executed]

\section*{Example}

When \(\mathrm{n}=10\), the number of executions is obtained as \(10 \times(10-1) / 2=45\) [times executed].
If \((52)=2\), then the number of scans until the completion of sort is calculated as
\(45 / 2=22.5 \rightarrow 23\) [scans].
(4) The device specified for © (the completion device) is turned OFF by the execution of the SORT instruction, and turned ON when the sort is completed. Because the device specified for (d1) is retained in the ON state after the completion of the sort, the user must turn it OFF if required.
(5) The 2 points from the device specified for (d2) are used by the system during the execution of the SORT instruction. These 2 points from the device specified for (d2) should therefore not be used by the user.
Changing these points may cause an error code to be returned.
(Error code: 4100)
(6) If the value of \(n\) is changed during the execution of the SORT instruction, the sort will be performed in accordance with the number of sort data blocks after the change.
(7) If the execution command is turned OFF during the execution of the SORT instruction, the sort is suspended. The sort resumes from the beginning when the execution command is turned ON again.
(8) To execute another sort operation immediately after the completion of the previous sort, turn OFF the execution command once, then turn it ON.

\section*{DSORT}
(1) Sorts 32-bit BIN data \(n\) points from (s1) in ascending or descending order. Sort order is specified by the ON or OFF status of SM703.
- When SM703 is OFF : Sorts in ascending order
- When SM703 is ON : Sorts in descending order

(2) Several scans are required for sorts performed by the DSORT instruction. The number of scans executed until completion is the value obtained by dividing the maximum number of times executed until the completion of the sort by the number of data blocks compared at one execution specified for ©22. (Fractional part is rounded up.)

When the value of \({ }_{(22}\) is increased, the number of scans until completion of the sort is reduced, but the amount of time per scan is lengthened.
(3) The maximum number of executions until completion of the sort should be calculated according to the following equation.

Maximum number of executions until completion \(=(n) \times(n-1) / 2\) [times executed]

\section*{Example}

When \(n=10\), the number of executions is obtained as \(10 \times(10-1) / 2=45\) [times executed]. If \(S 2=2\), then the number of scans until the completion of sort is calculated as \(45 / 2=22.5 \rightarrow 23\) [scans].
(4) The device specified for © (the completion device) is turned OFF by the execution of the DSORT instruction, and turned ON when the sort is completed. Because the device specified for (d1) is retained in the ON state after the completion of the sort, the user must turn it OFF if required.
(5) The 2 points from the device specified for (d2) are used by the system during the execution of a DSORT instruction. These 2 points from the device specified for (d2) should therefore not be used by the user.
Changing these points may cause an error code to be returned.
(Error code: 4100)
(6) If the value of \(n\) is changed during the execution of the DSORT instruction, the sort will be performed in accordance with the number of sort data blocks after the change.
(7) If the execution command is turned OFF during the execution of the DSORT instruction, the sort is suspended. The sort resumes from the beginning when the execution command is turned ON again.
(8) To execute another sort operation immediately after the completion of the previous sort, turn OFF the execution command once, then turn it ON.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- For the SORT instruction, the range of \(n\) points from \(\Im_{11}\) exceeds the corresponding device range.
(Error code: 4101)
- For the DSORT instruction, the range of \(2 \times \mathrm{n}\) points from \(\leqq 1\) exceeds the corresponding device range.
(Error code: 4101)
- The device range of \((\mathrm{n} / 2 \times \mathrm{n})\) points from the device specified for (s1) overlaps with the device range of 2 points from the device specified for (d2).
(Error code: 4101)
- \(\S_{2}\) is 0 or a negative value.
(Error code: 4100)

\section*{Program Example}
(1) In the following program, the 16-bit BIN data from D0 to D3 are sorted in the ascending or descending order when g_bool1 turns ON.
[Structured ladder/FBD]


\section*{[ST]}

OUT(g_bool1,g_bool2);
SORT(g_boo3,D0.4,1,g_bool4,g_int1);
[Operation]

(2) In the following program, the 32-bit BIN data from D0 to D9 are sorted in the ascending or descending order when g_bool1 turns ON.
[Structured ladder/FBD]


\section*{[ST]}

OUT(g_bool1,g_bool2);
DSORT(g_boo3,D0.5,1,g_bool4,g_int1);
[Operation]


\subsection*{7.5.13 Total calculation of 16 -bit data}

P: Executing condition
indicates any of the following

instructions.
WSUM WSUMP


\section*{Function}
(1) Adds all16-bit BIN data of \(n\) points from the device specified for (s), and stores the result to the device specified for (d).


\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The range of \(n\) points from the device specified for (s) exceeds the corresponding device.
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the 16-bit BIN data from D10 to D14 are added when X1C turns ON, and the result is stored to Var_D100.
[Structured ladder/FBD]

[ST]
WSUMP(X1C,D10,5,Var_D100);

\section*{[Operation]}
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|r|}{b15........b} \\
\hline D10 & 4500 (BIN) \\
\hline D11 & 2500 (BIN) \\
\hline D12 & -3276 (BIN) \\
\hline D13 & 6780 (BIN) \\
\hline D14 & 4444 (BIN) \\
\hline
\end{tabular}


D14 4444 (BIN)

\subsection*{7.5.14 Total calculation of 32-bit data}

DWSUM(P)
P: Executing condition
indicates any of the following

instructions.
DWSUM DWSUMP
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{3}{*}{Input argument,} & EN: & \multicolumn{5}{|l|}{Executing condition} & \multicolumn{4}{|l|}{:Bit} \\
\hline & s : & \multicolumn{9}{|l|}{Start number of the device that stores data whose total value is :ANY32 evaluated} \\
\hline & n : & \multicolumn{5}{|l|}{Number of data} & \multicolumn{4}{|l|}{:ANY16} \\
\hline \multirow[t]{6}{*}{Output argument,} & \[
\begin{aligned}
& \text { ENO: } \\
& \text { d: }
\end{aligned}
\] & \multicolumn{5}{|l|}{\begin{tabular}{l}
Execution result \\
Start number or the array of the device that stores the total value
\end{tabular}} & \multicolumn{4}{|l|}{\begin{tabular}{l}
:Bit \\
:Array of ANY16 (0..3)
\end{tabular}} \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{Jinl} & \multirow[t]{2}{*}{U...icina} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{\[
\begin{gathered}
\text { Constant } \\
\text { K, H }
\end{gathered}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (s) & - & & & & & - & & - & - \\
\hline & n & \(\bigcirc\) & & & & & \(\bigcirc\) & & \(\bigcirc\) & - \\
\hline & (d) & \(\bigcirc\) & & & & & - & & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Adds all 32-bit BIN data \(n\) points from the device specified for © (s), and stores the result to the 4 points of devices (4 words) from the one specified for (d).


\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The range of \(n\) points from the device specified for (s) exceeds the corresponding device. (Error code: 4101)
- The device specified for (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)

\section*{\(\square\) Program Example}

In the following program, the 32-bit BIN data in Var_D100 are added when X20 turns ON, and stores the result to Var_D10.
[Structured ladder/FBD]

[ST]
DWSUMP(X20,Var_D100,4,Var_D10);
[Operation]
Var_D100 \begin{tabular}{|r|}
\hline \(11245600(\) BIN \()\) \\
\hline \(27543200(\mathrm{BIN})\) \\
\hline \(558800(\mathrm{BIN})\) \\
\hline\(-15675000(\mathrm{BIN})\) \\
\cline { 2 - 3 } \\
\hline
\end{tabular}

Var_D10 23672600 (BIN)

\subsection*{7.5.15 Average calculation of 16 -/32-bit data}

MEAN, DMEAN

- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported

\begin{tabular}{|c|c|}
\hline & \multirow[t]{2}{*}{indicates any of the following} \\
\hline instructions. & \\
\hline MEAN & MEANP \\
\hline DMEAN & DMEANP \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{3}{*}{Input argument,} & EN: & \multicolumn{5}{|l|}{Executing condition} & \multicolumn{4}{|l|}{} \\
\hline & s : & \multicolumn{5}{|l|}{Start number of the device that stores data whose average value is evaluated} & \multicolumn{4}{|l|}{:ANY16/32} \\
\hline & n : & \multicolumn{5}{|l|}{Number of data, or device number that stores the number of data. (Setting range: 1 to 32767)} & \multicolumn{2}{|l|}{:ANY16} & & \\
\hline \multirow[t]{6}{*}{Output argument,} & \[
\begin{aligned}
& \text { ENO: } \\
& \text { d: }
\end{aligned}
\] & \multicolumn{5}{|l|}{Execution result} & \multicolumn{2}{|l|}{:Bit} & & \\
\hline & Setting & Inte & vice & R, ZR & & & u & Zn & Constant & Others \\
\hline & data & Bit & Word & & Bit & Word & & & K, H & \\
\hline & (5) & - & \(\bigcirc\) & \(\bigcirc\) & & & - & & - & - \\
\hline & n & - & \(\bigcirc\) & \(\bigcirc\) & & & O & & \(\bigcirc\) & - \\
\hline & (d) & - & \(\bigcirc\) & \(\bigcirc\) & & & - & & - & - \\
\hline
\end{tabular}

\section*{Function}

\section*{MEAN(P)}
(1) Evaluates the average value of \(n\) points (16-bit BIN data) from the device specified for (5), and stores the result to the device specified for © .

(2) If the result is not an integer value, the fractional part is rounded up.
(3) No processing is performed if 0 is specified for \(n\).

\section*{DMEAN(P)}
(1) Evaluates the average value of \(n\) points (32-bit BIN data) from the device specified for (s), and stores the result to the device specified for (d).
(s) +1, (s)
(s) +3 , 2 (s) +2
(s) \(+2 n-1\), (s) \(+2 n-2 ~\)
(2) If the result is not an integer value, the fractional part is rounded up.
(3) No processing is performed if 0 is specified for \(n\).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value specified for n is not within 0 to 32767.
(Error code: 4100)
- The n points of device specified for (s) exceed the specified device range.
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the average value of 16-bit BIN data in D0 to D2 is evaluated when M0 turns ON, and the result is stored to D10.
[Structured ladder/FBD]


\section*{[ST]}

MEAN(M0,D0,K3,D10);
[Operation]

(2) In the following program, the average value of 32-bit BIN data in D0 to D5 is evaluated when M0 turns ON, and the result is stored to D10, D11.
[Structured ladder/FBD]


\section*{[ST] \\ DMEAN(M0,D0,K3,D10);}
[Operation]
\begin{tabular}{l|r|} 
& \\
\begin{tabular}{l|l|} 
D1,D0 & \(623541(\mathrm{BIN})\) \\
D3,D2 & \(4753647(\mathrm{BIN})\) \\
D5,D4 & \(926342(\mathrm{BIN})\) \\
\cline { 2 - 3 } &
\end{tabular}\(\quad \square \mathrm{D} 11, \mathrm{D} 102101176(\mathrm{BIN})\) \\
\hline
\end{tabular}

\subsection*{7.6 Structured Instructions}

\subsection*{7.6.1 FOR to NEXT instruction loop}

FOR
NEXT



Input argument,
EN: Executing condition
:Bit
n : \(\quad\) Number of repeats of the FOR to NEXT instruction loop (1 to :ANY16
32767)

Output argument, ENO: Execution result
:Bit
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & & vice & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..alin} & \multirow{2}{*}{U...|cala} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant
\[
\mathrm{K}, \mathrm{H}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline n & \multicolumn{8}{|c|}{\(\bigcirc\)} & - \\
\hline
\end{tabular}

\section*{Function}
(1) Executes the processing of the next step of the NEXT instruction, when the process of the FOR to NEXT instruction loop is executed \(n\) times without conditions.
(2) The value of \(n\) can be specified with the value between 1 and 32767 . If it is specified in the range from -32768 to 0 , executes the same processing as \(n=1\).
(3) If you do not desire to execute the process of the FOR to NEXT instruction loop, use the CJ or SCJ instruction to jump.
(4) Up to 16 nesting levels are applicable for the FOR instruction.


\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The FEND or GOEND instruction was executed before the execution of the NEXT instruction and after the execution of the FOR instruction.
(Error code: 4200)
- The NEXT instruction is executed prior to the execution of the FOR instruction.
(Error code: 4201)
- The STOP instruction has been inserted between the FOR and NEXT instructions.
(Error code: 4200)
- The 17th FOR instruction is executed when FOR instructions have been nested.
(Error code: 4202)

\section*{\(\square\) Program Example}

In the following program, the FOR to NEXT instruction loop is executed when X8 is OFF, and not executed when X 8 is ON .
[Structured ladder/FBD]


Remark
1. To force an end to the repetitious execution of the FOR to NEXT instruction loop, insert the BREAK instruction. For details on the BREAK instruction, refer to Section 7.6.2.
2. Use the EGP or EGF instruction to perform the pulse operation of an indexed program in the FOR to NEXT instruction loop.
Note, however, that rise and fall instructions are not available on the operation output side.
For details on the EGP and EGF instructions, refer to Section 5.2.5. The program samples are shown below.

3. Branching into the FOR to NEXT instruction loop using the JMP instruction or other branch instructions are not applicable from the outside of the FOR to NEXT instruction loop.

\subsection*{7.6.2 Forced termination of FOR to NEXT instruction loop}

BREAK(P)
P: Executing condition

\begin{tabular}{lll} 
Input argument, & \(\mathrm{EN}:\) & Executing condition \\
\(\mathrm{p}:\) & \begin{tabular}{l} 
Pointer number of branch destination when the repeat process :ANY16 \\
is forcibly terminated
\end{tabular} \\
Output argument, & ENO: & \begin{tabular}{l} 
Execution result \\
\(\mathrm{d}:\)
\end{tabular}
\end{tabular} \begin{tabular}{l} 
Device number that stores the remaining numbers of repeats :ANY16
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & & evice & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...al} & \multirow[b]{2}{*}{U..ic...} & \multirow{2}{*}{Zn} & \multirow{2}{*}{Constant} & \multirow[t]{2}{*}{Others P} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline p & \multicolumn{7}{|c|}{-} & - & \(\bigcirc\) \\
\hline (d) & \multicolumn{7}{|c|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Forces an end to the FOR to NEXT instruction loop and shifts the operation to the pointer specified for Pn. Only a pointer within the same program file can be assigned to Pn. If a pointer of the other program file is used, an operation error will be returned.

(2) The remaining number of the FOR to NEXT instruction loops is stored to (d) at the time of forced termination.
Note that the remaining number includes the operation when the BREAK \((P)\) instruction is executed.
(3) The BREAK (P) instruction can be used only in the FOR to NEXT instruction loop.
(4) The BREAK(P) instruction can be used only when there is only one level of nesting. When termination is forced to the multiple nesting levels, execute the same number of the BREAK \((P)\) instructions for the nesting levels.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The BREAK \((P)\) instruction is used in a case other than with the FOR to NEXT instruction loop.
- The jump destination for the pointer specified for Pn does not exist.
(Error code: 4203)
- The pointer of another program file is specified for Pn.
(Error code: 4210)
(Error code: 4210)

\section*{\(\square\) Program Example}

In the following program, the forced termination is executed on the processes in the FOR to NEXT instruction loop when the value of Var_DO reaches 30 (when the FOR to NEXT instruction loop has been executed 30 times).
[Structured ladder/FBD]


\section*{Remark}

The value 71 is stored to Var_D1 when the BREAK \((P)\) instruction is executed.

\subsection*{7.6.3 Subroutine program call}

CALL


\section*{Function}
(1) When the CALL(P) instruction is executed, executes the subroutine program of the program specified for \(P n\).
[The CALL(P) instruction can execute subroutine programs specified for a pointer within the same program file and subroutine programs specified for a common pointer.

(2) Up to 16 nesting levels are applicable for the \(\operatorname{CALL}(P)\) instruction.

(3) Devices which are turned ON within subroutine programs will be latched even if the subroutine program is not executed.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The FEND, GOEND, or STOP instruction is executed after the execution of the CALL(P) instruction, and prior to the execution of the RET instruction.
(Error code: 4211)
- The RET instruction is executed prior to the execution of the CALL(P) instruction.
(Error code: 4212)
- The 17th nesting level is executed.
(Error code: 4213)
- There is no subroutine program for the pointer specified in the \(\operatorname{CALL}(\mathrm{P})\) instruction.
(Error code: 4210)

\section*{\(\square\) Program Example}

In the following program, the subroutine program is executed when X 20 turns ON .
[Structured ladder/FBD]


\subsection*{7.6.4 Return from subroutine program}

RET

Process
Redundant
Universal
LCPU

\section*{RET}


\section*{F Function}
(1) Indicates the end of the subroutine program.
(2) When the RET instruction is executed, returns to the next step of the CALL(P) instruction which called the subroutine program.


\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The FEND, GOEND or STOP instruction is executed after the execution of the CALL(P) instruction, and prior to the execution of the RET instruction.
(Error code: 4211)
- The RET instruction is executed prior to the execution of the CALL(P) instruction.
(Error code: 4212)

\subsection*{7.6.5 Refresh}

For the COM instruction of the following CPU modules, refer to Section 7.6.6.
- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later
- High Performance model QCPU: Supported if first 5 digits of the serial number are "04012" or later
- Process CPU: Supported if first 5 digits of the serial number are " 07032 " or later
- Redundant CPU
- Universal model QCPU
- LCPU

COM


Input argument, EN:

Executing condition
:Bit
Output argument,
ENO:
Execution result
:Bit

(1) Use the COM instruction in any of the following cases.
(a) For increasing the speed of transmission/reception processing to/from the remote I/O stations.
(b) For ensuring reliable data transmission/reception with other stations that use different scan times during the execution of the data link.
(2) The processing of the COM instruction differs depending on whether the special relay SM775 is ON or OFF.
- When SM775 is OFF: Performs auto refresh and service processing *1 *2
- When SM775 is ON: Performs service processing only *1
*1: The following processes are performed in service processing.
- Monitor processing of other stations
- Read processing by the serial communications module of the buffer memory of another intelligent function module
*2: The auto refresh includes the following processes.
- Refresh of MELSECNET/10 and MELSECNET/H
- CC-Link refresh
- Auto refresh of intelligent function modules.
(3) At the point of the execution of the COM instruction, the CPU module temporarily stops the processing of the sequence program, and performs the same operation as ordinary data processing as well as auto refresh of intelligent function modules (including link refreshes) at the END processing. However, the low speed cyclic refresh of
MELSECNET/10 or MELSECNET/H is not performed.

(4) The COM instruction can be used in a sequence program any number of times. However, note that the scan time of the sequence program will be lengthened by the time taken for service processing and the auto refresh (including the link refresh) of the intelligent function modules.
(5) Data communications using the COM instruction
(a) Example of data communications when the COM instruction is not used

(b) Example of data communications when the COM instruction has been used

1) When the COM instruction is used at the host station, it is possible to increase the number of data communication repetitions with the remote I/O station unconditionally, as shown in (b) above, and thus to speed up data communications.
2) In cases where the remote station scan time is longer than the scan time of the host station, the COM instruction used at the remote station side can avoid the occurrence of timing failure in which the data cannot be fetched, as shown in (a).
3) When the COM instruction has been used at the other station, a link refresh will be performed each time that station receives a command from the host station.
\(\left.\begin{array}{l}\text { Step } 0 \\
\text { COM instruction } \sim \text { COM instruction } \\
\text { COM instruction } \sim \text { END }\end{array}\right)\)\begin{tabular}{l} 
Link refresh can be performed \\
once in each of these intervals.
\end{tabular}
(6) If the scan time from the linked station is longer than the sequence program scan time at the host station, specifying the COM instruction at the host station will not increase the speed of data communications.


XPOINT
The programs in which the COM instruction cannot be used are shown below.
- Low-speed execution type programs
- Interrupt programs
- Fixed scan execution type programs

\section*{O Operation Error}

No operation error occurs in the execution of the COM instruction.

\subsection*{7.6.6 Selection of refresh}

For the COM instruction of the following CPU modules, refer to Section 7.6.5.
- Basic model QCPU: Supported if first 5 digits of the serial number are " 04121 " or earlier
- High Performance model QCPU: Supported if first 5 digits of the serial number are " 04011 " or earlier
- Process CPU : Supported if first 5 digits of the serial number are "07031" or earlier

\author{
Ver. Ver. Ver. \\ Basic \\ - Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later \\ - High Performance model QCPU: Supported if first 5 digits of the serial number are "04012" or later \\ - Process CPU: Supported if first 5 digits of the serial number are "07032" or later
}

\section*{COM}

\(\begin{array}{llll}\text { Input argument, } & \text { EN: } & \text { Executing condition } & : \text { Bit } \\ \text { Output argument, } & \text { ENO: } & \text { Execution result } & : \text { Bit }\end{array}\)

\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J?} & \multirow[b]{2}{*}{U:...ican} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline - & \multicolumn{9}{|c|}{-} \\
\hline
\end{tabular}
(1) Use the COM instruction in any of the following cases.
(a) For increasing the speed of transmission/reception processing to/from the I/O refresh.
(b) For executing data transmission/reception with other stations that use different scan times.
(2) When the COM instruction is executed, the following processing can be performed.
\begin{tabular}{|c|c|c|}
\hline Processing details & QCPU (Q mode) & LCPU \\
\hline 1/O refresh & \(\bigcirc\) & \(\bigcirc\) \\
\hline CC-Link refresh & \(\bigcirc\) & \(\bigcirc\) \\
\hline CC-Link IE Controller Network refresh & \(\bigcirc\) & - \\
\hline CC-Link IE Field Network refresh & *1 & \({ }^{*} 2\) \\
\hline MELSECNET/H refresh & \(\bigcirc\) & - \\
\hline Auto refresh of intelligent function modules & \(\bigcirc\) & \(\bigcirc\) \\
\hline Auto refresh using the QCPU standard area of the multiple CPU system & \(\bigcirc\) & - \\
\hline Importing input/output data from CPUs that are not within the multiple CPU system & \(\bigcirc\) & - \\
\hline Auto refresh using the multiple CPU high speed transmission area of the multiple CPU system & \(\bigcirc\) & - \\
\hline Communication with display module & - & \(\bigcirc\) \\
\hline Service processing (communication with programming tool, GOT or other peripherals) & \(\bigcirc\) & \(\bigcirc\) \\
\hline
\end{tabular}
*1: CPU modules with a serial number whose first 5 digits are '12012' or higher are supported.
*2: CPU modules with a serial number whose first 5 digits are '13012' or higher are supported.

\section*{Remark}

The following processes are performed in service processing.
- Monitoring process of other station
- Read process of another intelligent function module buffer memory by the serial communication module
(3) Turning OFF SM775 executes all processing except for I/O refresh.
(4) When selecting refresh items
(a) Select the processing by SD778, and turn SM775 ON.

The following table shows the refresh items when SM775 is ON or OFF, and the processing that can be specified for SD778.
\begin{tabular}{|c|c|c|c|c|}
\hline \multirow[b]{2}{*}{Processing details} & \multicolumn{2}{|c|}{QCPU (Q mode)} & \multicolumn{2}{|c|}{LCPU} \\
\hline & When SM775 is OFF & When SM775 is ON & When SM775 is OFF & When SM775 is ON \\
\hline I/O refresh & Not executed & \multirow{9}{*}{Whether to be executed or not can be selected} & Not executed & \multirow[t]{3}{*}{Whether to be executed or not can be selected.} \\
\hline CC-Link refresh & \multirow{8}{*}{Executed} & & Executed & \\
\hline CC-Link IE Controller Network refresh & & & - & \\
\hline CC-Link IE Field Network refresh & & & Executed & Whether to be executed or not can be selected \\
\hline MELSECNET/H refresh & & & - & - \\
\hline Auto refresh of intelligent function modules & & & Executed & Whether to be executed or not can be selected \\
\hline Auto refresh using the QCPU standard area of the multiple CPU system & & & - & - \\
\hline Importing input/output data from CPUs that are not within the multiple CPU system & & & - & - \\
\hline Auto refresh using the multiple CPU high speed transmission area of the multiple CPU system & & & - & - \\
\hline Communication with display module & - & - & & \\
\hline Service processing (communication with programming tool, GOT or other peripherals) & Executed & Whether to be executed or not can be selected & Executed & \begin{tabular}{l}
executed or not \\
can be selected
\end{tabular} \\
\hline
\end{tabular}
(b) Select whether to execute the processing or not using the bits of SD778.

Whether to execute each bit of SD778 or not can be specified as shown below.

\section*{1) \(Q C P U\) ( \(Q\) mode)}
\begin{tabular}{l|c|c|}
\hline \multicolumn{1}{|c|}{ Bit of SD778 } & Executed & Not executed \\
\hline b0 to b6 & 1 & 0 \\
\hline b15 & 0 & 1 \\
\hline
\end{tabular}


\section*{Example}

To make only the send/receive processing with the remote I/O station faster,
specify the MELSECNET/H refresh only.
(Set only b2 and b15 of SD778 to 1 (SD778: 8004H).)

Refresh between the multiple CPUs by the COM instruction is performed under the following conditions
- Receive operation from other CPUs : When b4 of SD778 (auto refresh of CPU shared memory) is 1 .
- Send operation from host CPU : When b15 of SD778 (execution/nonexecution of service processing) is 0
2) LCPU
\begin{tabular}{|l|c|c|}
\hline \multicolumn{1}{|c|}{ Bit of SD778 } & Executed & Not executed \\
\hline b0 to b3 & 1 & 0 \\
\hline b14 & 1 & 0 \\
\hline b15 & 0 & 1 \\
\hline
\end{tabular}


\section*{Example}

To make only the display module processing faster, specify the Communication with display module only.
(Set b14 and b15 of SD778 to 1 (SD778: C000H).)
(5) The COM instruction can be used in a sequence program any number of times.

(6) The COM instruction can be used in a sequence program any number of times.

However, note that the sequence program scan time will be lengthened by the time taken for the processing selected for SD778.
(7) For Universal model QCPU and LCPU, an interrupt is permitted during the execution of the COM instruction. However, note that the data may be separated if the refresh data are used in an interrupt program.
(8) For Built-in Ethernet port QCPU, Built-in Ethernet port LCPU, and LCPU, when the service processing is performed using the COM instruction with the Ethernet connected, the processing time may be extended.

\section*{XPOINT}
1. The programs in which the COM instruction cannot be used are shown below:
- Low-speed execution type programs
- Interrupt programs
- Fixed scan execution type programs
2. For the redundant CPU, there are restrictions on use of the COM instruction. For details, refer to the following manual.
- QnPRHCPU User's Manual (Redundant System)

No operation error occurs in the execution of the COM instruction.

\subsection*{7.6.7 Selection of refresh}

\section*{CCOM}

- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported


\section*{Function}

For details of the functions, refer to Section 7.6.6.

\section*{O Operation Error}

The \(\operatorname{CCOM}(P)\) instruction is executed in \(\operatorname{QnU}(\mathrm{D})(\mathrm{H}) \mathrm{CPU}\) with the first 5 digits of the serial number are 10101 or lower.
(Error code: 4100)
\(\square\) Program Example
The program in which the execution of refresh instruction can be switched by turning MOON/ OFF.
[Structured ladder/FBD]


\subsection*{7.7 Data Table Operation Instructions}

\subsection*{7.7.1 Writing data to data table}

\section*{FIFW}

P: Executing condition

instructions.
FIFW FIFWP

Input argument,
\begin{tabular}{lll} 
EN: & Executing condition & \(:\) Bit \\
\(\mathrm{s}:\) & Write data to the table, or start device number that stores data & \(:\) ANY16 \\
ENO: & Execution result & :Bit \\
\(\mathrm{d}:\) & Start number of the table & \(:\) ANY16
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...a.i.a} & \multirow[b]{2}{*}{U..19]:} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & \(\bigcirc\) & & & & & \(\bigcirc\) & & \(\bigcirc\) & - \\
\hline (d) & - & & & & & - & & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Stores the 16-bit data specified for (s) to the data table specified for (d).

The number of data blocks stored in the table is stored to (d), and the data specified for (s) are stored to © +1 and the following devices.

(s) \(\qquad\)
(2) When the FIFW(P) instruction is executed for the first time, any values specified for (d) should be cleared.
(3) The number of data blocks to be written in the data table and the data table range should be managed by a user.
(Refer to Program Example (2))

\section*{Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The data table range exceeds the corresponding device range when the FIFW(P) instruction is executed.

\section*{\(\square\) Program Example}
(1) In the following program, the data in Var_D0 are stored to the data table from R0 when X10 turns ON.
[Structured ladder/FBD]

[ST]
FIFWP(X10,Var_D0,R0);
[Operation]

(2) In the following program, the data from X20 to X2F are stored to the data table from D38 to D44 when X1B turns ON, and if there are more than 6 data blocks to be stored, Y60 is turned ON and the FIFW(P) instruction is disabled.
[Structured ladder/FBD]

[ST]
OUT(D38>=6,Y60);
FIFWP(X1B AND NOT(Y60),K4X20,D38);
[Operation]


\section*{Caution}

When using the FIFW instruction, do not branch a line from (d).
If branched, the operation result is not output to the devices or labels of branch destination correctly.

For details of branch point of line from destination, refer to Section 3.2.

\subsection*{7.7.2 Reading oldest data from data table}


FIFR(P)
P: Executing condition

\begin{tabular}{lll} 
Input argument, & EN: & Executing condition \\
& \(\mathrm{s}:\) & Start number of the device that stores read data from the table :ANY16 \\
Output argument, & ENO: & Execution result \\
& \(\mathrm{d}:\) & Start number of the table
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{\(\mathrm{R}, \mathrm{ZR}\)} & \multicolumn{2}{|c|}{J:} & \multirow[b]{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & \multicolumn{2}{|c|}{-} \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \multicolumn{2}{|c|}{-} \\
\hline
\end{tabular}

\section*{Function}
(1) Stores the oldest data ( \((\) d +1 ) input to the table specified for (d) to the device specified for (s). After the execution of the FIFR(P) instruction, the data in the table are all compressed up by one block.

(2) Set the interlock to avoid executing the \(\operatorname{FIFR}(P)\) instruction if the value stored in © is 0 . (Refer to Program Example (1))

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The \(\operatorname{FIFR}(P)\) instruction is executed when the value of © is 0 .
(Error code: 4100)
- The data table range exceeds the corresponding device range when the FIFR(P) instruction is executed.

\section*{\(\triangle\) Program Example}
(1) In the following program, the data in R1 from the data table R0 to R7 are stored to Var_D0 when X10 turns ON.
[Structured ladder/FBD]


\section*{[ST] \\ FIFRP(X10 AND R0>=1,Var_D0,R0);}
[Operation]
\begin{tabular}{|c|c|c|c|c|}
\hline \multicolumn{2}{|r|}{Data table} & \multicolumn{3}{|c|}{Data table} \\
\hline R0 & 5 & \}Number of stored data blocks R0 & 4 & Number of stored data blocks \\
\hline R1 & -123 & \(\rightarrow \mathrm{R} 1\) & 55 & \\
\hline R2 & 55 & \(\longrightarrow \mathrm{R} 2\) & 4321 & \\
\hline R3 & 4321 & \(\rightarrow \mathrm{R} 3\) & 123 & \\
\hline R4 & 123 & \(\longrightarrow \mathrm{R} 4\) & -234 & Data table \\
\hline R5 & -234 & R5 & 0 & \\
\hline R6 & 0 & R6 & 0 & \\
\hline R7 & 0 & R7 & 0 & \\
\hline & & Var_D0 & -123 & \\
\hline
\end{tabular}
(2) In the following program, the data in Var_D0 are stored to the data table D38 to D43 when X10 turns ON, and when the number of stored data reaches 5, the last-stored data in D39 of the data table are stored to Var_R0.
[Structured ladder/FBD]

[ST]
FIFWP(X10,Var_D0,D38);
FIFRP(D38=5,Var_R0,D38);
[Operation]


\section*{Caution}

When using the FIFR instruction, do not branch a line from (d).
If branched, the operation result is not output to the devices or labels of branch destination correctly.

For details of branch point of line from destination, refer to Section 3.2.

\subsection*{7.7.3 Reading newest data from data table}


\section*{Function}
(1) Stores the newest data input to the table specified for (d) to the device specified for © \((5\). After the execution of the \(\operatorname{FPOP}(P)\) instruction, the device storing the data read by the FPOP(P) instruction becomes 0 .

(2) Set the interlock to avoid executing the \(\operatorname{FPOP}(P)\) instruction when the value stored in (d) is 0 . [Refer to Program Example (1)]

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The \(\operatorname{FPOP}(P)\) instruction is executed when the value of (d) is 0 .
- The data table range exceeds the corresponding device range when the FPOP(P) instruction is executed.

\section*{IProgram Example}
(1) In the following program, the last-stored data in the data table R0 to R7 are stored to g_int1 when g_bool1 turns ON.
[Structured ladder/FBD]

[ST]
FPOPP(g_bool1 AND R0>=1,g_int1,R0);
[Operation]

(2) In the following program, the data in Var_D0 are stored to the data table D38 to D43 when X1C turns ON, and when the number of stored data reaches 5, and X1D turns ON, the laststored data in the data table are stored to Var_R0.
[Structured ladder/FBD]

[ST]
FIFWP(X1C,Var_D0,D38);
FPOPP(X1D AND D38=5,Var_R0,D38);
[Operation]


\section*{Caution}

When using the FPOP instruction, do not branch a line from (d).
If branched, the operation result is not output to the devices or labels of branch destination correctly.

For details of branch point of line from destination, refer to Section 3.2.

\subsection*{7.7.4 Deleting/inserting data from/to data table}

FDEL, FINS

\begin{tabular}{|ll|}
\hline & indicates any of the following \\
instructions. & \\
FDEL & FDELP \\
FINS & \\
\\
& \\
\\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{4}{*}{Input argument,} & EN: & \multicolumn{5}{|l|}{Executing condition} & \multicolumn{4}{|l|}{:Bit} \\
\hline & s: & \multicolumn{5}{|l|}{Start device number which stores insert data} & \multicolumn{4}{|l|}{:ANY16} \\
\hline & & \multicolumn{9}{|l|}{Start device number which stores delete data} \\
\hline & n : & \multicolumn{5}{|l|}{Position of the table where data are inserted or deleted} & \multicolumn{4}{|l|}{:ANY16} \\
\hline \multirow[t]{7}{*}{Output argument,} & ENO: & \multicolumn{5}{|l|}{Execution result} & \multicolumn{4}{|l|}{:Bit} \\
\hline & d: & \multicolumn{5}{|l|}{Start number of the table} & \multicolumn{4}{|l|}{:ANY16} \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...a)} & \multirow[t]{2}{*}{U...1an} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{\[
\begin{gathered}
\text { Constant } \\
\text { K, H }
\end{gathered}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (s) & \(\bigcirc\) & & & & & \(\bigcirc\) & & - & - \\
\hline & n & \(\bigcirc\) & & & & & \(\bigcirc\) & & \(\bigcirc\) & - \\
\hline & (d) & - & & & & & - & & - & - \\
\hline
\end{tabular}

\section*{\(\mathcal{Y}\) Function}

\section*{FDEL(P)}

Deletes the nth block of data from the data table specified for (d), and stores the deleted data to the device specified for (s).
After the execution of the FDEL(P) instruction, \((n+1)\) th data and the following data in the table are compressed forward by one block.


\section*{FINS(P)}
(1) Inserts the 16-bit data specified for ©s) in the nth block of the data table specified for (a). After the execution of the FINS(P) instruction, the data in the table following the inserted block are all dropped one position.


\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The nth position from (d) is larger than the data storage number at the execution of the FDEL(P) instruction.
(Error code: 4101)
- The nth position from (d) is larger than the "data storage number +1 " at the execution of the FINS(P) instruction.
(Error code: 4101)
- The value of \(n\) exceeds the device range of the table © at the execution of the FDEL(P) or FINS(P) instruction.
(Error code: 4101)
- The FDEL(P) or FINS \((\mathrm{P})\) instruction is executed when \(\mathrm{n}=0\).
(Error code: 4100)
- The FDEL(P) instruction is executed when the value of © is 0 .
(Error code: 4100)
- The data table range exceeds the corresponding device range when the FDEL(P) or FINS(P) instruction is executed.
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the second data from the table R0 to R7 are deleted when X10 turns ON, and the deleted data are stored to Var_D0.
[Structured ladder/FBD]

[ST]
FDELP(X10,Var_D0,2,R0);
[Operation]

(2) In the following program, the data in Var_D0 are inserted to the third position of the data table R0 to R7 when X10 turns ON.
[Structured ladder/FBD]

[ST]
FINSP(X10,Var_D0,3,R0);
[Operation]


\section*{Caution}

When using the FDEL/FINS instruction, do not branch a line from © .
If branched, the operation result is not output to the devices or labels of branch destination correctly.

For details of branch point of line from destination, refer to Section 3.2.

\subsection*{7.8 Buffer Memory Access Instructions}

\subsection*{7.8.1 Reading 1-/2-word data from intelligent function module} FROM, DFRO
- Universal model QCPU: Not supported for Q00UJCPU

FROM(P)
DFRO(P)

\begin{tabular}{|lc|}
\hline FROM & FROMP \\
instructions. & indicates any of the following \\
DFRO & \\
\\
& \\
\\
\\
\hline
\end{tabular}


\section*{FROM(P)}

Reads the data in n 3 words from the buffer memory address specified for n 2 of the intelligent function module specified for n 1 , and stores the data to the device specified for (d) and the following devices.


\section*{DFRO(P)}

Reads the data in ( \(\mathrm{n} 3 \times 2\) ) words from the buffer memory address specified for n 2 of the intelligent function module specified for n 1 , and stores the data to the device specified for © and the following devices.


\section*{®POINT}

Data read from intelligent function modules is also possible with the use of intelligent function module devices.
For intelligent function module devices, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.
1. The value of \(n 1\) is specified by the upper 3 digits of hexadecimal 4 -digit representation of the start I/O number of the slot in which the intelligent function module is mounted.
< QCPU (Q mode) >

< LCPU >

2. For QCPU (Q mode) and LCPU, the automatic interlock is set for the FROM or DFRO instruction.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- There has been no exchange of signals with the intelligent function module at the execution of the instruction.
(Error code: 1412)
- An error has been detected in the intelligent function module at the execution of the instruction.
(Error code: 1402)
- The I/O number specified for n 1 is not for the intelligent function module.
(Error code: 2110)
- The range of n 3 points ( \(2 \times \mathrm{n} 3\) words for the \(\mathrm{DFRO}(\mathrm{P})\) instruction) from the device specified for © \((\mathbb{C}\) exceeds the specified device range.
(Error code: 4101)
- The address specified for n 2 is outside the buffer memory range.

\section*{\(\square\) Program Example}
(1) In the following program, the CH 1 digital output value of the Q68ADV mounted with I/O numbers 040 to 05F is read to Var_D0 when X0 turned ON. (Reads 1 word of data from address 11 of the buffer memory.)
[Structured ladder/FBD]


\section*{[ST]}

FROMP(X0,H4,11,1,Var_D0);
(2) In the following program, the 1-axis current feed value of the QD75P4 mounted at the I/O numbers 040 to 05 F is read to Var_D0, when X0 turns ON. (Reads data in 2 words from the address 800 and 801 of the buffer memory.)
[Structured ladder/FBD]


\section*{[ST]}

DFROP(X0,H4,800,1,Var_D0);

\subsection*{7.8.2 Writing 1-/2-word data to intelligent function module}

TO, DTO

\section*{Basic}


\section*{Function}

\section*{TO(P)}

Writes the data stored in n3 words following the device specified for ©s to the buffer memory address specified for n 2 and the following addresses of the intelligent function module specified for n1.


Intelligent function module


When a constant is specified for (s), writes the same data (value specified for (s) to the area of n3 words following the specified buffer memory. (s) can be specified in the following range: -32768 to 32767 or OH to FFFFH.)

Intelligent function module

(When 5 is specified for (S))
buffer memory


\section*{DTO(P)}

Writes the data stored in ( \(\mathrm{n} 3 \times 2\) ) words following the device specified for (s) to the buffer memory address specified by n 2 and the following addresses of the intelligent function module specified for n 1 .


When a constant is specified for (s), writes the same data (value specified for ©s) to the area of ( \(\mathrm{n} 3 \times 2\) ) words following the specified buffer memory. (s can be specified in the following range: -2147483648 to 2147483647 or OH to FFFFFFFFh.)


\section*{XPOINT}

Data write to intelligent function modules is also possible with the use of intelligent function module devices.
For intelligent function module devices, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- There has been no exchange of signals with the intelligent function module at the execution of the instruction.
(Error code: 1412)
- An error has been detected in the intelligent function module at the execution of the instruction.
(Error code: 1402)
- The I/O number specified for n 1 is not for the intelligent function module.
(Error code: 2110)
- The n3 points ( \(2 \times n 3\) words for the DTO \((P)\) instruction) of the device specified for ©s exceed the specified device range.
(Error code: 4101)
- The address specified for n 2 is outside the buffer memory range. (Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the CH 1 and CH 2 of the Q68ADV with the I/O numbers \(\mathrm{X} / \mathrm{Y} 040\) to X/Y04F are set to the "Disable A/D conversion" mode, when X0 turns ON.
(Writes 3 to the buffer memory address 0 .)
[Structured ladder/FBD]


\section*{[ST]}

TOP(X0,3,H4,0,1);
(2) In the following program, the 1-axis positioning address/movement amount of the QD75P4 with the I/O numbers
X/Y040 to X/Y05F is set to 0 when X0 turns ON. (Writes 0 to buffer memory addresses 2006 and 2007.)
[Structured ladder/FBD]


\section*{[ST]}

DTOP(X0,0,U4,2006,1);
1. The value of \(n 1\) is specified by the upper 3 digits of hexadecimal 4 -digit representation of the start I/O number of the slot in which the intelligent function module is mounted.
< QCPU (Q mode) >

< LCPU >
 read target.
2. For QCPU (Q mode) and LCPU, the automatic interlock is set for the TO or DTO instruction.

\subsection*{7.9 Display Instructions}

\subsection*{7.9.1 Printing ASCII code}


PR


Input argument,
EN: Executing condition
:Bit
:ANY16/string
code
Output argument, ENO: Execution result
:Bit
\(\mathrm{d}: \quad\) Start number of the output module that outputs ASCII code :Bit
data
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...n} & \multirow[b]{2}{*}{U...ng} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant \$} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\triangle^{* 1}\)} & & - & & \(\bigcirc\) & \(\bigcirc\) & - \\
\hline (d) & O(Y only) & \multicolumn{2}{|c|}{-} & & - & & \(\bigcirc\) & - & - \\
\hline
\end{tabular}
*1 :Local devices and file registers per program cannot be used as setting data.

\section*{Function}
(1) Outputs ASCII code data stored in the device specified for (s) or ASCII code data stored in the specified device number and the following devices to the output module specified for (d). The number of characters output differs according to the ON or OFF status of SM701 (For switching the number of output characters).
(a) If SM701 is ON, characters 8 points (16 characters) from the device specified for (s) will be the target of the operation.
Device where ASCII code data are stored

(b) If SM701 is OFF, everything from the device specified for ©s to the 00 H code will be the target of the operation.

(2) The number of points used by the output module is 10 points from the Y address specified for (d).
(3) Output signals from the output module are transmitted at the rate of 30 ms per character. For this reason, the time required to complete the transmission of the specified number of characters ( n ) will be \(30 \mathrm{~ms} \times \mathrm{n}(\mathrm{ms})\).
At 10 ms interrupt intervals, the PR instruction executes data output, strobe signal ON, and strobe signal OFF. The other instructions are executed continuously during a period between the above processes.

(4) In addition to the ASCII code data, the output module also outputs a strobe signal ( 10 ms ON, 20ms OFF) from the © +8 device.
(5) Following the execution of the PR instruction, the PR instruction execution flag (© +9 device) remains ON until the completion of the transmission of the number of specified characters.
(6) The PR and PRC instructions can be used multiple times, but it is preferable to set the interlock with the PR instruction execution flag (© +9 device) so that they will not be ON simultaneously.
(7) If the content of the device in which ASCII code data are stored are changed during the ASCII code data output, the modified data after change will be output.

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- There is no 00 H code within the range of the device specified for (s) when SM701 is OFF.
(Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, the string data "ABCDEFGHIJKLMNOP" is converted to ASCII code data when g_bool1 turns ON and the result is stored to g_string1 (D0), and then the stored ASCII code data in g_string1 (D0) are output to Y14 to Y1B when g_bool2 turns ON. (When SM701 is OFF)
[Structured ladder/FBD]


When g_bool1 turns ON, converts "ABCDEFGHIJKLMOP" to the ASCII code and outputs the result to g_string1.

When g_bool2 turns ON, outputs the ASCII code in g_string1 to Y14 to Y1B.

\section*{[ST]}

IF g_bool1 THEN
RST(TRUE,SM701);
g_string1="ABCDEFGHIJKLMNOP";
END_IF;
PR(g_bool2,g_string1,Y14);
[Timing chart]


\subsection*{7.9.2 Printing comments}

PRC


PRC


Input argument,

Output argument,

PRC
:Bit
:ANY_SIMPLE
:Bit
:Bit
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..f)} & \multirow[b]{2}{*}{U} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant} & \multirow[t]{2}{*}{Others
P,I,J,U} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & & \(\bigcirc\) & & - & - & \(\bigcirc\) \\
\hline (d) & \(\bigcirc\) (Y only) & \multicolumn{2}{|c|}{-} & \multicolumn{3}{|c|}{-} & - & - & - \\
\hline
\end{tabular}

\section*{\(\sqrt{3}\) Function}
(1) Outputs comment data (ASCII code data) in the device specified for ©s to output module specified for (d).
The number of characters output differs according to the ON or OFF status of SM701.
- When SM701 is OFF : Comments in 32 characters
- When SM701 is ON: Comments of upper 16 characters

The number of points used by the output module is 10 points from the \(Y\) address specified for (d).

[Timing chart]

(2) Output signals from the output module are transmitted at the rate of 30 ms per character. For this reason, the time required to complete the transmission of the number of specified characters will be \(30 \mathrm{~ms} \times \mathrm{n}(\mathrm{ms})\).
At 10 ms interrupt intervals, the PRC instruction executes data output, strobe signal ON, and strobe signal OFF. The other instructions are executed continuously during a period between the above processes.

(3) In addition to the ASCII code data, the output module also outputs a strobe signal ( 10 ms ON, 20ms OFF) from the (d) +8 device.
(4) Following the execution of the PRC instruction, the PRC instruction execution flag (© +9 device) remains ON until the completion of the transmission of the number of specified characters.
(5) The PRC instruction can be used multiple times, but it is preferable to set the interlock with the PRC instruction execution flag (d) +9 device) so that they will not be ON simultaneously.
(6) If no comments have been registered to the device specified for ©s, processing will not be performed.
(7) When a comment is read, SM720 turns ON for one scan after the instruction is completed. SM721 turns ON during the execution of the instruction.
The PRC instruction cannot be executed while SM721 is ON. If an attempt is made, no processing is performed.

\section*{®POINT}
1. For device comments used in the PRC instruction, use comment files stored in the memory card or standard ROM.
Comment files stored in the internal memory cannot be used.
2. The comment file used in the PRC instruction can be set at the <<PLC File>> tab in the PLC parameter.
If no comment file has been set for use by the PLC file setting, it will not be possible to output device comments with the PRC instruction.
3. Do not execute the PRC instruction during an interrupt program.

Otherwise, malfunction may occur.

\section*{8 Operation Error}

In the following case, an operation error occurs and the error flag (SM0) is turned ON, and the corresponding error code is stored to SD0.
- The PRC instruction is executed while comments are written during online program change.
\(\triangle\) Program Example
In the following program, the comment data in Y 60 are output to Y 30 to Y 39 when X 0 turns ON .
[Structured ladder/FBD]

[ST]
PRC(X0 AND NOT(Y39),Y60,Y30);

\subsection*{7.9.3 Resetting error display or annunciator}

LEDR


\(\begin{array}{llll}\text { Input argument, } & \text { EN: } & \text { Executing condition } & \text { :Bit } \\ \text { Output argument, } & \text { ENO: } & \text { Execution result } & \text { :Bit }\end{array}\)


\section*{3 Function}

Resets the self-diagnostics error display so that annunciator display or operation can be continued.
With one execution of this instruction, either error display or annunciator is reset.
(1) Operation when self-diagnostics errors occur.
(a) When a self-diagnostics error that allows continued operation occurs. When the LEDR instruction is executed, the "ERR." LED is reset. It is necessary to reset SM0, SM1, and SD0 by the user program, because they are not reset automatically.
Since the cause of the self-diagnostics error being displayed has a higher priority over annunciator, no action for resetting the annunciator is taken.
(b) When a battery error occurs.

If the LEDR instruction is executed after the battery is replaced, the "BAT." LED is reset. SM51 is also turned OFF at this time.
(2) Operations when annunciators (F) are ON.

The following operations are performed when the LEDR instruction is executed.
1) "USER" LED flickers, and is turned OFF.
2) The annunciators (F) stored in SD62 and SD64 are reset, and the F numbers of SD65 to SD79 are moved up.
3) The data newly stored in SD64 are transmitted to SD62.
4) The value in SD63 is decremented by -1 . However, if SD63 is 0 , it remains 0 .

1. The defaults for the error item numbers set in special registers SD207 to SD209 and order of priority are given in the table below.
\begin{tabular}{|c|c|c|c|}
\hline Priority & Factor number (Hexadecimal) & Meaning & Remarks \\
\hline & & & Power off \\
\hline 1 & 1 & \begin{tabular}{l}
AC DOWN \\
SINGLE PS. DOWN \\
SINGLE PS. ERROR
\end{tabular} & \begin{tabular}{l}
Redundant base unit power supply voltage drop (QCPU (Q mode) only) \\
Redundant power supply module fault (QCPU (Q mode) only)
\end{tabular} \\
\hline 2 & 2 & UNIT VERIFY ERR. FUSE BREAK OFF SP. UNIT ERROR SP. UNIT DOWN & \begin{tabular}{l}
I/O module verify error (QCPU (Q mode) only) Blown fuse (QCPU (Q mode) only) \\
Special function module verify error Intelligent function module verification error (QCPU (Q mode) only) Intelligent function module error (LCPU only)
\end{tabular} \\
\hline 3 & 3 & OPERATION ERROR LINK PARA. ERROR SFCP OPE. ERROR SFCP EXE. ERROR REMOTE PASS.FAIL SNTP OPE.ERROR & \begin{tabular}{l}
Operation error \\
Link parameter error (QCPU (Q mode) only) \\
SFC instruction operation error \\
(QCPU (Q mode) only) \\
SFC program execution error \\
(QCPU (Q mode) only) \\
Remote password error (QCPU (Q mode) only) \\
SNTP error (LCPU only)
\end{tabular} \\
\hline 4 & 4 & \begin{tabular}{l}
ICM.OPE ERROR \\
FILE OPE. ERROR EXTEND INST. ERROR OPE. MODE DIFF. CAN'T EXE. MODE TRK.TRANS. ERR. TRK.SIZE ERROR TRK.DISCONNECT FLASH ROM ERROR
\end{tabular} & \begin{tabular}{l}
Memory card operation error \\
File access error \\
Extended instruction error (QCPU (Q mode) \\
only) \\
Operation status, switch mismatch \\
(QCPU (Q mode) only) \\
Current mode-time function execution disabled \\
(QCPU (Q mode) only) \\
Tracking data transmission error \\
(QCPU (Q mode) only) \\
Tracking capacity excess error \\
(QCPU (Q mode) only) \\
Tracking cable not connected, failure \\
(QCPU (Q mode) only) \\
Flash ROM access count exceeded error \\
(LCPU only)
\end{tabular} \\
\hline 5 & 5 & PRG.TIME OVER & Constant scan setting time over error Low speed execution monitoring time over error (QCPU (Q mode) only) \\
\hline 6 & 6 & CHK instruction & - \\
\hline 7 & 7 & Annunciators & - \\
\hline 8 & 8 & LED instruction & - \\
\hline 9 & 9 & BATTERY ERR. & - \\
\hline 10 & A & Clock data & - \\
\hline 11 & B & CAN'T SWITCH STANDBY SYS.DOWN MEM.COPY EXE. & \begin{tabular}{l}
System switching error (QCPU (Q mode) only) Standby system not started/stop error (QCPU (Q mode) only) \\
Memory copy function executed (QCPU (Q mode) only)
\end{tabular} \\
\hline 12 & C & DISPLAY ERROR & Display unit error (LCPU only) \\
\hline
\end{tabular}
2. If the highest priority is given to the annunciator, it can be reset with priority by the LEDR instruction. (For Basic model QCPU, High Performance model QCPU, Process CPU, Redundant CPU)

\subsection*{7.10 Debug/Error Diagnostics Instructions}

\subsection*{7.10.1 Special format error check}

CHKST, CHK

CHKST
CHK

\(\begin{array}{llll}\text { Input argument, } & \text { EN: } & \text { Executing condition } & : \text { Bit } \\ \text { Output argument, } & \text { ENO: } & \text { Execution result } & : \text { Bit }\end{array}\)

\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J"...alin} & \multirow{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline - & \multicolumn{9}{|c|}{-} \\
\hline
\end{tabular}

\section*{Function}

\section*{CHKST}
(1) The CHKST instruction is the instruction that starts the CHK instruction. If the command for the CHKST instruction is OFF, execution jumps from the CHK instruction to the next instruction.
If the command for the CHKST instruction is ON , the CHK instruction is executed.


\section*{CHK}
(1) The CHK instruction is the instruction used for the bidirectional operation as shown below to confirm the nature of the system error.
(a) When the CHK instruction is executed, an error diagnostic check is performed with the specified check conditions, and if a failure is detected, SM80 is turned ON, and the failure number is stored to SD80 as a BCD value.
The error code "9010" will be returned if an error is detected.
The contact number where the error was discovered is stored to the upper 3 digits of SD80 (refer to (3)), and the coil number where the error was detected (refer to (2)) is stored to the lower 1 digit of SD80.

At the error detection of
\begin{tabular}{lll} 
& \begin{tabular}{l} 
At the error detection of \\
Contact No.: 62, Coil No.: 3
\end{tabular} & \\
Before the error detection & After the error detection \\
SM80 OFF & & SM80 ON
\end{tabular}
(b) The contact instruction prior to the CHK instruction does not control the execution of the CHK instruction, but rather sets the check conditions.

(c) A ladder such as the one shown below can be created to perform a cycle time-out check for the system shown above.

(d) The following points should be taken into consideration when creating a ladder using the CHK instruction.
1) The contact numbers for the advance edge sensor and the retract edge sensor ( X must always be in serial. Further, the contact number ( \(\mathrm{X} \cdot \mathrm{j}\) ) for the advance edge sensor should be lower than that for the retract edge.
2) Controls for the advance edge sensor contact number ( X ) and output with the identical number (Yi. *1 are as follows.
When advance operation is in progress .... turn ON
When retract operation is in progress........ turn OFF
*1: Output \(\left(\mathrm{Y}^{-\Gamma}{ }^{-j}\right)\) is treated as an internal relay, and cannot be output to an external device.
(2) Depending on the specified contact, the CHK instruction undergoes processing identical to that shown for the ladder below.

(3) Numbers 1 to 150 from the left base line have been assigned as contact numbers during error detection.

(4) Reset SM80 and SD80 prior to forcing the execution of the CHK instruction.

After the execution of the CHK instruction, it cannot be performed once again until SM80 and SD80 have been reset.
(The content of SM80 and SD80 will be preserved until they are reset by user.)
(5) A CHKST instruction must be placed before the CHK instruction.

An error is returned if an instruction other than the LD, LDI, AND or ANI instruction is used between the CHK instruction and the CHKST instruction.
(Error code: 4235)
(6) The CHK instruction can be written at any step of the program.

However, there is a limit in the number of uses of the CHK instruction.
- Can be used up to two places in all program files being executed.
- Can be used only one place in a single program file.

An error is returned if the CHK instruction is used exceeding the number of uses specified above.
(Error code: 4235)
(7) Place LD and AND instructions prior to the CHK instruction to establish a check condition. Check conditions cannot be set using other contact instructions.
If a check condition has been set with the LDI or ANI instruction, the processing for the check condition they specify will not be performed.
However, contact numbers during error detection can also be assigned to the LDI and ANI instructions.

(8) The error detection method differs according to whether SM710 is ON or OFF.
(a) If SM710 is OFF, checks are performed on coil numbers 1 through 6 in order of contact number.
When the CHK instruction is executed, checks are performed on coils of contact No. 1 in order from coil No. 1 through coil No.6, then move on to contact No. 2 and check the coils in order from coil No. 1 through coil No. 6.
The CHK instruction is completed when coil No. 6 of contact No. \(n\) is checked.
(b) If SM710 is ON, checks are performed on contact numbers 1 through n in order of coil number.
When the CHK instruction is executed, checks are performed on the ladder with coil No. 1 in order from contact No. 1 through contact No. n, then move on to the ladder with coil No. 2 and checks the ladder in order from contact No. 1 through contact No. n. The CHK instruction is completed when contact No. n of coil No. 6 is checked.
(9) If more than one error is detected, the number of the first error detected is stored. Error numbers detected after this are ignored.
(10) The CHK instruction cannot be used in a low speed execution type program. If a low speed execution type program is set in a program file containing the CHK instruction, an operation error is returned, and the CPU module operation is suspended.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- There is a parallel ladder.
(Error code: 4235)
- There are more than 150 contact instructions.
(Error code: 4235)
- The CHK instruction is not executed following the CHKST instruction.
(Error code: 4235)
- The CHK instruction is executed when no CHKST instruction has been executed.
(Error code: 4235)
- The CHKST and CHK instructions are used in a low speed execution type program.
(Error code: 4235)
- There is a instruction other than the LD, LDI, AND or ANI instruction between the CHK instruction and the CHKST instruction.
(Error code: 4235)
- The CHK instruction is used at three places or more in all program files being executed.
(Error code: 4235)
- The CHK instruction is used at two places or more in a single program file.
(Error code: 4235)

\subsection*{7.10.2 Changing check format of the CHK instruction}

CHKCIR, CHKEND


CHKCIR
CHKEND


Input argument, EN:
Output argument,
N: Executing condition
ENO: Execution result
\begin{tabular}{|c|c|}
\hline & \multirow[t]{4}{*}{indicates any of the following} \\
\hline instructions. & \\
\hline CHKCIR & \\
\hline CHKEND & \\
\hline
\end{tabular}
:Bit
:Bit
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...1:} & \multirow{2}{*}{U".alain} & \multirow{2}{*}{Zn} & \multirow{2}{*}{Constant} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline
\end{tabular}

\section*{IT Function}

\section*{CHKCIR, CHKEND}
(1) The check ladder pattern that is used in the CHK instruction can be updated to any format desired.
The actual error checks are performed with the CHKST and CHK instructions.
(2) Error checks are performed according to the check conditions specified by the CHK instruction and the ladder pattern described between the CHKCIR and CHKEND instructions.

For details on the CHKST and CHK instructions, refer to Section 7.10.1.

\section*{XPOINT}

To change the check format of the CHK instruction using the CHKCIR to CHKEND instruction loop, create a ladder with index setting (ZO).
(a) The device numbers indicated at check conditions (X2 and X8 in the figure below) are index setting values for each device number (with the exception of annunciators (F)) described in the ladder patterns.
Example \(\times 10\) in the in the figure below would be as follows.
When corresponding to check condition X2 \(\qquad\) Processing is performed by X12
When corresponding to check condition X8 \(\qquad\) .Processing is performed by X18
However, the order in which error detection is executed differs depending on whether SM710 is ON or OFF.
1) If SM710 is OFF, checks are performed on coil numbers 1 through the end in order of contact number.
[Specified ladder between CHKCIR and CHKEND instructions]

[Order of check in CPU module]

2) If SM710 is ON, checks are performed on contact numbers 1 through the end in order of coil number.
[Specified ladder between CHKCIR [Order of check in CPU module] and CHKEND instructions]

(b) Error check checks the ON/OFF status of OUT F:-j by using the ladder pattern in the various check conditions.

In all check conditions, SM80 is turned ON if even one of the OUT F... is ON in a ladder pattern.
Further, the error numbers (contact numbers and coil numbers) corresponding to the
OUT F \({ }^{\circ-j}\) which are turned ON are stored to SD80 in BCD order.
(c) The instructions that can be used in ladder patterns are as follows.

Contacts ... LD, LDI, AND, ANI, OR, ORI, ANB, ORB, MPS, MPP, MRD, and comparison operation instructions

Coil \(\qquad\) OUT F:.j]
(d) The following devices can be used for ladder pattern contacts. Input (X), Output (Y)
(e) Only annunciators ( \(F\) ) can be used in ladder pattern coils.

However, since annunciators \((F)\) are used as a dummy, any value can be set for an annunciator ( F ).
Further, they can overlap with no difficulties.
(f) ON/OFF controls can be performed without error if an annunciator ( \(F\) ) used during the execution of the CHK instruction has the same number as an annunciator ( \(F\) ) used in some other context than the CHK instruction. They are treated differently during the CHK instruction than they are in the different context.
(g) Since the annunciators (F) used in the CHK instruction do not turn ON/OFF actually, they are turned ON/OFF if monitored by a peripheral.
(h) A ladder pattern can be created up to 256 steps.

Further, OUT F!-j can use up to 9 coils.
(3) Coil numbers for ladders specified within the CHKCIR to CHKEND instruction loop are assigned with coil numbers from 1 to 9 , from top to bottom.

(4) The CHKCIR and CHKEND instructions can be written at any step in the program desired. It can be used in up to two locations in all program files being executed.
However, the CHKCIR and CHKEND instructions cannot be used in more than 1 location in a single program file.
(5) The CHKCIR and CHKEND instructions cannot be used in low speed execution type programs.
If a program file in which the CHKCIR or CHKEND instruction is described is set as a low speed execution type program, an operation error occurs, and the High Performance model QCPU operation is suspended.

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The CHKCIR to CHKEND instruction loop is set three or more times in all program files.
(Error code: 4235)
- The CHKCIR or CHKEND instruction loop is set two or more times in a single program file.
(Error code: 4235)
- The CHKEND instruction is not executed following the execution of the CHKCIR instruction.
(Error code: 4230)
- The CHKEND instruction is executed although no CHKCIR instruction has been executed.
(Error code: 4230)
- The CHKST and CHK instructions are used in a low speed execution type program.
(Error code: 4235)
- There are 10 or more \(F\) instances in a ladder pattern.
(Error code: 4235)
- A ladder pattern has 257 or more steps.
(Error code: 4235)
- A device has been encountered which cannot be used in a ladder pattern.
(Error code: 4235)
- The index setting has been performed on a ladder pattern device.
(Error code: 4235)

\subsection*{7.11 String Processing Instructions}

\subsection*{7.11.1 16-/32-bit BIN data to decimal ASCII data conversion}

> BINDA, DBINDA

\begin{tabular}{|ll|}
\hline & indicates any of the following \\
\hline instructions. & \\
BINDA & BINDAP \\
DBINDA & \\
\\
& \\
& \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Input argument,} & EN: & \multicolumn{5}{|l|}{Executing condition} & \multicolumn{4}{|l|}{:Bit} \\
\hline & s : & \multicolumn{5}{|l|}{BIN data to be converted to ASCII code data} & \multicolumn{4}{|l|}{:ANY16/32} \\
\hline \multirow[t]{6}{*}{Output argument,} & ENO: & \multicolumn{5}{|l|}{Execution result} & \multicolumn{4}{|l|}{:Bit} \\
\hline & d: & \multicolumn{5}{|l|}{Start number of the device that stores conversion result} & \multicolumn{4}{|l|}{:String (8)/(12)} \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...1.]!} & \multirow[b]{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{\[
\begin{gathered}
\text { Constant } \\
\text { K, H }
\end{gathered}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (5) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{\(\bigcirc\)} & & - \\
\hline & (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & & - \\
\hline
\end{tabular}

\section*{I Function}

\section*{BINDA(P)}
(1) Converts each digit number of decimal notation of the 16 -bit BIN data specified for (s) to ASCII code data, and stores the results to the device specified for © and the following devices.
(s)


SM701 is OFF

For example, if -12345 is specified for © , the results are stored to the devices from (d) as shown below.
(s)

(2) BIN value between -32768 and 32767 can be specified for ©s.
(3) The operation results stored to (d) are as follows.
(a) The sign 20 H is stored if the BIN value is positive, and the sign 2DH is stored if it is negative.
(b) The sign 20 H is stored for the leading zeros of significant figures. (Zero suppression is performed.)
\[
\underbrace{00}_{2{ }_{2} 0_{\text {is }} \text { is set }} \underbrace{325}_{\text {Number of significant figures }}
\]
(c) The storage of data in devices specified for (d) +3 differs depending on the ON/OFF status of SM701 (signal for switching the number of output characters).

When SM701 is OFF \(\qquad\) Stores 0
When SM701 is ON ......Does not change

\section*{DBINDA(P)}
(1) Converts each digit number of decimal notation of the 32-bit BIN data specified for (s) to ASCII code data, and stores the results to the device specified for (d) and the following devices.


For example, if the value -12345678 has been specified for (s), the results will be stored to the devices from (d) as shown below.
\begin{tabular}{|c|c|c|c|}
\hline \multirow[b]{3}{*}{(s)} & (d) & 20н (space) & 2Dн (-) \\
\hline & (d) +1 & 31н (1) & 20н (space) \\
\hline & (d) +2 & 33н (3) & 32H (2) \\
\hline \multirow[t]{3}{*}{-1234 5678} & (d) +3 & 35 H (5) & 34H (4) \\
\hline & (d) +4 & 37\% (7) & 36\% (6) \\
\hline & (d) +5 & 0 or 20 H & 38\% (8) \\
\hline
\end{tabular}
(2) BIN value between -2147483648 and 2147483647 can be specified for (s).
(3) The operation results stored to (d) are stored in the following way.
(a) The sign 20 H is stored if the BIN value is positive, and the sign 2DH is stored if it is negative.
(b) The sign 20 H is stored for the leading zeros of significant figures. (Zero suppression is performed.)

20H is set Number of
significant figures
(c) The data stored in the upper 8 bits of the device specified for © +5 differs depending on the ON/OFF status of SM701 (signal for switching the number of output characters).

When SM701 is OFF......Stores 0
When SM701 is ON .......Stores 20 H

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device specified for (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the 16-bit BIN data in Var_W0 are converted to the decimal ASCII code data, and the results are output to Y 40 to Y 48 by the PR instruction. (Global label Var_D0 is assigned to device D0.)
[Structured ladder/FBD]


\section*{[ST]}

RST(SM400,SM701);
BINDAP(SM400,Var_W0,Var_D0);
PR(X0,D0,Y40);
[Operation]
The ASCII code data are output to Y 40 to Y 48 by the execution of the PR instruction when X0 turns ON.
Since SM701 is OFF, the PR instruction outputs ASCII code data up to 00 H .
\begin{tabular}{|c|c|c|c|c|}
\hline \multirow[b]{2}{*}{Var_W0} & \multicolumn{3}{|r|}{b15------------ b8 b7------------ b0} & \multirow[b]{3}{*}{PR} \\
\hline & D0 & 20н (space) & 20н (space) & \\
\hline 5126 & D1 & 31н (1) & 35 H (5) & \\
\hline 5126 & D2 & 36н (6) & 32н (2) & \\
\hline BIN value & D3 & & & utput. \\
\hline
\end{tabular}
(2) In the following program, the 32-bit BIN data in Var_W10 are converted to the decimal ASCII code data, and the results are output to Y 40 to Y 48 by the PR instruction. (Global label Var_D0 is assigned to device D0.)
[Structured ladder/FBD]

[ST]
RST(SM400,SM701);
DBINDAP(SM400,Var_W10,Var_D0);
PR(X0,D0,Y40);
[Operation]
The ASCII code data are output to Y 40 to Y 48 by the execution of the PR instruction when X0 turns ON.
Since SM701 is OFF, the PR instruction outputs ASCII code data up to 00 H .


\subsection*{7.11.2 16-/32-bit BIN data to hexadecimal ASCII data conversion BINHA, DBINHA}

BINHA(P)
DBINHA(P)



Input argument,
Output argument,

EN: Executing condition
s: \(\quad\) BIN data to be converted to ASCII code data
ENO: Execution result
\(\mathrm{d}: \quad\) Start number of the device that stores conversion result
 instructions.
BINHA BINHAP

DBINHA DBINHAP
:Bit
:ANY16 / 32
:Bit
:String (6)/(10)
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{Ј成,} & \multirow{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant
\[
\mathrm{K}, \mathrm{H}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{\(\bigcirc\)} & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{-} & - \\
\hline
\end{tabular}

\section*{Function}

\section*{BINHA(P)}
(1) Converts each digit number of hexadecimal notation of the 16-bit BIN data specified for (s) to ASCII code data, and stores the results to the device specified for (d) and the following devices.
(s)

(d) \begin{tabular}{|l|l|l|} 
ASCII code for the 3rd digit & ASCII code for the 4th digit \\
\hline
\end{tabular}
(d) +2
ASCll code for the 1st digit : ASCII code for the 2nd digit

Only when SM701 is OFF

For example, if 02A6H is specified for (s), the results are stored to the devices from (d) as shown below.
(s)

\begin{tabular}{|c|c|c|}
\hline (d) & 32н (2) & 30н (0) \\
\hline (d) +1 & 36н (6) & 41н (A) \\
\hline (d) +2 & & \\
\hline
\end{tabular}
(2) The BIN data specified for (s) can be in the range from OH to FFFFH.
(3) The operation results stored to (d) are processed as 4-digit hexadecimal values. For this reason, zeros which are significant figures on the left side of the value are processed as 0 . (No zero suppression is performed.)
(4) The data to be stored to the device specified for © +2 differs depending on the ON/OFF status of SM701 (signal for switching the number of output characters).

When SM701 is OFF......Stores 0
When SM701 is ON ......Does not change

\section*{DBINHA(P)}
(1) Converts each digit number of hexadecimal notation of the 32-bit BIN data specified for (s) to ASCII code data, and stores the results to the device specified for (d) and the following devices.


For example, if the value 03AC625EH is specified for (s), the results are stored to the devices from (d) as shown below.

(2) The BIN data specified for © (san be in the range from 0 H to FFFFFFFFFH.
(3) The operation results stored to (d) are processed as 8-digit hexadecimal values. For this reason, zeros which are significant figures on the left side of the value are processed as 0 . (No zero suppression is performed.)
(4) The data to be stored to the device specified for © +4 differs depending on the ON/OFF status of SM701 (signal for switching the number of output characters).

When SM701 is OFF......Stores 0
When SM701 is ON ......Does not change

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device specified for (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)

\section*{Program Example}
(1) In the following program, the 16-bit BIN data in Var_W0 are converted to the hexadecimal ASCII code data, and the results are output to Y40 to Y48 by the PR instruction. (Global label Var_D0 is assigned to device D0.)
[Structured ladder/FBD]

[ST]
RST(SM400,SM701);
BINHAP(SM400,Var_W0,Var_D0);
PR(X0,D0,Y40);

\section*{[Operation]}

The ASCII code data are output to Y 40 to Y 48 by the execution of the PR instruction when X0 turns ON.
Since SM701 is OFF, the PR instruction outputs ASCII code data up to 00H.
\begin{tabular}{|c|c|c|c|c|}
\hline \multicolumn{5}{|c|}{b15------------b8b7-------------b0} \\
\hline Var_W0 & D0 & 43н (C) & 39н (9) & PR \\
\hline 9C06H & D1 & 36н (6) & 30н (0) & \\
\hline BIN value & D2 & \multicolumn{2}{|c|}{00H} & "9C06" is output \\
\hline
\end{tabular}
(2) In the following program, the 32-bit BIN data in Var_W10 are converted to the hexadecimal ASCII code data, and the results are output to Y40 to Y48 by the PR instruction. (Global label Var_D0 is assigned to device D0.)
[Structured ladder/FBD]

[ST]
RST(SM400,SM701);
DBINHAP(SM400,Var_W10,Var_D0);
PR(X0,D0,Y40);
[Operation]
The ASCII code data are output to Y 40 to Y 48 by the execution of the PR instruction when X0 turns ON.
Since SM701 is OFF, the PR instruction outputs ASCII code data up to 00 H .
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multicolumn{5}{|r|}{b15-------------b8b7-------------b0} & \multirow[t]{3}{*}{} & \\
\hline & D0 & 42н (B) & 37\% (7) & \multirow[b]{2}{*}{PR} & & \\
\hline Var_W10 & D1 & 43н (C) & 33н (3) & & & \\
\hline 7 B 3 C 581 FH & D2 & 38н (8) & 35 ( 5 ) & \multicolumn{3}{|r|}{Y40 to Y48} \\
\hline & D3 & 46н (F) & 31н (1) & \multicolumn{3}{|c|}{\multirow[t]{2}{*}{"7B3C581F" is output}} \\
\hline & D4 & \multicolumn{2}{|c|}{00H} & & & \\
\hline
\end{tabular}

\subsection*{7.11.3 4-/8-digit BCD data to decimal ASCII data conversion}

BCDDA, DBCDDA

BCDDA(P)
DBCDDA(P)


Input argument,

EN: Executing condition
s: \(\quad\) BCD data to be converted to ASCII code data
ENO: Execution result
\(\mathrm{d}: \quad\) Start number of the device that stores conversion result
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J:..and} & \multirow[b]{2}{*}{U} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{\(\bigcirc\)} & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{-} & - \\
\hline
\end{tabular}

\section*{F Function}

\section*{BCDDA(P)}
(1) Converts each digit number of hexadecimal notation of the 4-digit BCD data specified for (s) to ASCII code data, and stores the results to the device specified for (d) and the following devices.
(s)
b15---b12 b11---b8 b7----b4 b3----b0
Thousands Hundreds Tens Units
place place place place
b15-------------b8 b7 -b0
A) ASCII code for hundreds place: ASCII code for thousands place
\begin{tabular}{c|c:c|} 
(d) +1 & ASCII code for units place & ASCII code for tens place \\
\cline { 2 - 3 } & (d) +2 & 0
\end{tabular}
Only when SM701 is OFF

For example, if the value 9105 is specified for (s), the results are stored to the devices from (d) as shown below.
(s)

\begin{tabular}{|c|c|c|}
\hline (d) & 31н(1) & 39 \({ }_{\text {H }}\) (9) \\
\hline (d) +1 & 35 \({ }_{\text {H }}\) (5) & \(30_{\mathrm{H}}(0)\) \\
\hline (d) +2 & \multicolumn{2}{|c|}{\(00_{H}\)} \\
\hline
\end{tabular}
(2) The BCD data specified for © can be in the range of from 0 to 9999.
(3) For the results of calculation stored to the device (d), all zeros on the left side of the significant figures are zero-suppressed.
\(\underbrace{00}\)
\(\underbrace{}_{20 \text { is set }}\)
(4) The data to be stored to the device specified for (d) +2 differs depending on the ON/OFF status of SM701 (signal for switching the number of output characters).

When SM701 is OFF \(\qquad\) Stores 0
When SM701 is ON
.......Does not change

\section*{DBCDDA(P)}
(1) Converts each digit number of hexadecimal notation of the 8 -digit BCD data specified for (s) to ASCII code data, and stores the results to the device specified for (d) and the following devices.


For example, if the value 01234056 is specified for (s), the result are stored to the devices from (d) as shown below.

(2) The BCD data specified for © can be in the range of 0 to 99999999.
(3) For the results of calculation stored to the device (d), all zeros on the left side of the significant figures are zero-suppressed.
\(\underbrace{000}_{\prod_{2}{ }^{\text {H }} \text { is set }} \underbrace{12098}_{\text {Number of significant figures }}\)
(4) The data to be stored to the device specified for © +4 differs depending on the ON/OFF status of SM701 (signal for switching the number of output characters).

When SM701 is OFF......Stores 0
When SM701 is ON ......Does not change

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The data of ©s during the operation of the \(\operatorname{BCDDA}(P)\) instruction is outside the range of from 0 to 9999.
(Error code: 4100)
- The data of (s) during the operation of the \(\operatorname{DBCDDA}(\mathrm{P})\) instruction is outside the range of 0 to 99999999.
(Error code: 4100)
- The device specified for (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\triangle\) Program Example}
(1) In the following program, the 4-digit BCD data in Var_W0 are converted to the decimal ASCII code data, and the results are output to Y40 to Y48 by the PR instruction. (Global label Var_D0 is assigned to device D0.)
[Structured ladder/FBD]

[ST]
RST(SM400,SM701);
BCDDAP(SM400,Var_W0,Var_D0);
PR(X0,D0,Y40);
[Operation]
The ASCII code data are output to Y40 to Y48 by the execution of the PR instruction when X0 turns ON.
Since SM701 is OFF, The PR instruction outputs ASCII code data up to 00 H .

(2) In the following program, the 8-digit BCD data in Var_W10 are converted to the decimal ASCII code data, and the results are output to Y40 to Y48 by the PR instruction. (Global label Var_D0 is assigned to device D0.)
[Structured ladder/FBD]

[ST]
RST(SM400,SM701);
DBCDDAP(SM400,Var_W10,Var_D0);
PR(X0,D0,Y40);

\section*{[Operation]}

The ASCII code data are output to Y 40 to Y 48 by the execution of the PR instruction when X0 turns ON.
Since SM701 is OFF, The PR instruction outputs ASCII code data up to 00 H .


\subsection*{7.11.4 Decimal ASCII data to 16-/32-bit BIN data conversion}

DABIN, DDABIN



\section*{DABIN(P)}
(1) Converts decimal ASCII data stored in the device specified for (s) and the following devices to 16 -bit BIN data, and stores the results to the device specified for (d).


For example, if the ASCII code " -25108 " is specified for the devices following (s), the results will be stored to (d) as shown below.

(2) The ASCII data specified for (s) to © +2 can be in the range of -32768 to 32767 .
(3) The sign 20 H will be stored if the BIN value is positive, and the sign 2DH will be stored if it is negative.
(If other than 20 H or 2DH is set, it will be processed as positive data.)
(4) ASCII code can be set for each position within the range from 30 H to 39 H .
(5) If the ASCII code set for each position is 20 H or 00 H , it will be processed as 30 H .

\section*{DDABIN(P)}
(1) Converts decimal ASCII data stored in the device specified for (s) and the following devices to 32-bit BIN data, and stores the results to the device specified for (d).


For example, if the ASCII code -1234543210 is specified for the devices following (s), the result will be stored to (d) as shown below.
\begin{tabular}{|c|c|c|c|}
\hline (s) & 31H (1) & 2D \({ }_{\text {H }}(-)\) & \\
\hline (S) +1 & 33H (3) & 32H (2) & \\
\hline (S) +2 & 35 \({ }_{\text {H (5) }}\) & 34 \({ }^{\text {(4) }}\) & \\
\hline (S) +3 & 33H (3) & 34 \({ }_{\text {H (4) }}\) & \\
\hline (5) +4 & 31H (1) & 32H (2) & \\
\hline (S) +5 & & 30 H (0) & \\
\hline
\end{tabular}
(2) The ASCII data specified for (s) to © +5 can be in the range of -2147483648 to 2147483647.

Further, data stored in the upper bytes of (s) +5 will be ignored.
(3) The sign 20 H will be stored if the BIN value is positive, and the sign 2DH will be stored if it is negative.
(If other than 20 H or 2 DH is set, it will be processed as positive data.)
(4) ASCII code can be set for each position within the range from 30 H to 39 H .
(5) If the ASCII code set for each position is 20 H or 00 H , it will be processed as 30 H .

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The ASCII code specified for (s) to ©s +5 for each number is other than 30 H to \(39 \mathrm{H}, 20 \mathrm{H}\), or 00 H .
(Error code: 4100)
- The ASCII data specified for (s) to © +5 is outside the ranges shown below.
(Error code: 4100)
When DABIN(P) instruction is used \(\qquad\) -32768 to 32767 When DDABIN \((P)\) instruction is used -2147483648 to 2147483647
- The device specified for (s) exceeds the corresponding device range. (For Universal model QCPU and LCPU)

\section*{\(\square\) Program Example}
(1) In the following program, the 5-digit decimal ASCII data and the signs set in Var_D20 are converted to BIN value, and the result is stored to Var_D0.
[Structured ladder/FBD]


\section*{[ST] \\ DABINP(SM400,Var_D20,Var_D0);}
[Operation]
\begin{tabular}{|c|c|c|c|c|}
\hline \multirow[b]{3}{*}{\begin{tabular}{l}
Var_D20 \\
Var D20+1
\end{tabular}} & & & \multirow{4}{*}{(Regarded as -00276)} & \multirow[t]{2}{*}{\[
\begin{aligned}
& \text { Var_D0 } \\
& \hline-276
\end{aligned}
\]} \\
\hline & 20н (space) & 2Dн (-) & & \\
\hline & 32н (2) & 20н (space) & & \\
\hline Var_D20+2 & 36н (6) & 37н (7) & & BIN value \\
\hline
\end{tabular}
(2) In the following program, the 10-digit decimal ASCII data and the signs set in Var_D20 are converted to BIN value, and the result is stored to Var_D10.
[Structured ladder/FBD]

[ST]
DDABINP(SM400,Var_D20,Var_D10);
[Operation]


\subsection*{7.11.5 Hexadecimal ASCII data to \(16-/ 32\)-bit BIN data conversion HABIN, DHABIN}

HABIN(P)
DHABIN(P)

\begin{tabular}{|ll|}
\hline instructions. & indicates any of the following \\
HABIN & HABINP \\
DHABIN & \\
\\
& \\
\\
\hline
\end{tabular}

Input argument,
EN: Executing condition :Bi
\(\mathrm{s}: \quad\) ASCII data to be converted to BIN value, or start number of
:String (4)/(8)
the device that stores ASCII data
Output argument,
\(\mathrm{d}: \quad\) Start number of the device that stores conversion result :ANY16/32
Execution result
:Bit
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Intemal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{} & \multirow[t]{2}{*}{U..1gam} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant \$} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{HABIN(P)}
(1) Converts hexadecimal ASCII data stored in the device specified for (s) and the following devices to 16 -bit BIN data, and stores the results to the device specified for (d).


For example, if the ASCII code 5A8DH is specified for the devices following ©s, the result will be stored to (d) as shown below.

(2) The ASCII data specified for (s) to © +1 can be in the range of 0000 H to FFFFH.
(3) The ASCII code can be in the range of 30 H to 39 H and 41 H to 46 H .

\section*{DHABIN(P)}
(1) Converts hexadecimal ASCII data stored in the device specified for ©s and the following devices to 32-bit BIN data, and stores the results to the device specified for (d).


For example, if the ASCII code 5CB807E1H is specified for the devices following ©s), the result will be stored to (d) +1 and (d) as shown below.

(2) The ASCII data specified for © (to © +3 can be in the range of 00000000 H to FFFFFFFFFH.
(3) The ASCII code can be in the range of 30 H to 39 H and 41 H to 46 H .

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The ASCII code for each number specified for ©s to ©s +3 are outside the range of 30 H to 39 H and 41 H to 46 H .
(Error code: 4100)
- The device specified for © exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the 4-digit hexadecimal ASCII data set in Var_D20 are converted to BIN value, and the results are stored to Var_D0.
[Structured ladder/FBD]


\section*{[ST]}

HABINP(SM400,Var_D20,Var_D0);
[Operation]

(2) In the following program, the 8-digit hexadecimal ASCII data set in Var_D20 are converted to BIN value, and the results are stored to Var_D10.
[Structured ladder/FBD]


\section*{[ST]}

DHABINP(SM400,Var_D20,Var_D10);
[Operation]


\subsection*{7.11.6 Decimal ASCII data to 4-/8-digit BCD data conversion}

DABCD, DDABCD

DABCD(P)
DDABCD (P)


\section*{I Function}

\section*{DABCD(P)}
(1) Converts decimal ASCII data stored in the device specified for © and the following devices to 4-digit BCD data, and stores the results to the device specified for © .


For example, if the ASCII code 8765 is specified for the devices following ©s, the results will be stored to (d) as shown below.

(d)
\begin{tabular}{|c|c|c|c|} 
b15-- b12 b11---b8 b7---- b4b3--- b 0 \\
\hline 8 & 7 & 6 & 5 \\
\hline
\end{tabular}
(2) The ASCII data specified for (s) to © +1 can be in the range of 0 to 9999.
(3) The ASCII code set at each digit can be in the range of 30 H to 39 H .
(4) If ASCII code for each digit is 20 H or 00 H , it is processed as 30 H .

\section*{DDABCD(P)}
(1) Converts decimal ASCII data stored in the device specified for © and the following devices to 8 -digit BCD data, and stores the results to the device specified for (d) and the following devices.


For example, if the ASCII code 87654321 is specified for the devices following © , the results will be stored to (d) as shown below.

(2) The ASCII data specified for © \({ }^{\text {( }) ~ t o ~}\) (s) +3 can be in the range of 0 to 99999999 .
(3) The ASCII code set at each digit can be in the range of 30 H to 39 H .
(4) If ASCII code for each digit is from 20 H to 00 H , it is processed as 30 H .

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- There are characters within the data of (s) that are outside the range of 0 to 9 .
(Error code: 4100)
- The device specified for (s) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\boxed{\square}\) Program Example}
(1) In the following program, the decimal ASCII data set in Var_D20 are converted to the 4-digit \(B C D\) data, and the result is output to Y 40 to Y 4 F .
[Structured ladder/FBD]


Output of the converted \(B C D\) value to the display.
[ST]
DABCDP(SM400,Var_D20,K4Y40);
[Operation]

(2) In the following program, the hexadecimal ASCII data set in Var_D20 are converted to 8-digit \(B C D\) data, and the result is stored to Var_D10, and also they are output to Y40 to Y5F.
[Structured ladder/FBD]


Output of the converted \(B C D\) value to the display.
[ST]
DDABCDP(SM400,Var_D20,Var_D10);
DMOV(SM400,Var_D10,K8Y40);
[Operation]


\subsection*{7.11.7 Reading device comment data}

COMRD


COMRD(P)
\(P\) : Executing condition
ง


Input argument,

Output argument,
in...................... indicates any of the following instructions.
COMRD COMRDP
:Bit
:ANY_SIMPLE
:Bit
:String (32)
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{Jatalat} & \multirow{2}{*}{U...iga} & \multirow{2}{*}{Zn} & \multirow{2}{*}{Constant} & \multirow[t]{2}{*}{Others (BLIS, BLITR, BL,P,I,J,U)} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & \(\bigcirc\) & & & & \(\bigcirc\) & & & & \(\bigcirc\) \\
\hline (d) & - & & & & - & & & & - \\
\hline
\end{tabular}
(1) Reads the comment data in the device specified for ©s, and stores it as ASCII code data to the device specified for © and the following devices.


For example, if the comment in the device specified for ©s is "NO. 1 \(\sqcup\) LINE \(\sqcup\) START", its data will be stored to the devices following (d) as shown below.

(2) If the comment range is set for the device number specified for (s), and no comment is registered, all of the characters for the comment would be processed as 20 H (space).
(3) The operation in the area following the device number where the final character of (d) is stored differs depending on the ON/OFF status of SM701 (signal for switching the number of output characters).
\begin{tabular}{ll} 
When SM701 is OFF & : Does not change \\
When SM701 is ON & : Stores 0
\end{tabular}
(4) When a comment is read, SM720 turns ON for one scan after the instruction is completed. SM721 turns ON during the execution of the instruction.
While SM721 is ON, the COMRD \((P)\) instruction cannot be executed. If the attempt is made, no processing is performed.

\section*{XPOINT}
1. For device comments used with the \(\operatorname{COMRD}(P)\) instruction, use comment files stored in the standard RAM, memory card or SD memory card.
Comment files stored in the program memory cannot be used.
2. Set the comment file used for the \(\operatorname{COMRD}(P)\) instruction can be set in \(\ll P L C\) file>> tab in the PLC parameter.
If the used comment file has not been set in the PLC file setting, device comments cannot be output with the \(\operatorname{COMRD}(P)\) instruction.
3. Do not execute the COMRD \((P)\) instruction in an interrupt program. Doing so will cause a malfunction.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The comment is not registered to the device number specified for (s).
(Error code: 4100)
- The device number specified for (d) is not a word device.
(Error code: 4101)
- The device specified for (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)

\section*{\(\triangle\) Program Example}

In the following program, the comment set in D100 is stored as ASCII code data in Var_W0 and the following devices when X1C turns ON.
[Structured ladder/FBD]


\section*{[ST]}

RST(X1C,SM701);
COMRDP(X1C,D100,Var_W0);
[Operation]

Comment at D100
LINE_A_TARGET
\begin{tabular}{|c|c|c|}
\hline Var_W0 & 49н (I) & 4Сн (L) \\
\hline Var_W0+1 & 45 (E) & 4Ен (N) \\
\hline Var_W0+2 & 41н (A) & 20 H (space) \\
\hline Var_W0+3 & 54н (T) & 20н (space) \\
\hline Var_W0+4 & 52н (R) & 41н (A) \\
\hline Var_W0+5 & 45H (E) & 47н (G) \\
\hline Var_W0+6 & 20н (space) & 54н (T) \\
\hline Var_W0+7 & 20н (space) & 20 н (space) \\
\hline & & \\
\hline Var_W0+15 & 20н (space) & 20H (space) \\
\hline Var_W0+16 & \multicolumn{2}{|c|}{00H} \\
\hline
\end{tabular}

\section*{Caution}
(1) The processing completes after several scans.
(2) The \(\operatorname{COMRD}(P)\) or PRC instruction is not executed if the start signal (execution command) of the \(\operatorname{COMRD}(\mathrm{P})\) or PRC instruction is turned ON before completion of the instruction (while SM721 is ON). Execute the COMRD(P) or PRC instruction when SM721 is OFF.
(3) Two or more file comments cannot be accessed simultaneously.
(4) The following instructions cannot be executed simultaneously because they use SM721 in common.
\begin{tabular}{l|l|l|l}
\multicolumn{1}{c|}{ Instruction name } & \begin{tabular}{l} 
ON during \\
execution
\end{tabular} & \multicolumn{1}{c|}{\begin{tabular}{c} 
ON for one scan after \\
completion
\end{tabular}} & ON after abnormal completion \\
\hline \begin{tabular}{l} 
SP_FREAD \\
SP_FWRITE
\end{tabular} & \multirow{3}{*}{\begin{tabular}{l} 
SM721
\end{tabular}} & Specified by instruction & (Device specified by instruction) +1 \\
\hline \begin{tabular}{ll} 
PRC \\
COMRD(P)
\end{tabular} & & SM720 & None \\
\hline
\end{tabular}
(5) For High-speed Universal model QCPU and LCPU, when comment files stored in the SD memory card are used, this instruction will not be executed while SM606 (SD memory card forced disable instruction) is ON. If executed, no processing is performed.

\subsection*{7.11.8 Character string length detection}

LEN

\begin{tabular}{|ll|}
\hline LEN \\
instructions. \\
\\
\\
\\
\\
\\
\hline
\end{tabular}

Input argument,
Output argument,
Executing condition
:Bit
\(\mathrm{s}: \quad\) Character string, or start number of the device that stores
:String character string
\(\mathrm{d}: \quad\) Device number that stores the detected character string length:ANY16
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{小成:} & \multirow[b]{2}{*}{} & \multirow[b]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant \$} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & \(\bigcirc\) & - \\
\hline (d) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{Function}

Detects length of character string specified for (s) and stores to the device specified for (d) and the following devices.

Processes the data from the device specified for (s) to the device storing 00 H as a character string.


For example, when the value "ABCDEFGHI" is stored to the devices following © \({ }^{\text {s }}\), the value 9 is stored to (d).


\section*{Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- There is no 00 H set within the corresponding device range following the device specified for (s).
(Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, the length of the character string from D0 is output to Y 40 to Y 4 F as 4digit BCD values.
[Structured ladder/FBD]

[ST]
LENP(SM400,D0,Var_D10);
BCDP(SM400,Var_D10,K4Y40);
[Operation]


\subsection*{7.11.9 16-/32-bit BIN data to character string data conversion}
- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later.


\section*{STR(P)}
(1) Adds a decimal point at the position specified for © (1) \(^{\text {, to the }} 16\)-bit BIN data specified for (s2), converts the data to character string data, and stores the results to the device specified for (d) and the following devices.

(2) \(\square\) \(-123\)
(2) The total number of digits that can be specified for ©s1 is from 2 to 8 digits.
(3) The number of digits in the fractional part that can be specified for (s1) +1 is from 0 to 5 digits. However, the number of digits in the fractional part must be smaller than or equal to the total number of digits minus 3 .
(4) BIN values between -32768 and 32767 can be specified for (s2).
(5) After conversion, character string data are stored to the devices following (d) as indicated below.
(a) The sign 20H (space) will be stored if the BIN value is positive, and the sign 2DH (minus sign) will be stored if it is negative.
(b) If the setting for the number of digits in the fractional part is anything other than \(0,2 \mathrm{EH}\) (. ) will automatically be stored at the position before the first of the specified number of digits.


If the number of digits in the fractional part is 0 , the ASCII code 2 EH (. ) will not be stored.
(c) If the total number of digits in the fractional part is greater than the number of BIN data digits, a zero will be added automatically and the number converted by shifting to the right, so that it would become 0 .

(d) If the total number of digits excluding the sign and the decimal point is greater than the number of BIN data digits, 20 H (space) will be stored between the sign and the numeric value.


If the number of BIN digits is greater, an error will be returned.
(e) The value 00 H is automatically stored at the end of the converted character string.

\section*{DSTR(P)}
(1) Adds a decimal point at the position specified for (s1) to the 32-bit BIN data specified for (s2), converts the data to character string data, and stores the results to the device specified for (d) and the following devices.

(2) The total number of digits that can be specified for (s1) is from 2 to 13 digits.
(3) The number of digits in the fractional part that can be specified for (s11 +1 is from 0 to 10 digits. However, the number of digits following the decimal point must be smaller than or equal to the total number of digits minus 3 .
(4) BIN values between -2147483648 and 2147483647 can be specified for \(\S_{2}\).
(5) After conversion, character string data are stored to the device numbers following (d) as indicated below.
(a) The sign 20 H (space) will be stored if the BIN value is positive, and the sign 2DH (minus sign) will be stored if it is negative.
(b) If the setting for the number of digits in the fractional part is anything other than \(0,2 \mathrm{EH}\) (. ) will automatically be stored at the position before the first of the specified number of digits.


If the number of digits in the fractional part is 0 , the ASCII code 2EH (. ) will not be stored.
(c) If the total number of digits in the fractional part is greater than the number of BIN data digits, a zero will be added automatically and the number converted by shifting to the right, so that it would become 0 .

(d) If the total number of digits excluding the sign and the decimal point is greater than the number of BIN data digits, 20 H (space) will be stored between the sign and the numeric value.


If the number of BIN digits is greater, an error will be returned.
(e) The value 00 H is automatically stored at the end of the converted character string.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The total number of digits specified for ©s1 is outside of the ranges shown below.
(Error code: 4100)
When the \(\operatorname{STR}(P)\) instruction is in use ............. 2 to 8 digits When the \(\operatorname{DSTR}(P)\) instruction is in use ........... 2 to 13 digits
- The number of digits in the fractional part specified for \((511)+1\) is outside of the range shown below.
(Error code: 4100) When the STR instruction is in use 0 to 5 digits When the DSTR instruction is in use 0 to 10 digits
- The relationship between the total number of digits specified for (51) and the number of digits in the fractional part specified for \((51)+1\) is not as shown below.
(Error code: 4100)
Total number of digits minus 3 is equal to or larger than the number of digits in the fractional part.
- The number of digits specified for \((51)\) is smaller than \(2+\) number of digits of the BIN data specified for \(\Omega_{2}\).
(Number of digits in © \({ }^{1}\) ) < Number of digits of the BIN data in \({ }^{52}\) ) without a sign + Number of digits for a sign (+ or -) + Number of digits for decimal point (. )) (Error code: 4100)
- The device range where the character string specified for (d) will be stored exceeds the corresponding device range.
(Error code: 4101)

\section*{\(\triangle\) Program Example}
(1) In the following program, the 16-bit BIN data stored in Var_D10 are converted to character string data in accordance with the digit specification of D0 and D1 when X0 turns ON, and stores the result to Var_D20. (Var_D0 is declared as a global variable assigned with D0 and D1.)
[Structured ladder/FBD]

[ST]
MOVP(X0,12672,Var_D10);
MOVP(X0,7,D0);
MOVP(X0,1,D1);
STRP(X0,Var_D0,Var_D10,Var_D20);
[Operation]

(2) In the following program, the 32 -bit BIN data stored in Var_D10 are converted to character string data in accordance with the digit specification of D0 and D1 when X0 turns ON, and stores the result to Var_D20. (Var_D0 is declared as a global variable assigned with D0 and D1.)
[Structured ladder/FBD]


Sets the data.

Sets the total number of digits.

Sets the number of digits
in decimal fraction.
[ST]
DMOVP(X0,12345678,Var_D10);
MOVP(X0,12,D0);
MOVP(X0,9,D1);
DSTRP(X0,Var_D0,Var_D10,Var_D20);
[Operation]


\subsection*{7.11.10 Character string data to \(16-/ 32\)-bit BIN data conversion}

VAL, DVAL

- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later

\begin{tabular}{|lc|}
\hline instructions. & VALP \\
VAL & DVALP \\
DVAL & \\
\\
& \\
\\
\\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Input argument,} & EN: & \multicolumn{5}{|l|}{Executing condition} & \multicolumn{4}{|l|}{} \\
\hline & s : & \multicolumn{5}{|l|}{Character string data to be converted to BIN data, or start number of the device that stores character string data} & \multicolumn{2}{|l|}{:String (8)/(13)} & & \\
\hline \multirow[t]{8}{*}{Output argument,} & ENO: & \multicolumn{5}{|l|}{Execution result} & \multicolumn{4}{|l|}{:Bit} \\
\hline & d1: & \multicolumn{5}{|l|}{Start number of the device that stores number of digits of the converted BIN data} & \multicolumn{4}{|l|}{:ANY32} \\
\hline & d2: & \multicolumn{5}{|l|}{Start number of the device that stores converted BIN data} & \multicolumn{4}{|l|}{:ANY16/32} \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...alin} & \multirow[t]{2}{*}{Unilag} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant \$} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (s) & - & & & & & - & & \(\bigcirc\) & - \\
\hline & (d1) & \(\bigcirc\) & & & & & - & & - & - \\
\hline & (d2) & \(\bigcirc\) & & & & & O & & - & - \\
\hline
\end{tabular}

\section*{VAL(P)}
(1) Converts character string data stored in the device specified for ©s and the following devices to 16 -bit BIN data, and stores the number of digits and BIN data to (d1) and (d2).

For converting character string data to BIN data, all data from the device specified for © to the device where 00 H is stored will be processed as character string data.


For example, if the character string -123.45 is specified for the devices following © \({ }^{\text {s }}\), the result would be stored to (d1) and (d2) as shown below.

(2) The total number of characters that can be specified as a character string for ©s is from 2 to 8 characters.
(3) From 0 to 5 characters from the character string specified for ©s can become the fractional part.
However, this number must not exceed the total number of digits minus 3 .
(4) The range of the numeric character string that can be converted to BIN value is from -32768 to 32767 , ignoring a decimal point.
Numeric value character strings, excluding the sign and the decimal point, can be specified only within the range from 30 H to 39 H .
The value ignoring a decimal point means:
Example: \(-12345.6 \rightarrow-123456\)
(5) The sign 20 H will be stored if the numeric value is positive, and the sign 2DH will be stored if it is negative.
(6) 2 E is set for the decimal point.
(7) The total number of digits stored to ©11) is the total number of characters that represent numeric values including a sign and decimal point.

The number of digits in the fractional part stored to © \(₫ 1\) +1 is the number of characters that represent fractional part separated by 2Ен (. ).

The BIN data stored to (d2) is the character string ignoring the decimal point that has been converted to BIN value.
(8) In cases where the character string specified for © contains 20 H (space) or \(30 \mathrm{H}(0)\) between the sign and the first numeric value other than 0 , these 20 H and 30 H are ignored in the conversion to BIN value.


\section*{DVAL(P)}
(1) Converts the character string data stored in the device specified for (s) and the following devices to 32-bit BIN data, and stores the digits numbers and BIN data to ©11) and (d2).

For converting character string data to BIN data, all data from the device specified for (s) to the device where 00 H is stored will be processed as character strings.

(2) The total number of characters in the character string specified for ©s is from 2 to 13 characters.
(3) From 0 to 10 characters in the character string specified for ©s can be the fractional part. However, this number must not exceed the total number of digits minus 3.
(4) The range of the numeric character string that can be converted to BIN value is from -2147483648 to 2147483647 , excluding the decimal point.
Numeric value character strings, excluding the sign and the decimal point, can be specified only within the range from 30 H to 39 H .
(5) The sign 20 H will be stored if the numeric value is positive, and the sign 2DH will be stored if it is negative.
(6) 2 EH is set for the decimal point.
(7) The total number of digits stored to (d1) is the total number of characters that represent numeric values including a sign and decimal point.

The number of digits in the fractional part stored to @1)+1 is the number of characters that represent fractional part separated by 2EH (. ).
The BIN data stored to (d2) is the character string ignoring the decimal point that has been converted to BIN value.
(8) In cases where the character string specified for © contains 20 H (space) or 30 H ( 0 ) between the sign and the first numeric value other than 0 , these 20 H and 30 H are ignored in the conversion to BIN value.


\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The number of characters in the character string specified for ©s is outside of the ranges shown below.
(Error code: 4100)
When \(\operatorname{VAL}(P)\) instruction is in use
2 to 8 characters
When \(\operatorname{DVAL}(P)\) instruction is in use
2 to 13 characters
- The number of characters in the fractional part of the character string specified for (s) falls outside the ranges shown below.
(Error code: 4100)
When \(\operatorname{VAL}(P)\) instruction is in use
0 to 5 characters
When \(\operatorname{DVAL}(P)\) instruction is in use
0 to 10 characters
- The total number of characters of the character string specified for © and the number of characters in the fractional part stand in a relationship that is outside of the range indicated below.
(Error code: 4100)
Total number of characters minus 3 is equal to or greater than the number of characters in the fractional part.
- ASCII code other than 20H or 2DH has been set for the sign.
(Error code: 4100)
- ASCII code other than from 30H to 39 H or 2EH (decimal point) has been set for digits of each number.
(Error code: 4100)
- There has been more than one decimal point set in the value.
(Error code: 4100)
- The value of the BIN value when converted falls outside of the following ranges.
(Error code: 4100)
When \(\operatorname{VAL}(P)\) instruction is in use \(\qquad\) -32768 to 32767
When DVAL(P) instruction is in use \(\qquad\) -2147483648 to 2147483647
- No 00 H is set within the range from the device number specified for ©s to the last device number of the corresponding device.
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the character string data in Var_D20 are read as an integer value and converted to the BIN value when X0 turns ON, and the result is stored to Var_D0.
[Structured ladder/FBD]

[ST]
VALP(X0,Var_D20,Var_D10,Var_D0);

\section*{[Operation]}

(2) In the following program, the character string data in Var_D20 are read as an integer value and converted to the BIN value when X0 turns ON, and the result is stored to Var_D0.
[Structured ladder/FBD]

[ST]
DVALP(X0,Var_D20,Var_D10,Var_D0);
[Operation]


\subsection*{7.11.11 Floating-point data to character string data conversion}

\begin{tabular}{|ll|}
\hline ESTR \\
ESTRP \\
\\
\\
\\
\\
\\
\hline
\end{tabular}
Input argument,
\begin{tabular}{lll} 
EN: & Executing condition & \(:\) Bit \\
\(\mathrm{s} 1:\) & \begin{tabular}{l} 
32-bit floating-point data to be converted, or start number of \\
the device that stores data
\end{tabular} & \(:\) Single-precision real \\
\(\mathrm{s} 2:\) & \begin{tabular}{l} 
Start number or the array of the device that stores display \\
specification of numeric values to be converted
\end{tabular} & \(:\) Array of ANY16 (0..2) \\
ENO: & \begin{tabular}{l} 
Execution result
\end{tabular} & \(:\) Bit \\
\(\mathrm{d}:\) & \begin{tabular}{l} 
Start number of the device that stores converted character \\
string data
\end{tabular} & \(:\) String
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{\begin{tabular}{l}
Setting \\
data
\end{tabular}} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..} & \multirow[b]{2}{*}{U...ic:a} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & - & \(\bigcirc\) & \(\bigcirc\) & - \\
\hline (3) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & - & - & - & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & - & - & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Converts the 32-bit floating-point real number data specified for © 51 to a character string data according to the display specification specified for \(\Omega_{\Omega 2}\), and stores the result to the device specified for (d) and the following devices.
(2) The post-conversion data differs depending on the display specification specified for \(\Im_{2}\).


\section*{When using decimal form}


For example, when the total number of digits is 8 , the number of digits in the fractional part is 3 , and -1.23456 is specified, the operation result would be stored to the devices following
(d) as shown below.

(a) The total number of digits that can be specified for \(\Im_{2}+1\) is as shown below. When the number of digits in the fractional part is 0
\(\qquad\) Number of digits (max.: 24 ) \(\geqq 2\)
When the number of digits in the fractional part is other than 0
\(\qquad\) Number of digits (max.: 24 ) \(\geqq\) (Number of digits in the fractional part +3 )
(b) The number of digits in the fractional part that can be specified for \(\left.\S_{2}\right)+2\) is from 0 to 7 . However, the number of digits in the fractional part must be smaller than or equal to the total number of digits minus 3 .
(c) The converted character string data are stored to the devices following (d) as indicated below.
1) For the sign of integer part, the sign 20 H (space) will be stored if the 32-bit floatingpoint real number is positive, and the sign 2DH (minus sign) will be stored if it is negative.
2) If the fractional part of a 32-bit floating-point real number is out of the range of the digits in the fractional part, the lower decimal values will be rounded off.

4) If the total number of digits, excluding the sign, the decimal point and the fractional part, is greater than the integer part of the 32-bit floating-point real number data, 20 H (space) will be stored between the sign and the integer part.


If the number of digits in the fractional part is 0 , the ASCII code 2EH (. ) will not be stored.
4) If the total number of digits, excluding the sign, the decimal point and the fractional part, is greater than the integer part of the 32-bit floating-point real number data, 20 H (space) will be stored between the sign and the integer part.

5) The value 00 H is automatically stored at the end of the converted character string.

When using exponential form


For example, when the total number of digits is 12 , the number of digits in the fractional part is 4 , and -12.34567 is specified, the operation results would be stored to the devices following (d) as shown below.

\begin{tabular}{|c|c|c|}
\hline (d) & 20н (space) & 2Dн (-) \\
\hline (d) +1 & 2Ен (.) & 31н (1) \\
\hline (d) +2 & 33н (3) & 32н (2) \\
\hline (d) +3 & 36н (6) & 34н (4) \\
\hline (d) +4 & \(2 \mathrm{CH}(+)\) & 45н (E) \\
\hline (d) +5 & 31н (1) & 30н (0) \\
\hline (d) +6 & \multicolumn{2}{|c|}{OOH} \\
\hline
\end{tabular}
(a) The total number of digits that can be specified for \(\S_{2}+1\) is as shown below. When the number of digits in the fractional part is 0
\(\qquad\) Number of digits (max.: 24 ) \(\geqq 2\)
When the number of digits in the fractional part is other than 0
.................... Number of digits (max.: 24) \(\geqq\) (Number of digits in the fractional part +7 )
(b) The number of digits in the fractional part that can be specified for \(\Omega_{2}+2\) is from 0 to 7 . However, the number of digits in the fractional part should be equal to or less than the total number of digits minus 7 .
(c) The converted character string data are stored to the devices following (d) as indicated below.
1) If the 32-bit floating-point real number is positive in value, the sign before the integer part will be stored as ASCII code 20 H (space), and if it is a negative value, the sign will be stored as 2DH ( - ).
2) The integer part is fixed to one digit.

20 H (space) will be stored between the integer part and the sign.

3) If the fractional part of the 32-bit floating-point real number is out of the range of the digits in the fractional part, the lower decimal values will be rounded off.

4) If the number of digits in the fractional part has been set at any value other than 0 , 2EH (. ) will automatically be stored at the position before the first of the specified number of digits.


If the number of digits in the fractional part of the number is 0 , the ASCII code 2EH (.) will not be stored.
5) The ASCII code \(2 \mathrm{CH}(+)\) will be stored as the sign for the exponent part of the value if the exponent is positive in value, and the code 2DH ( - ) will be stored if the exponent is a negative value.
6) The exponent part is fixed at 2 digits.

If the exponent part is only 1 digit, the ASCII code \(30 \mathrm{H}(0)\) will be stored between the sign and the exponent part of the number.

7) The value 00 H is automatically stored at the end of the converted character string.
(3) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The (s1) value is not within the range indicated below.
(Error code: 4100)
\(0, \pm 2^{-126 \leqq(51)}< \pm 2^{128}\)
- The format specified for (s2) was neither 0 nor 1.
(Error code: 4100)
- The total number of digits specified for \(\S_{2}+1\) is outside the ranges shown below.
(Error code: 4100)
When using decimal form
When the number of digits in the fractional part is 0
\(\qquad\) Total number of digits \(\geqq 2\)
When the number of digits in the fractional part is other than 0
\(\ldots \ldots . . . . . . . . . . . . .\). Total number of digits \(\geqq\) (Number of digits in the fractional part +3 )
When using exponential form
When the number of digits in the fractional part is 0
\(\qquad\) Total number of digits \(\geqq 6\)
When the number of digits in the fractional part is other than 0
\(\ldots \ldots . . . . . . . . . . . . .\). Total number of digits \(\geqq\) (Number of digits in the fractional part +7 )
- The number of digits specified in the fractional part of the value for \(\S_{2}+2\) was outside the ranges shown below.
(Error code: 4100)
When using the decimal form
\(\ldots . . . . . . . . . . . . . . . .\). Number of digits in the fractional part \(\leqq\) (Total number of digits -3 )
When using the exponential form
\(\qquad\) . Number of digits in the fractional part \(\leqq\) (Total number of digits -7 )
- The device range to store the character string data specified for (a) exceeds the corresponding device range.
(Error code: 4101)
- The device specified for \(\S_{2}\) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)
- The value of the specified device is - 0 , unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)

\section*{\(\square\) Program Example}
(1) In the following program, the 32-bit floating-point real number data in Var_R0 is converted in accordance with the conversion specification stored in Var_R10 when X0 turns ON, and the result is stored to Var_D0 and the following devices.
[Structured ladder/FBD]


\section*{[ST]}

ESTRP(X0,Var_R0,Var_R10,Var_D0);
[Operation]

(2) In the following program, the 32-bit floating-point real number data in Var_D0 is converted in accordance with the conversion specification stored in Var_R10 when X1C turns ON, and the result is stored to Var_D10 and the following devices.
[Structured ladder/FBD]


\section*{[ST]}

ESTRP(X1C,Var_D0,Var_R10,Var_D10);
[Operation]
\begin{tabular}{|c|c|c|}
\hline Var_R10[0] & 1 (exponential form) & \multirow[t]{4}{*}{\begin{tabular}{l}
Conversion format \\
Total number of digits \\
Number of digits in fractional part
\end{tabular}} \\
\hline Var_R10[1] & 12 & \\
\hline Var_R10[2] & 4 & \\
\hline & & \\
\hline & Var & \\
\hline & 0.0327 & 4578 \\
\hline
\end{tabular}


\subsection*{7.11.12 Character string data to floating-point data conversion}

EVAL
- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later

\section*{EVAL(P)}

P: Executing condition

\begin{tabular}{|l|l|}
\hline EVAL & \\
\\
\\
\\
\\
\\
\\
\\
\hline
\end{tabular}

Input argument,
EN: Executing condition
:Bit
s: \(\quad\) Character string data to be converted to 32-bit floating-point :String (24) real number data, or start number of the device that stores
character string data
Output argument, ENO: Execution result :Bit
d: Start number of the device that stores converted 32-bit :Single-precision real
floating-point real number data
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{小,} & \multirow[t]{2}{*}{U....igan} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant \$} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & & & - & & - & - & \(\bigcirc\) & - \\
\hline (d) & - & & & - & & O & - & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Converts character string data stored in the device specified for © and the following devices to 32-bit floating-point real number data, and stores the result to the device specified for (d).
(2) The specified character string data can be converted to 32-bit floating-point real number data either in the decimal form or the exponential form.

(a) When using decimal form

(b) When using exponential form
\begin{tabular}{|c|c|c|c|}
\hline (s) & 20 H (space) & 2Dн (-) & \multirow[b]{3}{*}{(d)} \\
\hline (s) +1 & 2Ен (.) & 31н (1) & \\
\hline (s) +2 & 32 H (2) & 33н (3) & \\
\hline (s) +3 & 31н (1) & 30н (0) & -1.320 1E+10 \\
\hline (s) +4 & 2Вн (+) & 45 (E) & \\
\hline (S) +5 & 30н (0) & 31н (1) & 32-bit floating-point \\
\hline (s) +6 & & & \\
\hline
\end{tabular}
(3) Excluding the sign, decimal point, and exponent part of the result, 6 digits of the character string data specified for ©s to be converted to a 32-bit floating-point real number data will be effective; the 7th digit and later digit will be cut from the result.
(a) When using decimal form

(b) When using exponential form

(4) In the decimal form, if \(2 \mathrm{BH}(+)\) is specified for the sign or if the specification of sign is omitted, conversion is made assuming a positive value.
If 2DH (-) is specified for the sign, the character string data are converted as a negative value.
(5) In the exponential form, if \(2 \mathrm{BH}(+)\) is specified for the sign in the exponent part or if the specification of sign is omitted, conversion is made assuming a positive value. If \(2 \mathrm{DH}(-)\) is specified for the sign in the exponent part, the character string data are converted as a negative value.
(6) When the ASCII code 20 H (space) or \(30 \mathrm{H}(0)\) exists between numbers not including the initial zero in a character string data specified for (s), it will be ignored when the conversion is performed.

(7) When the ASCII code \(30 \mathrm{H}(0)\) exists between the character " E " and a number in the exponential form character string, the 30 H would be ignored when the conversion is performed.

(8) If the 20 H (space) code is contained in the character string, the code is ignored in the conversion.
(9) Up to 24 characters can be set for a character string.

The codes 20 H (space) and \(30 \mathrm{H}(0)\) contained in the character string are also counted as a character.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The integer part or the fractional part contains a character other than the one in the range of \(30 \mathrm{H}(0)\) to \(39 \mathrm{H}(9)\).
(Error code: 4100)
- There are two or more 2EH (. ) in the character string specified for (d).
(Error code: 4100)
- The exponent part contains the code (character) other than 45H (E), 2BH (+), 45H (E) or 2DH ( - ), or the string contains more than one exponent part.
(Error code: 4100)
- Data after conversion are not within the following range.
(Error code: 4100)
\(0,2^{-126} \leqq \mid\) Data after conversion \(\mid<2^{128}\)
- The code 00 H does not appear within the range from (s) to the corresponding device.
(Error code: 4101)
- The number of characters in the character string following ©s is either 0 or more than 24.
(Error code: 4100)

\section*{\(\square\) Program Example}
(1) In the following program, the character string data in Var_R0 and the following devices are converted to the 32-bit floating-point real number data when X20 turns ON, and the result is stored to Var_D0.
[Structured ladder/FBD]

[ST]
EVALP(X20,Var_R0,Var_D0);
[Operation]

(2) In the following program, the character string data in Var_D10 and the following devices are converted to the 32-bit floating-point real number data when X20 turns ON, and the result is stored to Var_D100.
[Structured ladder/FBD]


\section*{[ST]}

EVALP(X20,Var_D10,Var_D100);

\section*{[Operation]}


\subsection*{7.11.13 Hexadecimal BIN data to ASCII data conversion}

ASC



Input argument,

Output argument,

EN: Executing condition
s: \(\quad\) Start number of the device that stores BIN data to be converted to character string data
n : \(\quad\) Number of characters to be stored
ENO: Execution result
\(\mathrm{d}: \quad\) Start number of the device that stores converted character string data

(1) Converts the 16-bit BIN data stored in the device specified for ©s and the following devices to ASCII data by treating the BIN data in hexadecimal representation. Then, stores the converted data to the device specified for (d) and the following devices, for the number of characters specified for n .


\begin{tabular}{|c|c|c|c|c|}
\hline (s) & 1н & 2 H & 3н & 4 H \\
\hline (s) +1 & 5 H & 6 H & 7 H & 8H \\
\hline (s) +2 & FH & EH & D & CH \\
\hline (S) +3 & Ан & 9 H & Bн & 6 H \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline & & & \multirow[t]{2}{*}{} \\
\hline & & & \\
\hline (d) +1 & 31н(1) & 32н(2) & \multirow[b]{4}{*}{When " 15 " is set for n} \\
\hline \(\rightarrow\) (d) +2 & 37н (7) & 38 H (8) & \\
\hline \(\rightarrow\) (d) +3 & \(35^{+}(5)\) & 36н (6) & \\
\hline \(\rightarrow\{\) (d) +4 & 44H (D) & 43H (C) & \\
\hline (d) +5 & 46н (F) & 45 H (E) & \\
\hline (d) +6 & 42H(B) & 36н (6) & \\
\hline (d) +7 & 0 OH & 39н (9) & \(\checkmark\) \\
\hline
\end{tabular}
(2) The use of \(n\) to set the number of characters causes the BIN data range specified for ©s and the character string storage device range specified for (d) to be set automatically.
(3) Processing will be performed accurately even if the device range where BIN data to be converted is being stored overlaps with the device range where the converted ASCII data will be stored.
\begin{tabular}{|c|c|c|c|c|}
\hline D11 & 4 H & 3 H & 2H & 1H \\
\hline D12 & 8H & 7H & 6 H & 5H \\
\hline D13 & & & Ан & 9 H \\
\hline
\end{tabular}

\begin{tabular}{|c|c|c|}
\hline D10 & 32H & 31H \\
\hline D11 & 34 & 33 H \\
\hline D12 & 36 H & 35 \\
\hline D13 & 38 & 37 \\
\hline D14 & 41H & 39H \\
\hline
\end{tabular}
(4) If an odd number of characters has been specified for \(n\), the ASCII code 00 H will be automatically stored in the upper 8 bits of the final device in the range where the character string is to be stored.
When 5 characters have been specified for \(n\).

(5) If the number of characters specified for n is 0 , conversion processing will not be performed.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The range of the number of characters specified for n following the device number specified for (s) exceeds the corresponding device range.
- The range of the number of characters specified for \(n\) following the device number specified for (d) exceeds the corresponding device range.

\section*{\(\triangle\) Program Example}

In the following program, the BIN data in D0 are read as a hexadecimal value, and converted to the character string data when X0 turns ON, and the results are stored to the devices from D10 to D14.
[Structured ladder/FBD]

[ST]
ASCP(X0,D0,10,D10);
[Operation]


\subsection*{7.11.14 ASCII data to hexadecimal BIN data conversion}

HEX

\begin{tabular}{|l|l|}
\hline HEX & indicates any of the following \\
instructions. & \\
\\
\\
\\
\\
\\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{3}{*}{Input argument,} & EN: & \multicolumn{6}{|l|}{Executing condition} & & & \\
\hline & S: & \multicolumn{9}{|l|}{Start number of the device that stores character string data to :String be converted to BIN data} \\
\hline & n : & \multicolumn{5}{|l|}{Number of characters to be stored} & \multicolumn{4}{|l|}{:ANY16} \\
\hline \multirow[t]{7}{*}{Output argument,} & ENO: & \multicolumn{5}{|l|}{Execution result} & \multicolumn{4}{|l|}{:Bit} \\
\hline & d : & \multicolumn{5}{|l|}{Start number of the device that stores converted BIN data} & \multicolumn{4}{|l|}{:ANY16} \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{\(\mathrm{R}, \mathrm{ZR}\)} & & & \multirow[t]{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant
\[
\mathrm{K}, \mathrm{H}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (S) & - & & & & & - & & & - \\
\hline & (n) & \(\bigcirc\) & & & & & \(\bigcirc\) & & & - \\
\hline & (d) & - & & & & & - & & & - \\
\hline
\end{tabular}

\section*{Function}
(1) Converts the \(n\) characters of hexadecimal ASCII data in the device specified for (s) and the following devices to the BIN value and stores the results to the device specified for (d) and the following devices.


For example, if the number 9 has been specified for n , the operation will be as shown below.


Code " 38 H " remains unchanged since the specified number of characters is " 9 ".
(2) When the number of characters is specified for \(n\), the range of characters specified for (s) as well as the device range specified for (d) in which the BIN data will be stored are automatically decided.
(3) Accurate processing will be performed even in cases where the range of devices where the ASCII data to be converted is being stored overlaps with the range of devices that will store the converted BIN data.

(4) If the number of characters specified for n is not divisible by 4,0 will be automatically stored after the specified number of characters to the final device number of the devices which are storing the converted BIN value.

(5) If the number of characters specified for n is 0 , conversion processing will not be performed.
(6) ASCII code that can be specified for ©s is within the range of 30 H to 39 H and 41 H to 46 H .

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- Characters outside the hexadecimal character string (characters that are not within the range of 30 H to 39 H , or 41 H to 46 H ) have been set for \((\mathrm{s}\).
(Error code: 4100)
- The range of the number of characters specified for n following the device number specified for (s) exceeds the corresponding device range.
(Error code: 4101)
- The range of the number of characters specified for n following the device number specified for (d) exceeds the corresponding device range.
(Error code: 4101)
- The number of characters specified for n is a negative value.
(Error code: 4101)

\section*{/Program Example}

In the following program, the character string data in D0 to D4 are converted to the BIN data when X0 turns ON, and the results are stored to the devices from D10 to D14.
[Structured ladder/FBD]

[ST]
HEXP(X0,D0,10,D10);
[Operation]


\subsection*{7.11.15 Extraction of character string data from right/left}


\section*{Function}

\section*{RIGHT(P)}
(1) Stores n characters from the right side of the character string data (the end of the character string data) which is stored in the device specified for (s) and the following devices, to the device specified for (d) and the following devices.




When \(\mathrm{n}=5\)
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multirow[b]{3}{*}{(s)
\[
\text { (s) }+1
\]} & & & \multirow[b]{3}{*}{(d)} & \multicolumn{2}{|l|}{\multirow[t]{2}{*}{b15------------b8b7-------------b0}} \\
\hline & 42н (B) & 41H (A) & & & \\
\hline & 44н (D) & 43н (C) & & 32н (2) & 31н(1) \\
\hline (S) +2 & 46н (F) & 45н (E) & d) +1 & 34 \({ }^{\text {(4) }}\) & 33н (3) \\
\hline (s) +3 & 32н (2) & 31н (1) & (d) +2 & 00H & 35H (5) \\
\hline (S) +4 & 34н (4) & 33н (3) & & & \\
\hline (S) +5 & 00 H & 35 \({ }^{\text {(5) }}\) & ASCII code for & charact & \\
\hline
\end{tabular}
(2) The NULL code \((00 \mathrm{H})\) indicating the end of the character string is automatically appended at the end of the character string data. For the format of the character string data, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
(3) If the number of characters specified for n is 0 , the NULL code \((00 \mathrm{H})\) will be stored to © .

\section*{LEFT(P)}
(1) Stores n characters from the left side of the character string data (the beginning of the character string data) which is stored in the device specified for (s) and the following devices, to the device specified for (d) and the following devices.


When \(\mathrm{n}=7\)
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{(s)
\[
\text { (s) }+1
\]} & \multicolumn{2}{|l|}{b15------------b8b7-------------b0} & \multirow[b]{2}{*}{(d)} & \multicolumn{2}{|l|}{b15------------b8b7-------------b0} \\
\hline & 42H (B) & 41н (A) & & 42н (B) & 41H (A) \\
\hline (s) +2 & 44 H (D) & 43H (C) & \(\Rightarrow\) (d) +1 & 44н (D) & 43н (C) \\
\hline (5) +2 & 46н (F) & 45H (E) & (d) +2 & 46н (F) & 45 (E) \\
\hline (S) +3 & 32H(2) & 31н(1) & (d) +3 & 00H & 31н (1) \\
\hline (s) +4 & 34 H (4) & 33 (3) & ASCII code for & & \\
\hline (s) +5 & 00H & 35 H (5) & 7th character & & \\
\hline
\end{tabular}
(2) The NULL code \((00 \mathrm{H})\) indicating the end of the character string is automatically appended to the end of the character string data.
For the format of the character string data, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
(3) If the number of characters specified for n is 0 , the NULL code \((00 \mathrm{H})\) will be stored to (d).

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of \(n\) exceeds the number of characters specified for (s).
(Error code: 4101)
- The range of n characters from (d) exceeds the corresponding device.
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the 4 characters of data from the rightmost of the character string data stored in Var_R0 and the following devices are extracted and stored to Var_D0 and the following devices when X0 turns ON.
[Structured ladder/FBD]

[ST]
RIGHTP(X0,Var_R0,4,Var_D0);
[Operation]

(2) In the following program, the number of characters corresponding to the value being stored in Var_D0 from the leftmost of the character string data in Var_D100 are extracted and stored to Var_R10 and the following devices when X1C turns ON.
[Structured ladder/FBD]


\section*{[ST]}

Var_D0:=6;
LEFTP(X1C,Var_D100,Var_D0,Var_R10);
[Operation]

"SQONHD"

\subsection*{7.11.16 Random selection and replacement in character string data MIDR, MIDW}

\begin{tabular}{|l|l|}
\hline & indicates any of the following \\
\hline instructions. & \\
MIDR & MIDRP \\
MIDW & \\
\\
& \\
\\
& \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline \multirow[t]{5}{*}{Input argument,} & EN: & Executing condition & :Bit \\
\hline & s1: & Character string data, or start number of the device that stores character string data & :String \\
\hline & \multirow[t]{3}{*}{s2:} & \multicolumn{2}{|l|}{\multirow[t]{3}{*}{\begin{tabular}{l}
Start number or the array of the device that stores positions of :Array of ANY16 (1..2) first character and number of characters \\
s2[0]: Position of first character \\
s2[1]: Number of characters
\end{tabular}}} \\
\hline & & & \\
\hline & & & \\
\hline Output argument, & ENO: & Execution result & :Bit \\
\hline & d: & Character string data from operation result & :String \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..)} & \multirow[b]{2}{*}{UWig\%} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant \$} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (2) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & - & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{\(\sqrt{3}\) Function}

\section*{MIDR(P)}
(1) Extracts the character string data of \(\Omega_{2}\) [1] characters, from the position specified for \({ }_{(22}\) [0], counted from the left end of the character string data specified for (51), and stores the extracted data to the device specified for (d) and the following devices.

(2) The NULL code \(\left(00_{\mathrm{H}}\right)\) indicating the end of the character string is automatically appended to the end of the character string data.
For the format of the character string data, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
(3) If the number of characters specified for \(\otimes_{2}\) [1] is 0 , the NULL code \(\left(00_{\mathrm{H}}\right)\) is stored in the top of (d).
(4) If the number of characters specified for \({ }_{(22}\) [1] is -1 , stores the data up to the final character specified for (s1) to the device specified for (d) and the following devices.


\section*{MIDW(P)}
(1) Extracts the character string data of \({ }_{(22}\) [1] characters, from the left end of the character string data specified for (51), and stores the extracted data to the character string data specified for (d) to the devices starting from the position specified for \(\mathrm{s}_{2}\) [ 0 ] from the left end.

Before execution
\begin{tabular}{|c|c|c|}
\hline \multirow[b]{3}{*}{\begin{tabular}{l}
(d) \\
(d) +1
\end{tabular}} & \multicolumn{2}{|l|}{b15------------b8b7------------ b0} \\
\hline & 42H (B) & 41н (A) \\
\hline & 44н (D) & 43н (C) \\
\hline (d) +2 & 46H (F) & 45H (E) \\
\hline (d) +3 & 48H (H) & 47H (G) \\
\hline (d) +4 & 00 H & 49H (I) \\
\hline \multicolumn{3}{|c|}{"ABCDEFGHI"} \\
\hline \multicolumn{3}{|c|}{After execution} \\
\hline
\end{tabular}


12345678"
Position counted from the left end of character string data specified for (d)
Number of characters counted from the left end of the character string data specified for s1)
\begin{tabular}{|c|c|c|}
\hline (d) & 42н (B) & 41н (A) \\
\hline (d) +1 & 31H (1) & \(30_{\text {H }}(0)\) \\
\hline (d) +2 & 33H (3) & 32н (2) \\
\hline (d) +3 & 35H (5) & 34 (4) \\
\hline (d) +4 & 00H & 49н (1) \\
\hline
\end{tabular}
(2) The NULL code \(\left(00_{H}\right)\) indicating the end of the character string is automatically appended to the end of the character string data.
For the format of the character string data, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
(3) If the number of characters specified for \(\otimes_{2}\) [1] is 0 , the NULL code \(\left(00_{\mathrm{H}}\right)\) is stored in the top of (d).
(4) If the number of characters specified for ©(22 [1] exceeds the final character from the character string data specified for (d), data will be stored up to the final character.

(5) If the number of characters specified for \(\circledR_{\Omega 2}\) [1] is -1 , stores the data up to the final character specified for (s1) to the device specified for (d) and the following devices.


Position counted from the left end of character string data specified for (d)
Number of characters counted from the left end of character string data specified for (s1)

\begin{tabular}{|c|c|c|}
\hline (d) & 30н (0) & 41н (A) \\
\hline (d) +1 & 32н (2) & 31н (1) \\
\hline (d) +2 & 34н (4) & 33H (3) \\
\hline (d) +3 & 48н (H) & 35н (5) \\
\hline (d) +4 & 4Ан (J) & 49н (I) \\
\hline (d) +5 & OOH & 4Bн (K) \\
\hline
\end{tabular}

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.

For MIDR(P) instruction
- The value of \(\Omega_{2}[0]\) exceeds the number of characters specified for \((511\). (Error code: 4101)
- The \(\S_{2}\) [1] number of characters from position (d) exceeds the device range of (d).
(Error code: 4101)
- When the value of \(\S_{2}\) is 0 .
(Error code: 4101)
- 00 H does not exist within the specified device range following the devices specified for \(\mathrm{s}_{2}\).
(Error code: 4101)
For MIDW(P) instruction
- The value of \(\S_{2}[0]\) exceeds the number of characters specified for © (Error code: 4101)
- The ©2 [1] value exceeds the number of characters in (51).
(Error code: 4101)
- When the value of \(\S_{2}\) is 0 .
(Error code: 4101)
- 00 H does not exist within the specified device range following the devices specified for \(\Omega_{2}\).
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the 3rd character through the 6th character from the left of the character string data stored in Var_D10 and the following devices are extracted and stored to Var_D0 and the following devices when X0 turns ON.
[Structured ladder/FBD]


\section*{[ST]}

MIDRP(X0,Var_D10,Var_R0,Var_D0);
[Operation]

(2) In the following program, the 4 characters of the character string data stored in Var_D0 and the following devices are extracted and stored to the devices from the 3rd character from the left of the character string data in Var_D100 and the following devices when X0 turns ON.
[Structured ladder/FBD]

[ST]
MIDWP(X0,Var_D0,Var_R0,Var_D100);
[Operation]
\begin{tabular}{|c|c|c|}
\hline Var_D0 & 31н (1) & 32н (2) \\
\hline & 45 (E) & 46н (F) \\
\hline & 33 (3) & 30н (0) \\
\hline & \multicolumn{2}{|c|}{00H} \\
\hline & \multicolumn{2}{|c|}{"21FE03"} \\
\hline
\end{tabular}
\begin{tabular}{l|l|}
\hline Var_RO[0] & 3 \\
\cline { 2 - 3 } & \\
\cline { 2 - 2 } & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline \multirow[t]{3}{*}{} & \multicolumn{2}{|c|}{Before execution} \\
\hline & \multicolumn{2}{|l|}{b15-------------b8b7--------------b0} \\
\hline & 53H (S) & 55\% (U) \\
\hline & 59н (Y) & 43н (C) \\
\hline & 31н (1) & 5Ан (Z) \\
\hline \[
\|
\] & 42H (B) & 30н (0) \\
\hline & \multicolumn{2}{|c|}{00\%} \\
\hline \[
1 \mid
\] & \multicolumn{2}{|c|}{"USCYZ10B"} \\
\hline  & \multicolumn{2}{|c|}{After execution} \\
\hline Var_D100 & 53H (S) & 55\% (U) \\
\hline \(\square\) & 31н (1) & 32н (2) \\
\hline \(\longrightarrow\) & 45H (E) & 46н (F) \\
\hline & 42H (B) & 30н (0) \\
\hline & \multicolumn{2}{|c|}{00\%} \\
\hline & \multicolumn{2}{|c|}{"US21FE0B"} \\
\hline
\end{tabular}

\subsection*{7.11.17 Character string data search}

INSTR(P)



Input argument,
EN: Executing condition
:Bit
s1: Character string data to be searched, or start number of the
:String
s2:
s2: Character string data in which character string data are
:String
searched, or start number of the device that stores character
string data in which character string data are searched
n : Search start position
ANY16
Output argument,
ENO: Execution result
:Bit
\(\mathrm{d}: \quad\) Start number of the device that stores the search result
:ANY16
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J管,} & \multirow[b]{2}{*}{} & \multirow[b]{2}{*}{Zn} & \multicolumn{2}{|l|}{Constant} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & K, H & \$ & \\
\hline (51) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & \(\bigcirc\) & - \\
\hline (3) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & \(\bigcirc\) & - \\
\hline n & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - & - \\
\hline (d) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & - & - & - \\
\hline
\end{tabular}

\section*{\(\sqrt{3}\) Function}
(1) Searches for the character string data specified for (51) in the devices from the nth character from the left of the character string data specified for \(\Im_{2}\) and stores the result to the device specified for © .
As the result of search, the location of match, counted in the number of characters from the first character of the character string data specified for \(\S_{(2)}\), is stored.
When \(\mathrm{n}=3\)

(d) 5 Stores the position of the found character, counted by the number of characters from the 1st character in the character string data specified for (22).
(2) If there is no matching character string data, stores 0 to (d).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of \(n\) exceeds the number of characters in \(\Omega_{2}\).
(Error code: 4100)
- 00 H (NULL) does not exist within the corresponding device range following the devices specified for (s1) and ©2).
(Error code: 4100)
- When n is a negative value or 0 .

\section*{\(\square\) Program Example}
(1) In the following program, the character string data stored in Var_D0 and the following devices are searched from the 5th character from the left of the character string data stored in Var_R0 and the following devices when X0 turns ON, and the result is stored to Var_D100.
[Structured ladder/FBD]


\section*{[ST]}

INSTRP(X0,Var_D0,Var_R0,5,Var_D100);
[Operation]


Stores "0" because there are no matches.
(2) In the following program, the character string data " AB " is searched from the 3rd character from the left of the character string data stored in Var_D0 and the following devices when X1C turns ON, and the result is stored to Var_D100.
[Structured ladder/FBD]


\section*{[ST]}

INSTRP(X0,"AB",Var_D0,3,Var_D100);
[Operation]


\subsection*{7.11.18 Character string data insert}

STRINS

- \(\mathrm{QnU}(\mathrm{D})(\mathrm{H}) \mathrm{CPU}\) and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported

STRINS(P)
P: Executing condition
?a) indicates any of the following
instructions.
STRINS
STRINSP


Input argument,
EN: Executing condition
:Bit
s: Character string data to be inserted, or start number of the
device in which character string data are stored
\(\mathrm{n}: \quad\) Insert position (Setting range \(1 \leqq \mathrm{n} \leqq 16383\) )
Output argument,
ENO: Execution result
:Bit
\(\mathrm{d}: \quad\) Start number of the device that stores character string data
:String
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J......} & \multirow[b]{2}{*}{U} & \multirow[b]{2}{*}{Zn} & \multicolumn{2}{|l|}{Constant} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & K, H & \$ & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & \(\bigcirc\) & - \\
\hline n & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Inserts the character string data specified for © to the nth character (insert position) from the start of the character string data specified for (d).
When the insert position \(n=3\)


Shifts the character string from the 3rd character to the left for the number of characters of (s), and inserts the character string data of s.

Insert position at
the 3rd character

(d) +5 and the following data equal to the number of data to be inserted are over written.
(2) If the number of characters inserted (s + © ) is an even number, the NULL code ( 00 H ) is stored to the device ( 1 word) after the last device of the character string data.
(3) If the number of characters inserted ( + (d) ) is an odd number, the NULL code (00H) is stored to the last device (upper 8 bits) of the character string data.
(4) If the number of characters of © +1 is specified for \(n\), the character string © (s connected to the last of the character string (d).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The number of characters of the character string (s), the character string (d) or the inserted character string (© + (d) exceeds 16383.
(Error code: 4100)
- n is not within the range. \((1 \leqq \mathrm{n} \leqq 16383)\)
(Error code: 4100)
- The value specified for \(n\) exceeds the number of characters of the character string © +1 .
(Error code: 4100)
- Any device of character string © and the character string © overlaps. (Error code: 4101)
- The inserted character string data (s) + (d) exceed the range of the specified device.
(Error code: 4101)
- The NULL code \((00 \mathrm{H})\) does not exist within the specified device range following the devices specified for (s) and (d).
(Error code: 4101)
- The inserted character string data (s) + (d) overlap with the device that stores the character string (s).
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the character string data stored in D0 and the following devices are inserted to the 4th character from the start of the character string data of device D20, when M0 turns ON.
[Structured ladder/FBD]

[ST]
STRINS(M0,D0,K4,D20);
[Operation]


Insert data between "0" and "G" of character string D20

\begin{tabular}{|c|c|c|}
\hline \multirow[b]{2}{*}{D20} & \multicolumn{2}{|c|}{After execution} \\
\hline & 52н (R) & 50н (P) \\
\hline \(\rightarrow\) D21 & 35н (5) & 4FH (O) \\
\hline D22 & 34 \({ }^{\text {(4) }}\) & 38H (8) \\
\hline D23 & 52H (R) & 47\% (G) \\
\hline D24 & 4D (M) & 41H (A) \\
\hline D25 & 42н (B) & 41н (A) \\
\hline D26 & 44 \({ }^{\text {(D) }}\) & 43н (C) \\
\hline D27 & & \\
\hline
\end{tabular}

Character string \begin{tabular}{|l|l|l|}
\hline PRO & 58 & GRAMABCD \\
\hline
\end{tabular} of D20

\subsection*{7.11.19 Character string data delete}

\section*{STRDEL}

- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported

\section*{STRDEL(P)}


Input argument,
\begin{tabular}{llc} 
EN: & Executing condition & \(:\) Bit \\
\(\mathrm{n} 1:\) & Deletion start position (Setting range \(1 \leqq \mathrm{n} \leqq 16383)\) & \(:\) ANY16 \\
\(\mathrm{n} 2:\) & Number of characters to be deleted (Setting range \(0 \leqq \mathrm{n} 2 \leqq:\) ANY16 \\
& \begin{tabular}{l}
\(16384-\mathrm{n} 1)\)
\end{tabular} \\
ENO: & \begin{tabular}{l} 
Execution result \\
\(\mathrm{d}:\)
\end{tabular} & \begin{tabular}{l} 
Start number of the device that stores character string data to \\
be deleted
\end{tabular}
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...an} & \multirow[b]{2}{*}{U...iga} & \multirow[t]{2}{*}{Zn} & Constant & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & K, H & \\
\hline n1 & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline n2 & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Deletes n 2 characters from n 1 th character (deletion start position) from the start of the character string data specified for (d).
When the deletion position \(n 1=3\), and the number of deletion characters \(n 2=5\)

(2) After the deletion, if the character string © is an even number, the NULL code (00H) is stored to the device (1 word) after the last device of the character string data.
(3) After the deletion, if the character string © is an odd number, the NULL code \((00 \mathrm{H})\) is stored to the last device (upper 8 bits) of the character string data.
(4) After shifting the data following the deleted character string for n 2 characters to the right, the NULL codes \((00 \mathrm{H})\) are stored to the empty devices.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The number of characters of the character string © exceeds 16383. (Error code: 4100)
- n 1 is not within the range. \((1 \leqq \mathrm{n} 1 \leqq 16383)\)
- The value specified for n 1 exceeds the number of characters of the character string (d).
(Error code: 4100)
- The value specified for n 2 exceeds the number of characters from n 1 to the last character of the character string (a).
- A negative value is specified for n 2 .

\section*{Program Example}
(1) In the following program, the character string data stored in D0 and the following devices are deleted from the 4th character for 7 characters, when MO turns ON.
[Structured ladder/FBD]


\section*{[ST]}

STRDEL(M0,K4,K7,D0);
[Operation]


\subsection*{7.11.20 Floating-point data to BCD format conversion}


Input argument,
\(\left.\begin{array}{ll}\text { EN: } & \text { Executing condition } \\
\text { s1: } & \text { 32-bit floating-point real number data, or start number of the } \\
\text { device that stores 32-bit floating-point real number data }\end{array}\right\}\)\begin{tabular}{ll} 
s2: & Fractional part data \\
ENO: & Execution result \\
d: & \begin{tabular}{l} 
Start number or the array of the device that stores data \\
converted to BCD format
\end{tabular}
\end{tabular}
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{\(\square\) indicates any of the following instructions.} \\
\hline EMOD & EMODP \\
\hline
\end{tabular}
:Bit
:Single-precision real
:ANY16
:Bit
\(\begin{array}{lll}\text { Output argument, } & \text { ENO: Execution result } \\ \mathrm{d}: & \text { Start number or the array of the device that stores data }\end{array}\)
:Array of ANY16 (1..5) converted to BCD format
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J.alin} & \multirow[b]{2}{*}{U...|c:...} & \multirow{2}{*}{Zn} & \multicolumn{2}{|l|}{Constant} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & K, H & E & \\
\hline (11) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{2}{|l|}{-} & \(\bigcirc\) & - \\
\hline (2) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{2}{|l|}{\(\bigcirc\)} & - & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & & - & & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Converts the 32-bit floating-point real number data specified for (s1) to 32-bit floating-point data in BCD format based on the fractional part digits specified for ©s2), and stores the result to the device specified for (d) and the following devices.

(s2) specifies the fractional part digits of the 32-bit floating-point real number data (s1) as shown below.
Values between 0 and 7 can be specified for (s2).
3.25427
( 14
(s2) \(=3\)

(2) The 7th digit of the significant figure of BCD stored in © [1] and © [2] is rounded off to make a 6-digit number.

(3) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{OOperation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The fractional part digits specified for \(\Theta_{2}\) ) is outside the range of 0 to 7 .
(Error code: 4100)
- The device range specified for (d) exceeds the range of the corresponding device.
(Error code: 4101)
- The 32-bit floating-point real number specified for (s1) is outside the following range.
\(0,2^{-126} \leqq \mid\) value of the specified device \(\mid<2^{128}\)
(Error code: 4100)
- The device specified for (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)
- The value of the specified device is -0 , unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)

\section*{\(\square\) Program Example}

In the following program, the 32-bit floating-point real number data in g_real1 is converted to BCD format according to the fractional part digits as specified for g_int1 when g_bool1 turns ON, and stores the results to D100 and the following devices.
[Structured ladder/FBD]


\section*{[ST]}
g_real1:=-0.987654;
g_int1:=3;
EMOD(g_bool1,g_real1,g_int1,D100);
[Operation]

g_int1 \(\qquad\)
\begin{tabular}{c|c|} 
D100 & 1 \\
\cline { 2 - 2 } D101 & 9876540 H \\
D102 & \\
D103 & 1 \\
D104 & 4 \\
\hline
\end{tabular}

\subsection*{7.11.21 BCD format to floating-point data conversion}

\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{-} \\
\hline instructio & \\
\hline EREXP & EREXPP \\
\hline
\end{tabular}


\section*{Function}
(1) Converts the 32-bit floating-point data in BCD format specified for (s1) to the floating-point real number data according to the fractional part digits specified for \(\Omega_{2}\), and stores the result to the device specified for (d) and the following devices.

(2) The sign in \((51)\) and the sign for the exponent part in \((91+3\) is set to 0 for a positive value and to 1 for a negative value.
(3) 0 to 38 can be set for the BCD exponent of (91) +4 .
(4) 0 to 7 can be set for the fractional part digits of \(\AA_{2}\).


\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The format specified for (s1) is neither 0 nor 1.
(Error code: 4100)
- Values other than 0 to 9 are specified for each digit of \((51)+1\) and \((51)+2\).
(Error code: 4100)
- The format specification specified for (s1) +3 is other than 0 or 1 .
(Error code: 4100)
- The exponent data specified for \((51)+4\) is outside the range of 0 to 38 .
(Error code: 4100)
- The fractional part digits specified for (s2) is outside the range of 0 to 7 .
(Error code: 4100)
- The device specified for (51) exceeds the corresponding device range.
(For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, the 32-bit floating-point data in BCD format stored in Var_D0 and the following devices are converted to the floating-point real number data based on the fractional part digits stored in Var_D10 when X0 turns ON, and the result is stored to Var_D100.
[Structured ladder/FBD]

[ST]
D10:=3;
EREXPP(X0,Var_D0,Var_D10,Var_D100);
[Operation]


\subsection*{7.12 Special Function Instructions}

\subsection*{7.12.1 SIN operation on floating-point data (single precision)}
- Basic model QCPU: Supported if first 5 digits of the serial number are "04122" or later
\(\operatorname{SIN}(P) \quad: \quad: \pm\).



\section*{Function}
(1) Evaluates the sine function on the angle specified for \({ }^{\text {(s) }}\), and stores the result to the device specified for (d).

(2) The angle specified for (s) is set in radians (degrees \(\times \pi / 180\) ). For conversion between degrees and radians, refer to the RAD \((P)\) and \(D E G(P)\) instructions.
(3) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the specified device is \(-0 .{ }^{* 1}\) (For Basic model QCPU and High Performance QCPU)
(Error code: 4100)
*1: There are CPU modules that will not result in an operation error if -0 is specified. For details, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
- The operation result exceeds the following range. (An overflow occurs.) (For Universal model QCPU and LCPU) \(2^{128} \leqq\) | operation result |
- The value of the specified device is -0 , unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)

\section*{Program Example}
(1) In the following program, the sine function is evaluated on the angle set in BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 32-bit floating-point real number.
[Structured ladder/FBD]


Inputs the angle data on which the sine function is evaluated. (1)

Converts the input angle data to 32-bit floating-point real number data. (2))

Converts degrees to radians. (3)

Sine operation on the converted angle data in radians. (4)
```

[ST]
BIN(SM400,K4X20,Var_D30);
FLT(SM400,Var_D30,Var_D20);
RAD(SM400,Var_D20,Var_D10);
SIN(SM400,Var_D10,Var_D0);

```
[Operation example when the value of 150 is specified for X20 to X 2 F ]


\subsection*{7.12.2 SIN operation on floating-point data (double precision)}

\begin{tabular}{llll} 
Input argument, & \(\mathrm{EN}:\) & \begin{tabular}{l} 
Executing condition \\
\(\mathrm{s}:\)
\end{tabular} & \begin{tabular}{l} 
Angle data on which the sine function is evaluated, or start \\
number of the device that stores angle data
\end{tabular}
\end{tabular} \begin{tabular}{l} 
:Bit \\
Output argument,
\end{tabular}\(\quad\)\begin{tabular}{l} 
ENO: \\
\(\mathrm{d}:\)
\end{tabular}\(\quad\)\begin{tabular}{l} 
Execution result \\
Start number of the device that stores operation result
\end{tabular}\(\quad\)\begin{tabular}{l} 
:Bit \\
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J景: 1} & \multirow[b]{2}{*}{U...ig:} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{\begin{tabular}{l}
Constant \\
E
\end{tabular}} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Evaluates the sine function on the angle specified for \({ }^{(s)}\), and stores the result to the device specified for (d).

(2) The angle specified for (s) is set in radians (degrees \(\times \pi / 180\) ).

0 or values between \(2^{-1022}\) and \(2^{1024}\) can be specified.
For conversion between degrees and radians, refer to the RADD(P) and DEGD(P) instructions.
(3) When the operation results in -0 or an underflow, the result is processed as 0 .
(4) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value of the specified device is not within the following range. (Error code: 4140)
\(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 .
(Error code: 4140)
- The operation result exceeds the following range. (An overflow occurs.) \(2^{1024} \leqq\) | operation result |
(Error code: 4141)

\section*{\(\square\) Program Example}

In the following program, the sine function is evaluated on the angle set in BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 64-bit floating-point real number.
[Structured ladder/FBD]


Inputs the angle data on which the sine function is evaluated. (1)
Converts the input angle data to 64-bit floating-point real number data. (2))

Converts degrees to radians. (3)

Sine operation on the converted angle data in radians. (4))
[ST]
BIN(SM400,K4X20,Var_D30);
FLTD(SM400,Var_D30,Var_D20);
RADD(SM400,Var_D20,Var_D10);
SIND(SM400,Var_D10,Var_D0);
[Operation example when the value of 150 is specified for X20 to X2F]



\subsection*{7.12.3 \(\operatorname{COS}\) operation on floating-point data (single precision)}
- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later


Function
(1) Evaluates the cosine function on the angle specified for © (s) and stores the result to the device specified for (d).

(2) The angle specified for (s) is set in radians (degrees \(\times \pi / 180\) ).

For conversion between degrees and radians, refer to the \(R A D(P)\) and \(D E G(P)\) instructions.
(3) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the specified device is -0 . \({ }^{* 1}\) (For Basic model QCPU and High Performance model QCPU)
(Error code: 4100)
*1: There are CPU modules that will not result in an operation error if -0 is specified. For details, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
- The operation result exceeds the following range. (An overflow occurs.) (For Universal model QCPU and LCPU) \(2^{128} \leqq\) | operation result |
(Error code: 4141)
- The value of the specified device is -0 , unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)

\section*{\(\square\) Program Example}

In the following program, the cosine function is evaluated on the angle set in BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 32-bit floating-point real number.
[Structured ladder/FBD]

[Operation example when the value of 60 is specified for X20 to X2F]


\subsection*{7.12.4 COS operation on floating-point data (double precision)}
\(\operatorname{cosD}(P)\)
P: Executing condition
\(\operatorname{CosD}(\mathrm{P}) \quad\left(\begin{array}{ll}\hline \text { P: Executing condition } & : \pm\end{array}\right)\)

\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Input argument,} & \multirow[t]{2}{*}{\[
\begin{aligned}
& \mathrm{EN}: \\
& \mathrm{s}:
\end{aligned}
\]} & \multicolumn{9}{|l|}{Executing condition : B} \\
\hline & & \multicolumn{9}{|l|}{Angle data on which the cosine function is evaluated, or start :Double-precision rea number of the device that stores angle data} \\
\hline \multirow[t]{5}{*}{Output argument,} & \begin{tabular}{l}
ENO: \\
d:
\end{tabular} & \multicolumn{5}{|l|}{\begin{tabular}{l}
Execution result \\
Start number of the device that stores operation result
\end{tabular}} & \multicolumn{4}{|l|}{\begin{tabular}{l}
:Bit \\
:Double-precision real
\end{tabular}} \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...al} & \multirow{2}{*}{U} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (5) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & \(\bigcirc\) & - \\
\hline & (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Evaluates the cosine function on the angle specified for (s), and stores the result to the device specified for (d).

(2) The angle specified for (s) is set in radians (degrees \(\times \pi / 180\) ).

For conversion between degrees and radians, refer to the RADD \((P)\) and \(\operatorname{DEGD}(P)\) instructions.
(3) When the operation results in -0 or an underflow, the result is processed as 0 .
(4) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value of the specified device is not within the following range. (Error code: 4140) \(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 .
(Error code: 4140)
- The operation result exceeds the following range. (An overflow occurs.) \(2^{1024} \leqq\) | operation result |
(Error code: 4141)

\section*{\(\square\) Program Example}

In the following program, the cosine function is evaluated on the angle set in BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 64-bit floating-point real number.
[Structured ladder/FBD]

[Operation example when the value of 60 is specified for X20 to X2F]


\subsection*{7.12.5 TAN operation on floating-point data (single precision)}
- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later


\section*{Function}
(1) Evaluates the tangent function on the angle specified for (s), and stores the result to the device specified for © .

(2) The angle specified for © is set in radians (degrees \(\times \pi / 180\) ).

For conversion between degrees and radians, refer to the \(R A D(P)\) and \(D E G(P)\) instructions.
(3) When the angle specified for (s) is \(\pi / 2\) radians, or (3/2) \(\pi\) radians, evaluation differences are generated for the radian value, but it does not cause an operation error.
(4) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- Operation results are outside of the range shown below. (For Basic model QCPU and High Performance model QCPU)
\(0,2^{-126} \leqq\) |operation result \(\mid<2^{128}\)
(Error code: 4100)
- The value of the specified device is -0 . \({ }^{* 1}\) (For Basic model QCPU and High Performance model QCPU)
(Error code: 4100)
*1: There are CPU modules that will not result in an operation error if -0 is specified. For details, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
- The operation result exceeds the following range. (An overflow occurs.)
(For Universal model QCPU and LCPU)
\(2^{128} \leqq\) | operation result |
(Error code: 4141)
- The value of the specified device is -0, unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)

\section*{\(\triangle\) Program Example}

In the following program, the tangent function is evaluated on the angle set in BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 32-bit floating-point real number.
[Structured ladder/FBD]


Inputs the angle data on which the tangent function is evaluated. (1))

Converts the input angle data to 32-bit floating-point real number data. (②)

Converts degrees to radians. (3)

Tangent operation on the converted angle data in radians. (4)
[ST]
BIN(SM400,K4X20,Var_D30);
FLT(SM400,Var_D30,Var_D20);
RAD(SM400,Var_D20,Var_D10);
TAN(SM400,Var_D10,Var_D0);
[Operation example when the value of 135 is specified for X20 to X2F]


\subsection*{7.12.6 TAN operation on floating-point data (double precision)}

TAND



\section*{Function}
(1) Evaluates the tangent function on the angle specified for (s), and stores the result to the device specified for (d).

(2) The angle specified for (s) is set in radians (degrees \(\times \pi / 180\) ). For conversion between degrees and radians, refer to the RADD \((P)\) and \(\operatorname{DEGD}(P)\) instructions.
(3) When the angle specified for © is \(\pi / 2\) radians, or (3/2) \(\pi\) radians, evaluation differences are generated for the radian value, but it does not cause an operation error.
(4) When the operation results in -0 or an underflow, the result is processed as 0 .
(5) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the specified data is not within the following range.
(Error code: 4140) \(0,2^{-1022} \leqq \mid\) value of the specified data \(\mid<2^{1024}\)
- The value of the specified data is -0 .
(Error code: 4140)
- The operation result exceeds the following range. (An overflow occurs.)
\[
\begin{equation*}
2^{1024} \leqq \mid \text { operation result } \mid \tag{Errorcode:4141}
\end{equation*}
\]

\section*{\(\square\) Program Example}

In the following program, the tangent function is evaluated on the angle set in BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 64-bit floating-point real number.
[Structured ladder/FBD]


Inputs the angle data on which the tangent function is evaluated. (1)

Converts the input angle data to 64-bit floating-point real number data. (2)

Converts degrees to radians. (3)

Tangent operation on the converted angle data in radians. (4)
[ST]
BIN(SM400,K4X20,Var_D30);
FLTD(SM400,Var_D30,Var_D20);
RADD(SM400,Var_D20,Var_D10);
TAND(SM400,Var_D10,Var_D0);
[Operation example when the value of 135 is specified for X20 to X2F]


\subsection*{7.12.7 \(\mathrm{SIN}^{-1}\) operation on floating-point data (single precision)}

ASIN(P)
P: Executing condition
indicates any of the following

instructions.
ASIN ASINP

Input argument,
EN: Executing condition
Bit
s: \(\quad\) Sine values on which the arcsine function is evaluated, or start :Single-precision real number of the device that stores sine values
Output argument, ENO: Execution result :Bit
d: Start number of the device that stores operation result :Single-precision real
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J:.} & \multirow[b]{2}{*}{U...icina} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & O & - & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & O & - & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Evaluates the arcsine function on the value specified for (s) to define an angle, and stores the result to the device specified for (d).

(2) The \(\operatorname{SIN}\) value specified for (s) can be in the range from -1.0 to 1.0 .
(3) The angle (operation result) stored to (d) is stored in radians.

For more information on the conversion between radians and degrees, refer to the \(D E G(P)\) and \(\operatorname{RAD}(P)\) instructions.
(4) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value specified for (s) is outside the range of -1.0 to 1.0 .
(Error code: 4100)
- The value of the specified device is outside the following range.
(For Universal model QCPU and LCPU)
\(0,2^{-126} \leqq \mid\) value of the specified device \(\mid<2^{128}\)
(Error code: 4140)
- The value of the specified device is \(-0 .{ }^{* 1}\)
(For High Performance model QCPU)
(Error code: 4100)
*1: There are CPU modules that will not result in an operation error if -0 is specified. For details, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
- The operation result exceeds the following range. (An overflow occurs.)
(For Universal model QCPU and LCPU)
\(2^{128} \leqq\) | operation result |
(Error code: 4141)
- The value of the specified device is -0 , unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)

\section*{\(\square\) Program Example}

In the following program, the arcsine function is evaluated on the 32-bit floating-point real number in Var_D0, and its angle is output to Y40 to Y4F in BCD 4 digits.
[Structured ladder/FBD]


Evaluates the angle data (in radians) by the arcsine operation. (①)

Converts radians to degrees. (2))

Converts the angle data in 32-bit floating-point real number to integer value. (3)

Outputs the angle data converted to integer value to the display. (4)
[ST]
ASIN(SM400,Var_D0,Var_D10);
DEG(SM400,Var_D10,Var_D20);
INT(SM400,Var_D20,Var_D30);
BCDP(SM400,Var_D30,K4Y40);
[Operation example when the value of 0.5 is set for Var_D0]


\subsection*{7.12.8 \(\mathrm{SIN}^{-1}\) operation on floating-point data (double precision)}

ASIND


ASIND(P)
P: Executing condition
4
ASIND \((P) \quad\left(\begin{array}{ll}\text { P: Executing condition } & : \uparrow\end{array}\right)\)



\section*{Function}
(1) Evaluates the arcsine function on the value specified for (s) to define an angle, and stores the result to the device specified for (d).

(2) The \(\operatorname{SIN}\) value specified for (s) can be in the range from -1.0 to 1.0 .
(3) The angle (operation result) stored to (d) is stored in radians.

For more information on the conversion between radians and degrees, refer to the DEGD(P) and RADD \((P)\) instructions.
(4) When the operation results in -0 or an underflow, the result is processed as 0 .
(5) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the specified device is not within the following range. (Error code: 4140) \(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 .
(Error code: 4140)
- The value specified for (s) is within the double-precision floating-point range and outside the range of -1.0 to 1.0 .
(Error code: 4100)
- The operation result exceeds the following range. (An overflow occurs.) \(2^{1024} \leqq\) | operation result |
(Error code: 4141)

\section*{\(\triangle\) Program Example}

In the following program, the arcsine function is evaluated on the 64-bit floating-point real number in Var_D0, and its angle is output to Y 40 to Y 4 F in BCD 4 digits.
[Structured ladder/FBD]


Evaluates the angle data (in radians) by the arcsine operation. (1)

Converts radians to degrees. (2))
Converts the angle data in 64-bit floating-point real number to integer value. (3)

Outputs the angle data converted to integer value to the display. (4))
[ST]
ASIND(SM400,Var_D0,Var_D10);
DEGD(SM400,Var_D10,Var_D20);
INTD(SM400,Var_D20,Var_D30);
BCDP(SM400,Var_D30,K4Y40);
[Operation example when the value of 0.5 is set for Var_D0]


\subsection*{7.12.9 \(\mathrm{COS}^{-1}\) operation on floating-point data (single precision)}

ACOS(P)
P: Executing condition
4


Input argument,
EN: Executing condition
Bit
\(\mathrm{s}: \quad\) Cosine values on which the arccosine function is evaluated, or :Single-precision real start number of the device that stores cosine values
Output argument, ENO: Execution result :Bit
d: Start number of the device that stores operation result :Single-precision real
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{} & \multirow[b]{2}{*}{U")} & \multirow[b]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & - & & & - & & O & - & \(\bigcirc\) & - \\
\hline (d) & - & & & - & & O & - & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Evaluates the arccosine function on the value specified for ©s to define an angle, and stores the result to the device specified for (d).

(2) The \(\operatorname{COS}\) value specified for (s) can be in the range from -1.0 to 1.0.
(3) The angle (operation result) stored to (d) is stored in radians.

For more information on the conversion between radians and degrees, refer to the DEG(P) and \(R A D(P)\) instructions.
(4) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value specified for (s) is outside the range of -1.0 to 1.0.
(Error code: 4100)
- The value of the specified device is outside the following range.
(For Universal model QCPU and LCPU)
\(0,2^{-126} \leqq \mid\) value of the specified device \(\mid<2^{128}\)
(Error code: 4140)
- The value of the specified device is -0 . \({ }^{* 1}\)
(Error code: 4100)
(For High Performance model QCPU)
*1: There are CPU modules that will not result in an operation error if -0 is specified. For details, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
- The operation result exceeds the following range. (An overflow occurs.)
(For Universal model QCPU and LCPU)
\(2^{128} \leqq \mid\) operation result |
(Error code: 4141)
In the following program, the arccosine function is evaluated on the 32-bit floating-point real number in Var_D0, and its angle is output to Y40 to Y4F in BCD 4 digits.

\section*{Program Example}

In the following program, the arccosine function is evaluated on the 32-bit floating-point real number in Var_D0, and its angle is output to Y 40 to Y 4 F in BCD 4 digits.
[Structured ladder/FBD]


Evaluates the angle data (in radians) by the arccosine operation. (1))

Converts radians to degrees. (2))

Converts the angle data in 32-bit floating-point real number to integer value. (3)

Outputs the angle data converted to integer value to the display. (4))
[ST]
ACOS(SM400,Var_D0,Var_D10);
DEG(SM400,Var_D10,Var_D20);
INT(SM400,Var_D20,Var_D30);
BCDP(SM400,Var_D30,K4Y40);
[Operation example when the value of 0.5 is set for Var_D0]
 real number

\subsection*{7.12.10 \(\mathrm{COS}^{-1}\) operation on floating-point data (double precision)}
\(\operatorname{ACOSD}(\mathrm{P})\)
P: Executing condition
4
indicates any of the following

instructions.
ACOSD ACOSDP

Input argument,
EN: Executing condition
:Bit
\(\mathrm{s}: \quad\) Cosine values on which the arccosine function is evaluated, or :Double-precision real start number of the device that stores cosine values
Output argument,
ENO: Execution result :Bit
\(\mathrm{d}: \quad\) Start number of the device that stores operation result :Double-precision real
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..A)} & \multirow[b]{2}{*}{U} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Evaluates the arccosine function on the value specified for ©s to define an angle, and stores the result to the device specified for (d).

(2) The COS value specified for (s) can be in the range from -1.0 to 1.0.
(3) The angle (operation result) stored in (d) is stored in radians.

For more information on the conversion between radians and degrees, refer to the DEGD(P) and RADD \((P)\) instructions.
(4) When the operation results in -0 or an underflow, the result is processed as 0 .
(5) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the specified device is not within the following range. (Error code: 4140) \(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 .
(Error code: 4140)
- The value specified for (s) is within the double-precision floating-point range and outside the range of -1.0 to 1.0 .
(Error code: 4100)
- The operation result exceeds the following range. (An overflow occurs.) \(2^{1024} \leqq\) | operation result |
(Error code: 4141)

\section*{\(\triangle\) Program Example}

In the following program, the arccosine function is evaluated on the 64-bit floating-point real number in Var_D0, and its angle is output to Y 40 to Y 4 F in BCD 4 digits.
[Structured ladder/FBD]


Evaluates the angle data (in radians) by the arccosine operation. (1))

Converts radians to degrees. (②)
Converts the angle data in 64-bit floating-point real number to integer value. (3)

Outputs the angle data converted to integer value to the display. (4)
[ST]
ACOSD(SM400,Var_D0,Var_D10);
DEGD(SM400,Var_D10,Var_D20);
INTD(SM400,Var_D20,Var_D30);
BCDP(SM400,Var_D30,K4Y40);
[Operation example when the value of 0.5 is set for Var_D0]


\subsection*{7.12.11 TAN \(^{-1}\) operation on floating-point data (single precision)}

ATAN(P)
P: Executing condition

\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{\multirow[t]{2}{*}{\(\qquad\) indicates any of the following instructions.}} \\
\hline & \\
\hline ATAN & ATANP \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Input argument,} & \multirow[t]{2}{*}{\begin{tabular}{l}
EN: \\
s:
\end{tabular}} & \multicolumn{9}{|l|}{Executing condition} \\
\hline & & \multicolumn{9}{|l|}{Tangent values on which the arctangent function is evaluated, or :Single-precision real start number of the device that stores tangent values} \\
\hline \multirow[t]{6}{*}{Output argument,} & ENO: & \multicolumn{5}{|l|}{Execution result} & :Bit & & & \\
\hline & d: & \multicolumn{5}{|l|}{Start number of the device that stores operation result} & \multicolumn{2}{|l|}{:Single-precision real} & & \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...a)} & \multirow[b]{2}{*}{Uning} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (s) & - & & & - & & & - & \(\bigcirc\) & - \\
\hline & (d) & - & & & - & & & - & - & - \\
\hline
\end{tabular}

Function
(1) Evaluates the arctangent function on the value specified for ©s to define an angle, and stores the result to the device specified for (d).

(2) The angle (operation result) stored to (d) is stored in radians.

For more information on the conversion between radians and degrees, refer to the \(D E G(P)\) and \(\operatorname{RAD}(P)\) instructions.
(3) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value of the specified device is outside the following range.
(For Universal model QCPU and LCPU)
\(0,2^{-126} \leqq \mid\) value of the specified device \(\mid<2^{128}\)
(Error code: 4140)
- The value of the specified device is -0 . \({ }^{* 1}\) (For High Performance model QCPU)
(Error code: 4100)
*1: There are CPU modules that will not result in an operation error if -0 is specified. For details, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
- The operation result exceeds the following range. (An overflow occurs.) (For Universal model QCPU and LCPU) \(2^{128} \leqq\) | operation result |
(Error code: 4141)
- The value of the specified device is -0 , unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)

\section*{\(\triangle\) Program Example}

In the following program, the arctangent function is evaluated on the 32-bit floating-point real number in Var_D0, and its angle is output to Y40 to Y4F in BCD 4 digits.
[Structured ladder/FBD]


Evaluates the angle data (in radians) by the arctangent operation. (1)

Converts radians to degrees. (2))

Converts the angle data in 32-bit floating-point real number to integer value. (3))

Outputs the angle data converted to integer value to the display. (4))
[ST]
ATAN(SM400,Var_D0,Var_D10);
DEG(SM400,Var_D10,Var_D20);
INT(SM400,Var_D20,Var_D30);
BCDP(SM400,Var_D30,K4Y40);
[Operation example when the value of 1 is set for Var_D0]


\subsection*{7.12.12 \(\mathrm{TAN}^{-1}\) operation on floating-point data (double precision)}

ATAND


ATAND(P)
P: Executing condition
\(\operatorname{ATAND}(P) \quad(\) P: Executing condition \(\quad: \uparrow \quad)\)

\begin{tabular}{lll} 
Input argument, & EN: & Executing condition \\
\(\mathrm{s}:\) & \begin{tabular}{l} 
Tangent values on which the arctangent function is evaluated, or : Double-precision real \\
start number of the device that stores tangent values
\end{tabular} & :Bit \\
Output argument, & ENO: & \begin{tabular}{l} 
Execution result \\
Start number of the device that stores operation result
\end{tabular} \\
& dable-precision real
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...} & \multirow[b]{2}{*}{U} & \multirow[b]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Evaluates the arctangent function on the value specified for © to define an angle, and stores the result to the device specified for © .

(2) The angle (operation result) stored to (d) is stored in radians.

For more information on the conversion between radians and degree, refer to the DEGD(P) and RADD \((P)\) instructions.
(3) When the operation results in -0 or an underflow, the result is processed as 0 .
(4) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value of the specified device is not within the following range. (Error code: 4140) \(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 .
(Error code: 4140)
- The operation result exceeds the following range. (An overflow occurs.) \(2^{1024} \leqq\) | operation result |
(Error code: 4141)

\section*{\(\triangle\) Program Example}

In the following program, the arctangent function is evaluated on the 64-bit floating-point real number in Var_D0, and its angle is output to Y 40 to Y 4 F in BCD 4 digits.
[Structured ladder/FBD]


Evaluates the angle data (in radians) by the arctangent operation. (1)

Converts radians to degrees. (2))

Converts the angle data in 64-bit floating-point real number to integer value. (3)

Outputs the angle data converted to integer value to the display. (4)
[ST]
ATAND(SM400,Var_D0,Var_D10); DEGD(SM400,Var_D10,Var_D20); INTD(SM400,Var_D20,Var_D30);
BCDP(SM400,Var_D30,K4Y40);
[Operation example when the value of 1 is set for Var_D0]


\subsection*{7.12.13 Degree to radian conversion on floating-point data (single precision)}
- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later

\begin{tabular}{|c|c|c|c|}
\hline \multirow[t]{2}{*}{Input argument,} & EN: & Executing condition & :Bit \\
\hline & s: & Angle data whose unit is converted from degrees to radia start number of the device that stores angle data & :Single-precision real \\
\hline \multirow[t]{2}{*}{Output argument,} & ENO: & Execution result & :Bit \\
\hline & d: & Start number of the device that stores converted value in & :Single-precision real \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...alin} & \multirow[b]{2}{*}{U...igat} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & \(\bigcirc\) & - & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & \(\bigcirc\) & - & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Converts the unit of angle from degrees specified for © to radians, and stores the result to the device specified for (d).

(2) Conversion from degrees to radians is performed according to the following equation.
\[
\text { Radians }=\text { Degrees } x \frac{\pi}{180}
\]
(3) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value of the specified device is outside the following range.
(For Universal model QCPU and LCPU)
\(0,2^{-126} \leqq \mid\) value of the specified device \(\mid<2^{128}\)
(Error code: 4140)
- The value of the specified device is -0 . \({ }^{* 1}\)
(For Basic model QCPU and High Performance model QCPU)
(Error code: 4100)
*1: There are CPU modules that will not result in an operation error if - 0 is specified. For details, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
- The operation result exceeds the following range. (An overflow occurs.) (For Universal model QCPU and LCPU) \(2^{128} \leqq\) | operation result |
- The value of the specified device is -0 , unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)

\section*{\(\square\) Program Example}

In the following program, the angle (in degrees) set in BCD 4 digits in the devices from X20 to X2F is converted to radians, and the result is stored to Var_D20 as a floating-point real number.
[Structured ladder/FBD]


Inputs the angle data to be converted to radians. (①)

Converts the input angle data to 32-bit floating-point real number data. (2))

Converts degrees to radians. (3)

> [ST]
> BIN(SM400,_K4X20,Var_D0);
> FLT(SM400,Var_D0,Var_D10);
> RAD(SM400,Var_D10,Var_D20);
[Operation example when the value of 120 is specified for X20 to X2F]


\subsection*{7.12.14 Degree to radian conversion on floating-point data (double precision)}

RADD


\(\begin{array}{lll}\text { Input argument, } & \text { EN: } & \text { Executing condition } \\ \mathrm{s}: & \begin{array}{l}\text { Angle data whose unit is converted from degrees to radians, or :Double-precision real } \\ \text { start number of the device that stores angle data }\end{array} \\ \text { Output argument, } & \text { ENO: } & \begin{array}{l}\text { Execution result } \\ \mathrm{d}:\end{array} \\ \begin{array}{ll}\text { Start number of the device that stores converted value in } \\ \text { radians }\end{array} & : \text { : Dit }\end{array}\)
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J:..al} & \multirow[b]{2}{*}{U:..ici...} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Converts the unit of angle from degrees specified for © to radians, and stores the result to the device specified for (d).

(2) Conversion from degrees to radians is performed according to the following equation.
\[
\text { Radians }=\text { Degrees } x \frac{\pi}{180}
\]
(3) When the operation results in -0 or an underflow, the result is processed as 0 .
(4) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the specified device is not within the following range. (Error code: 4140) \(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 .
(Error code: 4140)
- The operation result exceeds the following range. (An overflow occurs.)
\[
\begin{equation*}
2^{1024} \leqq \mid \text { operation result } \mid \tag{Errorcode:4141}
\end{equation*}
\]

\section*{\(\triangle\) Program Example}

In the following program, the angle (in degrees) set in BCD 4 digits in the devices from X20 to X2F is converted to radians, and the result is stored to Var_D20 as a 64-bit floating-point real number.
[Structured ladder/FBD]


Inputs the angle data to be converted to radians. (1)

Converts the input angle data to 64-bit floating-point real number data. (2))

Converts degrees to radians. (3)

> [ST]
> BIN(SM400,K4X20,Var_D0);
> FLTD(SM400,Var_D0,Var_D10);
> RADD(SM400,Var_D10,Var_D20);
[Operation example when the value of 120 is specified for X20 to X2F]


\subsection*{7.12.15 Radian to degree conversion on floating-point data (single precision)}
- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later

\begin{tabular}{|c|c|c|c|}
\hline \multirow[t]{2}{*}{Input argument,} & EN: & Executing condition & :Bit \\
\hline & s : & Angle data whose unit is converted from radians to degree start number of the device that stores radians angle & :Single-precision real \\
\hline \multirow[t]{2}{*}{Output argument,} & ENO: & Execution result & :Bit \\
\hline & d: & Start number of the device that stores converted value in degrees & :Single-precision real \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...al} & \multirow[b]{2}{*}{Ulala} & \multirow[b]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & - & & & - & & O & - & \(\bigcirc\) & - \\
\hline (d) & - & & & - & & O & - & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Converts the unit of angle from radians specified for (s) to degrees, and stores the result to the device specified for (d).

(2) The conversion from radians to degrees is performed according to the following equation.
\[
\text { Degrees }=\text { Radians } \times \frac{180}{\pi}
\]
(3) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value of the specified device is -0 . \({ }^{* 1}\) (For Basic model QCPU and High Performance model QCPU)
(Error code: 4100)
*1: There are CPU modules that will not result in an operation error if -0 is specified. For details, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
- The operation result exceeds the following range. (An overflow occurs.) (For Universal model QCPU and LCPU) \(2^{128} \leqq\) | operation result |
- The value of the specified device is -0 , unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)

\section*{\(\square\) Program Example}

In the following program, the radian value set as a 32-bit floating-point real number in Var_D20 is converted to degrees, and the result is output to Y 40 to Y 4 F as a BCD value.
[Structured ladder/FBD]


Converts radians to degrees. (1)

Converts the angle data in 32-bit floating-point real number to integer value. (2))

Outputs the angle data converted to integer value to the display. (3)
[ST]
DEG(SM400,Var_D20,Var_D10);
INT(SM400,Var_D10,Var_D0);
BCDP(SM400,Var_D0,K4Y40);
[Operation example when the value of 1.435792 is set for Var_D20]
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline Var_D20 & (1)Angle conversion & Var_D10 & (2) BIN conversion & \[
\begin{array}{r}
\text { Var_D0 } \\
\text { b15-- b }
\end{array}
\] & (3) BCD conversio & & -Y40 \\
\hline 1.435792 & & 82.26482 & & 82 & \(\square\) & 0 & 0 812 \\
\hline 32-bit floating-point real number & DEG & 32-bit floating-point real number & INT & BIN value & BCD & & D value \\
\hline
\end{tabular}

\subsection*{7.12.16 Radian to degree conversion on floating-point data (double precision)}

DEGD


\section*{Function}
(1) Converts the unit of angle from radians specified for (s) to degrees, and stores the result to the device specified for (d).
0 or values between \(2^{-1022}\) and \(2^{1024}\) can be specified for (5).

(2) The conversion from radians to degrees is performed according to the following equation.
\[
\text { Degrees }=\text { Radians } \times \frac{180}{\pi}
\]
(3) When the operation results in -0 or an underflow, the result is processed as 0 .
(4) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value of the specified device is not within the following range. (Error code: 4140) \(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 .
(Error code: 4140)
- The operation result exceeds the following range. (An overflow occurs.)
\[
2^{1024} \leqq \mid \text { operation result } \mid
\]
(Error code: 4141)

\section*{\(\triangle\) Program Example}

In the following program, the radian value set as a 64-bit floating-point real number in Var_D20 is converted to degrees, and the result is output to Y 40 to Y 4 F as a BCD value.
[Structured ladder/FBD]


Converts radians to degrees. (1)

Converts the angle data in 64-bit floating-point real number to integer value. (2))

Outputs the angle data converted to integer value to the display. (3)
```

[ST]
DEGD(SM400,Var_D20,Var_D10);
INTD(SM400,Var_D10,Var_D0);
BCDP(SM400,Var_D0,K4Y40);

```
[Operation example when the value of 1.435792 is set for Var_D20]
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline Var_D20 & (1) Angle conversion & Var_D10 & \multirow[t]{2}{*}{(2) BIN conversion} & \[
\begin{aligned}
& \text { Var_D0 } \\
& \text { b15---b0 }
\end{aligned}
\] & (3) BCD conversion & \[
\text { Y } 4 \mathrm{~F}
\] & Y40 \\
\hline 1.435792 & & 82.26482 & & 82 & \(\longrightarrow\) & \(0{ }^{0} 0\) & 8)2 \\
\hline 64-bit floating-point real number & DEGDD & t floating-point number & INTD & BIN value & BCD & \(B C D\) & value \\
\hline
\end{tabular}

\subsection*{7.12.17 Exponentiation on floating-point data (single precision)}

- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported

\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{(,................................... indicates any of the following} \\
\hline instructions. & \\
\hline POW & POWP \\
\hline
\end{tabular}

Input argument,
EN: Executing condition
:Bit
s1: Data on which the exponentiation is evaluated, or start number :Single-precision real of the device that stores data on which the exponentiation is evaluated
s2: Exponent data, or start number of the device that stores the :Single-precision real data
Output argument,

Execution result
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{} & \multirow[b]{2}{*}{U". \({ }_{\text {and }}\)} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & \multicolumn{2}{|l|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (s2) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & \multicolumn{2}{|l|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & & \multicolumn{2}{|l|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{F Function}
(1) Evaluates the exponentiation on the 32-bit floating-point real number specified for (s1) and 32-bit floating-point real number specified for \(\Im_{\Omega 2}\), and stores the result to the device specified for (d).

(2) Values which can be specified for (51) and (s2), and can be stored, are as follows:
\(0,2^{-126} \leqq \mid\) specified value (storing value) \(\mid<2^{128}\)
(3) When the operation results in -0 or an underflow, the result is processed as 0 .
(4) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value specified for \((51)\) or \((22\) is not within the following range. (Error code: 4140) \(0,2^{-126} \leqq \mid\) specified value (storing value) \(\mid<2^{128}\)
- The value specified for \((51)\) or \((22\) is -0 .
- The operation result is within the following range. (An overflow occurs.) \(2^{128} \leqq\) | operation result |

\section*{Program Example}

In the following program, the exponentiation is evaluated on the 32-bit floating-point real number in D0, D1 and the 32-bit floating-point real number in D10, D11 when X10 turns ON, and the result is stored to D20, D21.
[Structured ladder/FBD]

[ST]
EMOV(M0,E0.22,D0);
EMOV(M0,E1.2,D10);
POW(X10,D0,D10,D20);
[Operation]


\subsection*{7.12.18 Exponentiation on floating-point data (double precision)}

- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported

\section*{POWD(P)}


\section*{\(\mathcal{Y}\) Function}
(1) Evaluates the exponentiation on the 64-bit floating-point real number specified for ©(1) and 64-bit floating-point real number specified for \((22\), and stores the result to the device specified for (©).


\section*{}

\[
\text { 64-bit floating-point real number } \quad \text { 64-bit floating-point real number }
\]
(2) Values which can be specified for (51) and (2), and can be stored, are as follows:
\(0,2^{-1022} \leqq \mid\) specified value (storing value) \(\mid<2^{1024}\)
(3) When the operation results in -0 or an underflow, the result is processed as 0 .
(4) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{0 \\ Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value specified for (①) or \({ }_{(22}\) is not within the following range. (Error code: 4140) \(0,2^{-1022} \leqq \mid\) specified value (storing value) \(\mid<2^{1024}\)
- The value specified for \((11)\) or \((22\) is -0 .
(Error code: 4140)
- The operation result is within the following range. (An overflow occurs.) \(2^{1024} \leqq\) | operation result |

\section*{\(\square\) Program Example}

In the following program, the exponentiation is evaluated on the 64-bit floating-point real number in D200 to D203 and the 64-bit floating-point real number in D0 to D3 when X10 turns ON, and the result is stored to D100 to D103.
[Structured ladder/FBD]

[ST]
EDMOV(SM402,E15.6,D200);
EDMOV(SM402,E3,D0);
POWD(X10,D200,D0,D100);
[Operation]


\subsection*{7.12.19 Square root operation on floating-point data (single precision)}

- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later

\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Input argument,} & \multirow[t]{2}{*}{\begin{tabular}{l}
EN: \\
s:
\end{tabular}} & \multicolumn{5}{|l|}{Executing condition} & \multicolumn{3}{|l|}{} \\
\hline & & \multicolumn{5}{|l|}{Data on which the square root function is evaluated, or start number of the device that stores data} & \multicolumn{3}{|l|}{:Single-precision real} \\
\hline \multirow[t]{5}{*}{Output argument,} & ENO:
\[
\mathrm{d}:
\] & \begin{tabular}{l}
Execu \\
Start \(n\)
\end{tabular} & \begin{tabular}{l}
esult \\
\(r\) of the
\end{tabular} & vice that & op & result & & & \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...alin} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & \\
\hline & (5) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & \(\bigcirc\) & - & \(\bigcirc\) & - \\
\hline & (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & - & \(\bigcirc\) & - & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Evaluates the square root function on the value specified for ©s, and stores the result to the device specified for (d).

(2) Only positive values can be specified for (s). (Operation cannot be performed on negative numbers.)
(3) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value specified for (s) is a negative number.
(Error code: 4100)
- The value of the specified device is outside the following range.
(For Universal model QCPU and LCPU)
\(0,2^{-126} \leqq \mid\) value of the specified device \(\mid<2^{128}\)
(Error code: 4140)
- The value of the specified device is -0 . \({ }^{* 1}\)
(For Basic model QCPU and High Performance model QCPU) (Error code: 4100)
*1: There are CPU modules that will not result in an operation error if -0 is specified. For details, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
- The operation result exceeds the following range. (An overflow occurs.)
(For Universal model QCPU and LCPU)
\(2^{128} \leqq\) | operation result |
(Error code: 4141)
- The value of the specified device is -0 , unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)

\section*{\(\triangle\) Program Example}

In the following program, the square root function is evaluated on the value set in the BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 32-bit floating-point real number.
[Structured ladder/FBD]

```

[ST]
BIN(SM400,K4X20,Var_D20);
FLT(SM400,Var_D20,Var_D10);
SQRP(SM400,Var_D10,Var_D0);

```
[Operation example when the value of 650 is specified for X20 to X2F]


\subsection*{7.12.20 Square root operation on floating-point data (double precision)}

SQRD


SQRD (P)
P: Executing condition
4

\begin{tabular}{llll} 
Input argument, & \(\mathrm{EN}:\) & Executing condition & :Bit \\
s: & \begin{tabular}{l} 
Data on which the square root function is evaluated, or start \\
number of the device that stores data
\end{tabular} & :Double-precision real \\
Output argument, & ENO: & \begin{tabular}{l} 
Execution result
\end{tabular} & :Bit \\
& \(\mathrm{d}:\) & Start number of the device that stores operation result & :Double-precision real
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...al} & \multirow[b]{2}{*}{U} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Evaluates the square root function on the value specified for ©s, and stores the result to the device specified for (d).

(2) Only positive values can be specified for ©s. (Operation cannot be performed on negative numbers.)
(3) When the operation results in -0 or an underflow, the result is processed as 0 .
(4) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value specified for (s) is a negative number.
(Error code: 4100)
- The value of the specified device is not within the following range.
(Error code: 4140)
\(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 .
(Error code: 4140)
- The operation result exceeds the following range. (An overflow occurs.) \(2^{1024} \leqq\) | operation result |
(Error code: 4141)

\section*{\(\square\) Program Example}

In the following program, the square root function is evaluated on the value set in the BCD 4 digits in the devices from X20 to X2F, and the result is stored to Var_D0 as a 64-bit floating-point real number.
[Structured ladder/FBD]


> [ST]
> BIN(X0,K4X20,Var_D20);
> FLTD(X0,Var_D20,Var_D10);
> SQRD(XO,Var_D10,Var_D0);
[Operation example when the value of 650 is specified for X20 to X2F]


\subsection*{7.12.21 Exponential operation on floating-point data (single precision)}

EXP

- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later

\begin{tabular}{|l|l|}
\hline EXP & EXPP \\
\\
\\
\\
\\
\\
\\
\\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Input argument,} & EN: & \multicolumn{5}{|l|}{Executing condition} & \multicolumn{4}{|l|}{} \\
\hline & s : & \multicolumn{5}{|l|}{Data on which the exponential function is evaluated, or start number of the device that stores data} & \multicolumn{4}{|l|}{:Single-precision real} \\
\hline \multirow[t]{5}{*}{Output argument,} & \[
\begin{aligned}
& \text { ENO: } \\
& \text { d: }
\end{aligned}
\] & \multicolumn{5}{|l|}{Execution result} & \multicolumn{4}{|l|}{:Bit} \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..n:} & \multirow[b]{2}{*}{U...ical} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (5) & - & & & - & \(\bigcirc\) & & - & \(\bigcirc\) & - \\
\hline & (d) & - & & & - & \(\bigcirc\) & & - & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Evaluates the exponential function on the value specified for \(\subseteq\), and stores the result to the device specified for (d).

(2) Exponential operation is evaluated taking the base (e) as 2.71828 .
(3) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- Operation result is not within the following range.
(Error code: 4100) \(2^{-126} \leqq \mid\) operation result | \(\leqq 2^{128}\) (For High Performance model QCPU) \(2^{-126} \leqq\) | operation result | < \(2^{128}\) (For Basic model QCPU)
- The value of the specified device is -0 . \({ }^{* 1}\) (For Basic model QCPU and High Performance model QCPU)
(Error code: 4100)
*1: There are CPU modules that will not result in an operation error if -0 is specified. For details, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
- The operation result exceeds the following range. (An overflow occurs.) (For Universal model QCPU and LCPU) \(2^{128} \leqq\) | operation result |
(Error code: 4141)
- The value of the specified device is -0, unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)

\section*{\(\triangle\) Program Example}

In the following program, the exponential function is evaluated on the value set in BCD 2 digits in the devices from X20 to X27, and the result is stored to Var_D0 as a 32-bit floating-point real number.
[Structured ladder/FBD]

```

[ST]
BIN(X0,K3X20,Var_D20);
OUT(Var_D20>89,M0);
FLT(NOT (M0),Var_D20,Var_D10);
EXP(NOT (M0),Var_D10,Var_D0);

```
[Operation example when the value of 13 is specified for X20 to X27]

*2: The operation result will be under 2129 if the \(B C D\) value of X 20 to X 27 is less than 89 , from the calculation loge \(2129=89.4\).
Because setting a value of over 90 will return an operation error, turn MO ON if a value of over 90 has been set to avoid the error.

\section*{®POINT}

Conversion from natural logarithm to common logarithm In the CPU module, a natural logarithm operation is performed.
To obtain a common logarithm value, enter a common logarithm value divided by 0.43429 in (s) .
\[
10^{x}=e^{\frac{x}{0.43429}}
\]

\subsection*{7.12.22 Exponential operation on floating-point data (double precision)}

EXPD



\section*{Function}
(1) Evaluates the exponential function on the value specified for \((s)\), and stores the result to the device specified for (d).

(2) Exponential operation is evaluated taking the base (e) as 2.71828 .
(3) When the operation results in -0 or an underflow, the result is processed as 0 .
(4) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value of the specified device is not within the following range. (Error code: 4140)
\(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 .
(Error code: 4140)
- The operation result exceeds the following range. (An overflow occurs.) \(2^{1024} \leqq \mid\) operation result |
(Error code: 4141)

\section*{Program Example}

In the following program, the exponential function is evaluated on the value set in BCD 2 digits in the devices from X20 to X31, and the result is stored to Var_D0 as a 64-bit floating-point real number.
[Structured ladder/FBD]


\section*{[ST]}

BIN(X0,K2X20,Var_D20);
OUT(Var_D20>709,M0);
IF NOT(M0) THEN
FLTD(TRUE,Var_D20,Var_D10); EXPD(TRUE,Var_D10,Var_D0);
END_IF;
[Operation example when the value of 13 is specified for X20 to X31]

*1: The operation result will be under 2129 if the BCD value of \(X 20\) to \(X 31\) is less than 89 , from the calculation loge \(2129=89.4\).
Because setting a value of over 90 will return an operation error, turn MO ON if a value of over 90 has been set to avoid the error.

XPOINT
Conversion from natural logarithm to common logarithm In the CPU module, a natural logarithm operation is performed.
To obtain a common logarithm value, enter a common logarithm value divided by
\[
\begin{aligned}
& 0.43429 \text { in © (s . } \\
& 10^{\times}=\mathrm{e}^{\frac{\mathrm{x}}{0.43429}}
\end{aligned}
\]

\subsection*{7.12.23 Natural logarithm operation on floating-point data (single precision)}
- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later


\section*{Function}
(1) Evaluates the natural logarithm function on the value specified for © taking (e) as a base, and stores the result to the device specified for (d).

(2) Only positive values can be specified for (s). (Operation cannot be performed on negative numbers.)
(3) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value specified for (s) is negative.
(Error code: 4100)
- The value specified for (s) is 0 .
(Error code: 4100)
- The value of the specified device is outside the following range.
(For Universal model QCPU and LCPU)
\(0,2^{-126} \leqq \mid\) value of the specified device \(\mid<2^{128}\)
(Error code: 4140)
- The value of the specified device is \(-0 .{ }^{* 1}\)
(For Basic model QCPU and High Performance model QCPU)
(Error code: 4100)
*1: There are CPU modules that will not result in an operation error if -0 is specified. For details, refer to MELSEC-Q/L/F Structured Programming Manual (Fundamentals).
- The operation result exceeds the following range. (An overflow occurs.)
(For Universal model QCPU and LCPU)
\(2^{128} \leqq\) | operation result |
(Error code: 4141)
- The value of the specified device is -0 , unnormalized number, nonnumeric or \(\pm \infty\). (For Universal model QCPU and LCPU)
(Error code: 4140)

\section*{\(\triangle\) Program Example}

In the following program, the natural logarithm function is evaluated on the value 10 set in D50, and the result is stored to Var_D30.
[Structured ladder/FBD]


Inputs the data on which the natural logarithm function is evaluated. (1))

Converts the input data to 32-bit floating-point real number data. (②)

Natural logarithm operation (3)
[ST]
Var_D50 :=10;
FLT(SM400,Var_D50,Var_D40);
LOG(SM400,Var_D40,Var_D30);
[Operation]


\subsection*{7.12.24 Natural logarithm operation on floating-point data (double precision)}

LOGD

LOGD(P)
P: Executing condition
4

\begin{tabular}{llll} 
Input argument, & EN: & Executing condition & :Bit \\
\(\mathrm{s}:\) & \begin{tabular}{l} 
Data on which the natural logarithm function is evaluated, or \\
start number of the device that stores data
\end{tabular} & :Double-precision real \\
Output argument, & ENO: & \begin{tabular}{l} 
Execution result
\end{tabular} & :Bit \\
& \(\mathrm{d}:\) & Start number of the device that stores operation result & :Double-precision real
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...alind} & \multirow[t]{2}{*}{U} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant E} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{\(\sqrt[3]{ }\) Function}
(1) Evaluates the natural logarithm function on the value specified for © taking (e) as a base, and stores the result to the device specified for (d).

(2) Only positive values can be specified for (s). (Operation cannot be performed on negative numbers.)
(3) When the operation results in -0 or an underflow, the result is processed as 0 .
(4) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value specified for (s) is negative. (Error code: 4100)
- The value specified for (d) is 0 .
(Error code: 4100)
- The value of the specified device is not within the following range.
(Error code: 4140) \(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value of the specified device is -0 .
(Error code: 4140)
- The operation result exceeds the following range. (An overflow occurs.) \(2^{1024} \leqq \mid\) operation result |
(Error code: 4141)

\section*{\(\triangle\) Program Example}

In the following program, the natural logarithm function is evaluated on the value 10 set in D50, and the result is stored to Var_D30.
[Structured ladder/FBD]


\section*{[ST]}
```

Var_D50 :=10;
FLTD(SM400,Var_D50,Var_D40);
LOGD(SM400,Var_D40,Var_D30);

```
[Operation]


\subsection*{7.12.25 Common logarithm operation on floating-point data (single precision)}

LOG10

- \(\mathrm{QnU}(\mathrm{D})(\mathrm{H}) \mathrm{CPU}\) and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported


\section*{Function}
(1) Evaluates the common logarithm function on the value specified for (s) taking 10 as a base, and stores the result to the device specified for (d).

(2) Only positive values can be specified for ©s. (Operation cannot be performed on negative numbers.)
(3) When the operation results in -0 or an underflow, the result is processed as 0 .
(4) If an input value is set using a programming tool, a rounding error may occur. For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value specified for © is negative. (Error code: 4100)
- The value specified for (s) is 0 .
(Error code: 4100)
- The value of the specified device is not within the following range.
(Error code: 4140)
\(0,2^{-126} \leqq \mid\) value of the specified device \(\mid<2^{128}\)
- The value specified for (s) is -0 .
(Error code: 4140)
- The operation result is within the following range. (An overflow occurs.) \(2^{128} \leqq\) | operation result |
(Error code: 4141)

\section*{\(\triangle\) Program Example}

In the following program, the common logarithm function is evaluated on the 32-bit floating-point real number stored in D600 and D601 when M0 turns ON, and the result is stored to D123 and D124.
[Structured ladder/FBD]

[ST]
EMOV(M0,E2.806,D600);
LOG10(M0,D600,D123);
[Operation]


\subsection*{7.12.26 Common logarithm operation on floating-point data (double precision)}

- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported

\begin{tabular}{|l:l|}
\hline LOG10D & LOGdicates any of the following \\
instructions. & \\
\\
\\
\\
\\
\hline
\end{tabular}


\section*{Function}
(1) Evaluates the common logarithm function on the value specified for ©s taking 10 as a base, and stores the result to the device specified for (d).

(2) Only positive values can be specified for ©s. (Operation cannot be performed on negative numbers.)
(3) When the operation results in -0 or an underflow, the result is processed as 0 .
(4) If an input value is set using a programming tool, a rounding error may occur.

For precautions when setting an input value using a programming tool, refer to the MELSEC-Q/L/F Structured Programming Manual (Fundamentals).

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value specified for © (s is negative. (Error code: 4100)
- The value specified for (s) is 0 .
(Error code: 4100)
- The value of the specified device is not within the following range.
(Error code: 4140) \(0,2^{-1022} \leqq \mid\) value of the specified device \(\mid<2^{1024}\)
- The value specified for (s) is -0 .
(Error code: 4140)
- The operation result is within the following range. (An overflow occurs.) \(2^{1024} \leqq \mid\) operation result |

\section*{Program Example}

In the following program, the common logarithm function is evaluated on the 32-bit floating-point real number stored in D600 to D603 when M0 turns ON, and the result is stored to D123 to D126.
[Structured ladder/FBD]


\section*{[ST] \\ EDMOV(M0,E2.806,D600); \\ LOG10D(M0,D600,D123);}
[Operation]


\subsection*{7.12.27 Random number generation and series update}

RND, SRND

- Basic model QCPU: Supported if first 5 digits of the serial number are " 04122 " or later
\begin{tabular}{lll} 
Input argument, & EN: & Executing condition \\
\(\mathrm{s}:\) & \begin{tabular}{l} 
Random number series data, or start number of the device \\
that stores random number series data (When using the SRND
\end{tabular} & \(:\) ANY16
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...)} & \multirow[b]{2}{*}{U...ic:...} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & \multicolumn{7}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) & \multicolumn{7}{|c|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{Function}

The random number generation instruction generates random numbers conforming to a certain calculation formula. In the calculation using the formula, the result of previous calculation is used as a coefficient.
The random series update instruction can change the random number generation pattern.

\section*{RND(P)}

Generates random numbers from 0 to 32767 , and stores them to the device specified for (d).

\section*{SRND(P)}

Updates random number series according to the 16 -bit BIN data stored in the device specified for ©s.

No operation error occurs in the execution of the RND \((P)\) and \(\operatorname{SRND}(P)\) instructions.

\section*{\(\square\) Program Example}
(1) In the following program, the random number is stored to Var_D100 when X10 turns ON. [Structured ladder/FBD]


\section*{[ST]}

RND(X10,Var_D100);
(2) In the following program, the random number series is updated according to the value of Var_D0 when X10 turns ON.
[Structured ladder/FBD]


\footnotetext{
[ST]
SRND(X10,Var_D0);
}

\subsection*{7.12.28 Square root operation on 4-/8-digit \(B C D\) data}

BSQR, BDSQR

BSQR(P)
BDSQR(P)

P: Executing condition

\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{-} \\
\hline instructio & \\
\hline BSQR & BSQRP \\
\hline BDSQR & BDSQRP \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Input argument,} & \multirow[t]{2}{*}{\begin{tabular}{l}
EN: \\
s:
\end{tabular}} & \multicolumn{9}{|l|}{Executing condition} \\
\hline & & \multicolumn{9}{|l|}{Data on which the square root function is evaluated, or start :ANY16/32 number of the device that stores data} \\
\hline \multirow[t]{5}{*}{Output argument,} & \begin{tabular}{l}
ENO: \\
d:
\end{tabular} & \multicolumn{5}{|l|}{Execution result} & \begin{tabular}{l}
:Bit \\
:ANY32
\end{tabular} & & & \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{} & \multirow{2}{*}{U-1.inc:} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (S) & \multicolumn{7}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline & (d) & \multicolumn{7}{|c|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{Function}

\section*{BSQR(P)}
(1) Evaluates the square root function on the value specified for ©s, and stores the result to the device specified for (d).

(2) Values that can be specified for (s) are BCD values with a maximum of 4 digits (from 0 to 9999).
(3) The operation results of (d) and © +1 are stored as their respective BCD values between 0 and 9999.
(4) The value of the operation result is rounded off at the fifth digit of fractional part. For this reason, the fourth digit of fractional part has an error of \(\pm 1\).

\section*{BDSQR(P)}
(1) Evaluates the square root function on the values specified for \((\) s and \((\mathrm{s})+1\) and stores the results to the device specified for (d).

(2) BCD value of a maximum of 8 digits ( 0 to 99999999 ) can be specified for (s) and \((\) s +1 .
(3) The operation results of (d) and © +1 are stored as their respective BCD values between 0 and 9999.
(4) The value of the operation result is rounded off at the fifth digit of fractional part. For this reason, the fourth digit of fractional part has an error of \(\pm 1\).

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value specified for (s) is not a BCD value.

\section*{\(\square\) Program Example}
(1) In the following program, the square root function is evaluated on 1325 in BCD value, and the integer part is output in BCD 4 digits to the devices from Y50 to Y5F, and the fractional part is output in BCD 4 digits to the devices from Y40 to Y4F.
[Structured ladder/FBD]


\section*{[ST]}

MOV(SM400,H1325,D0);
BSQR(SM400,D0,D1);
K4Y50 :=D1;
K4Y40 :=D2;
[Operation]

(2) In the following program, the square root function is evaluated on 74625813 in BCD value, and the integer part of the result is output in BCD 4 digits to the devices from Y50 to Y5F, and the fractional part is output in BCD 4 digits to the devices from Y40 to Y4F.
[Structured ladder/FBD]


Sets the data (BCD value) on which the square root function is evaluated. (1))

Square root operation (②)

Outputs the integer part of the operation result to the display. (3)

Outputs the fractional part of the operation result to the display. (4)

\section*{[ST]}

DMOV(SM400,H74625813,D0);
BDSQR(SM400,D0,D2);
K4Y50 :=D2;
K4Y40 :=D3;

\section*{[Operation]}


\subsection*{7.12.29 SIN operation on data in BCD format}

BSIN

\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{} \\
\hline instru & \\
\hline BSIN & BSINP \\
\hline
\end{tabular}
\begin{tabular}{llll} 
Input argument, & \(\mathrm{EN}:\) & Executing condition \\
S: & Data on which the sine function is evaluated, or start number & \(:\) ANY16 \\
of the device that stores data \\
Ontput argument, & \(\mathrm{ENO}:\) & \begin{tabular}{l} 
Execution result \\
Start number or the array of the device that stores the \\
operation result
\end{tabular} & \(:\) :Array of ANY16 (1..3)
\end{tabular}


\section*{\(\mathcal{W}\) Function}
(1) Evaluates the sine function on the value (angle) specified for © [0], and stores the sign of the operation result to the device specified for (d), and the operation result to the devices specified for (d) [1] and (d) [2].
(2) The value specified for (s) is a BCD value which can be between 0 and 360 degrees (in degrees).
(3) The sign for the operation result stored to © [0] will be 0 if the result is a positive value, and 1 if the result is a negative value.
(4) The operation results stored to (d) [0], © [1], and © [2] are BCD values between -10.000 and 10.000 .
(5) The value of the operation result is rounded off at the fifth digit of fractional part.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value specified for (s) is not a BCD value.
(Error code: 4100)
- The value specified for ©s is not in the range of 0 to 360 degrees. (Error code: 4100)
- The device specified for (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the sine function is evaluated on the data specified as BCD 3 digits in X 20 to X 2 B , and the integer part of the result is output in BCD 1 digit to the devices form Y50 to Y53, and the fractional part is output in BCD 4 digits to the devices form Y40 to Y4F.

Y60 is turned ON if the operation result is negative. (If a value has been set for X20 to X2F that is greater than 360, it will be adjusted to be in the range from 0 to 360 degrees.)
[Structured ladder/FBD]


Adjusts the input angle data to be in the range of 0 to 360 degrees. ()

Evaluates the sine function. (2))
Outputs the integer part of the operation result to the display. (③)
Outputs the fractional part of the operation result to the display. (4)

Outputs the sign of the operation result by turning the coil ON or OFF. (5)
[ST]
BINP(SM400,K3X20,Var_D50);
BINP(SM400,H360,Var_D51);
BCDP(SM400,Var_D50 MOD Var_D51,Var_D10);
BSIN(SM400,Var_D10,Var_D20);
K1Y50 :=Var_D20[1];
K4Y40 :=Var_D20[2];
OUT(Var_D20[0]<>0,Y60);
[Operation example when the value 590 is specified for X 20 to X 2 B ]


\subsection*{7.12.30 COS operation on data in BCD format}

BCOS


\section*{\(\{3\) Function}
(1) Evaluates the cosine function on the value (angle) specified for © [ 0 ], then stores the sign of the operation result to the word device specified for (d), and the operation result to the word device specified for (d) [1] and (d) [2].
(2) The value specified for © (s) is a BCD value which can be between 0 and 360 degrees (in degrees).
(3) The sign for the operation result stored to © [0] will be 0 if the result is a positive value, and 1 if the result is a negative value.
(4) The operation results stored to © [0], © [1], and © [2] are BCD values between - 10.000 and 10.000 .
(5) The value of the operation result is rounded off at the fifth digit of fractional part.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value specified for (s) is not a BCD value.
(Error code: 4100)
- The value specified for \({ }^{\text {© }}\) is not in the range of 0 to 360 degree. (Error code: 4100)
- The device specified for (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the cosine function is evaluated on the data specified as BCD 3 digits in X20 to X2B, and the integer part of the result is output in BCD 1 digit to the devices from Y50 to Y53, and the fractional part is output in BCD 4 digits to the devices from Y40 to Y4F.

Y 60 is turned ON if the operation result is negative.
[Structured ladder/FBD]


Adjusts the input angle data to be in the range of 0 to 360 degrees. (1)

Evaluates the cosine function. (2)
Outputs the integer part of the operation result to the display. (3)
Outputs the fractional part of the operation result to the display. (4)

Outputs the sign of the operation result by turning the coil ON or OFF. (5)
```

[ST]
BINP(SM400,K3X20,Var_D50);
BINP(SM400,H360,Var_D51);
BCDP(SM400,Var_D50 MOD Var_D51,Var_D11);
BCOS(SM400,Var_D11,Var_D20);
K1Y50 :=Var_D20[1];
K4Y40 :=Var_D20[2];
OUT(Var_D2O[0]<>0,Y60);

```
[Operation example when the value 430 is specified for X 20 to X 2 B ]


\subsection*{7.12.31 TAN operation on data in BCD format}


\section*{\(\sqrt{3}\) Function}
(1) Evaluates the tangent function on the value (angle) specified for © , and stores the sign of the operation result to the word device specified for © [0], and the operation result to the word device specified for (d) [1] and (d) [2].
(2) The value specified for (s) is a BCD value which can be between 0 and 360 degrees (in degrees).
(3) The sign for the operation result stored to (d) [0] will be 0 if the result is a positive value, and 1 if the result is a negative value.
(4) The operation results stored to (d) [1] and (d) [2] are BCD values between - 57.2901 and 57.2902.
(5) The value of the operation result is rounded off at the fifth digit of fractional part.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value specified for (s) is not a BCD value. (Error code: 4100)
- The value specified for © is not in the range of 0 to 360 degree. (Error code: 4100)
- The value specified for (s) is 90 or 270 degree.
(Error code: 4100)
- The device specified for (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)

\section*{\(\triangle\) Program Example}

In the following program, the tangent function is evaluated on the data specified as BCD 3 digits in X 20 to X 2 B , and the integer part of the result is output in the BCD 4 digits to the devices from Y50 to Y53, and the fractional part is output in BCD 4 digits to the devices from Y40 to Y4F.

Y60 is turned ON if the operation result is negative.
[Structured ladder/FBD]


Adjusts the input angle data to be in the range of 0 to 360 degrees. (1)

Use M1 for the interlock to disable the operation when the input angle data is 90 or 270 degrees.

Evaluates the tangent function. (2))
Outputs the integer part of the operation result to the display. (3))
Outputs the fractional part of the operation result to the display. (4))

Outputs the sign of the operation result by turning the coil ON or OFF. (5)
[ST]
BINP(SM400,K3X20,Var_D50);
BINP(SM400,H360,Var_D51);
BCDP(SM400,Var_D50 MOD Var_D51,Var_D10);
OUT(Var_D10=H90 OR Var_D10=H270,M1);
IF NOT(M1) THEN
BTAN(TRUE,Var_D10,Var_D20);
K1Y50 :=Var_D20[1];
K4Y40 :=Var_D20[2];
END_IF;
OUT(Var_D20[0]<>0,Y60);
[Operation example when the value 390 is specified for X 20 to X 2 B ]


\subsection*{7.12.32 \(\mathrm{SIN}^{-1}\) operation on data in BCD format}

BASIN

BASIN(P)
P: Executing condition
\&


Input argument,

Output argument,

EN: Executing condition
\(\mathrm{s}: \quad\) Data on which the arcsine function is evaluated, or start number of the device that stores data
ENO: Execution result
d: Data on which the arcsine function is evaluated, or start number or the array of the device that stores data
indicates any of the following instructions.
BASIN BASINP
:Bit
Array of ANY16 (1..3)
:Bit
:ANY16
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..ala} & \multirow[b]{2}{*}{U:...ican} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{-} & - \\
\hline (d) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{\(\bigcirc\)} & - \\
\hline
\end{tabular}

\section*{Function}
(1) Evaluates the arcsine function on the value specified for ©s and stores the operation result (angle) to the device specified for (d).
\[
\begin{array}{ccc} 
& \text { S }[0] & \text { S }[1] \\
\operatorname{SiN}^{-1}=\left(\begin{array}{ll}
\text { Sign } & \text { S }[2] \\
& \text { Integer part. } \\
& \text { Fractional part } \\
\end{array}\right)=\text { (d) }
\end{array}
\]
(2) A sign of the data to be evaluated is set for © [0].

If the evaluation value is a positive value, it stores 0 , and if it is a negative value, it stores 1 .
(3) The integer part and fractional part are stored in © [1] and © [2] respectively in BCD values. (Values between 0 and 1.0000 can be set.)
(4) Operation results stored to (d) are BCD values between 0 and 90 degrees, and 270 and 360 degrees (in degrees).
(5) The fractional part is rounded off for the operation result.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value specified for (s) is not a BCD value.
(Error code: 4100)
- The value specified for \({ }^{\text {(s }}\) is not in the range of -10.000 and 10.000. (Error code: 4100)
- The device specified for (s) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the arcsine function is evaluated on the data in which X0 with positive and negative signs (positive when X0 is OFF, and negative when X0 is ON), the integer part in BCD 1 digit from X30 to X33 and the fractional part in BCD 4 digits from X20 to X2F, and the evaluated angle is output in BCD 4 digits to the devices from Y40 to Y4F.
[Structured ladder/FBD]

[ST]
\(\operatorname{MOV}(X 0,1, \mathrm{DO})\);
MOV(NOT(X0),0,D0);
D1 :=K1X30;
D2 :=K4X20;
OUT((D1=1 AND D2<>0) OR D1>1,M0);
BASIN(NOT(MO),Var_D0,K4Y40);
[Operation example when the value 0.4753 is specified for X 20 to X 33 ]


\subsection*{7.12.33 \(\mathrm{COS}^{-1}\) operation on data in BCD format}

BACOS

BACOS(P)
P: Executing condition



Input argument,
EN: Executing condition
\(\mathrm{s}: \quad\) Data on which the arccosine function is evaluated, or start number of the device that stores data
Output argument,
\begin{tabular}{|ll|}
\hline BACOS & \\
instructions. & \\
\\
\\
\\
\\
\hline
\end{tabular}
:Bit
:Array of ANY16 (1..3)

Bit
:ANY16
\(\mathrm{d}: \quad\) Data on which the arccosine function is evaluated, or start number or the array of the device that stores data
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J)} & \multirow[b]{2}{*}{U} & \multirow[b]{2}{*}{Zn} & \multirow[b]{2}{*}{Constant} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline (d) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Evaluates the arccosine function on the value specified for (s), and stores operation result (angle) to the device specified for (d).
(2) A sign of data to be evaluated is set for © [0].

If the evaluation value is a positive value, it stores 0 , and if it is a negative value, it stores 1 .
(3) The integer part and fractional part are stored to © [1] and (s) [2] respectively in BCD values. (Values between 0 and 1.0000 can be set.)
(4) The operation results stored to © are BCD values between 0 and 180 degrees (in degrees).
(5) The fractional part is rounded off for the operation result.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value specified for (s) is not a BCD value.
(Error code: 4100)
- The value specified for \({ }^{\text {s }}\) is not in the range of -10.000 and 10.000. (Error code: 4100)
- The device specified for (s) exceeds the corresponding device range. (For Universal model QCPU and LCPU)

\section*{\(\triangle\) Program Example}

In the following program, the arccosine function is evaluated on the data in which X0 with positive and negative signs (positive when X0 is OFF, and negative when X0 is ON), the integer part in BCD 1 digit from X30 to X33 and the fractional part in BCD 4 digit from X20 to X2F, and the evaluated angle is output in BCD 4 digits to the devices from Y 40 to Y 4 F .
[Structured ladder/FBD]

[ST]
MOV(X0,1,D0);
MOV(NOT(X0),0,D0);
D1 :=K1X30;
D2 :=K4X20;
OUT((D1=1 AND D2<>0) OR D1>1,M0);
BACOS(NOT(M0),Var_D0,K4Y40);
[Operation example when the value -0.7650 is specified for X0 and X20 to X33]


\subsection*{7.12.34 TAN \({ }^{-1}\) operation on data in BCD format}

BATAN

\(\operatorname{BATAN}(P) \quad(\mathrm{P}:\) Executing condition \(\quad: \uparrow\)
\(\operatorname{BATAN}(P) \quad:\left(\begin{array}{ll}\text { P: Executing condition }\end{array}\right)\)


Input argument,
EN: Executing condition
:Bit
\(\mathrm{s}: \quad\) Data on which the arctangent function is evaluated, or start
number of the device that stores data
Output argument,
ENO: Execution result
Array of ANY16 (1..3)
d: \(\quad\) Start number of the device that stores the operation result
\begin{tabular}{|l|l|}
\hline BATAN & \\
\\
instructions. & \\
\\
\\
\\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{} & \multirow[b]{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline (d) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{\(\widehat{Y}\) Function}
(1) Evaluates the arctangent function on the value specified for (s) and stores the operation result (angle) to the device specified for (d).
(2) A sign of data to be evaluated is set for \((\mathrm{s}\) [ 0\(]\).

If the evaluation value is a positive value, it stores 0 , and if it is a negative value, it stores 1 .
(3) The integer part and fractional part are stored to (s)[1] and © [2] respectively in BCD values. (Values between 0 and 9999.9999 can be set.)
(4) Operation results stored to (d) are BCD values between 0 and 90 degrees, and 270 to 360 degrees (in degrees).
(5) The fractional part is rounded off for the operation result.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value specified for (s) is not a BCD value.
(Error code: 4100)
- The device specified for (s) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, the arctangent function is evaluated on the data in which X0 with positive and negative signs (positive when X0 is OFF, and negative when XO is ON ), the integer part in BCD 4 digits from X20 to X2F and the fractional part in BCD 4 digits from X30 to X3F, and the evaluated angle is output in BCD 4 digits to the devices from Y 40 to Y 4 F .
[Structured ladder/FBD]

[ST]
MOV (X0,1,D0);
MOV(NOT(X0),0,D0);
D1:=K4X20;
D2 :=K4X30;
BATANP(TRUE,Var_D0,K4Y40);
[Operation example when the value 1.2654X0 is specified for X0 and X20 to X2F]


\subsection*{7.13 Data Control Instructions}

\subsection*{7.13.1 Upper and lower limit controls of 16-/32-bit BIN data}
LIMIT(_P)
DLIMIT(_P)

> P: Executing condition

\section*{LIMIT(P)}
(1) Controls the output value to be stored to the device specified for (d) by checking the input value (BIN 16 bits) specified for ©3) is within the range of upper and lower limit values specified for (s1) and (52).
Output value is controlled in the way shown below.

(2) Values between -32768 and 32767 can be specified for (51), (32), and (33).
(3) To perform controls based only on the upper limit value, set the lower limit value specified for (s1) to -32678 .
(4) To perform controls based only on the lower limit value, set the upper limit value specified for (22) to 32767.

\section*{DLIMIT(P)}
(1) Controls the output value to be stored to the device specified for (d) by checking the input value (BIN 32 bits) specified for \(\left(51\right.\) and \(\Im_{2}\) ) is within the range of upper and lower limit values specified for ©3)
\begin{tabular}{|c|c|c|c|c|c|}
\hline & (s) & (3) & & (51) & (d) \\
\hline \multirow[t]{2}{*}{- When} & Lower limit value & Input value & & Lower limit value & Output value \\
\hline & (2) & (3) & & (32) & (d) \\
\hline \multirow[t]{2}{*}{- When} & Upper limit value & Input value & & Upper limit value & Output value \\
\hline & (51) & (3) & (52) & (3) & (d) \\
\hline - When & Lower limit value & Input value & Upper limit value & Input value & Output value \\
\hline
\end{tabular}

(2) Values between -2147483648 and 2147483647 can be specified for \((51)\), \(\Omega_{2}\), and ( 33 .
(3) To perform controls based only on the upper limit value, set the lower limit value specified for (51) to -2147483648.
(4) To perform controls based only on the lower limit value, set the upper limit value specified for (s2) to 2147483647 .

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The lower limit value specified for (s1) is larger than the upper limit value specified for (s2).
(Error code: 4100)

\section*{\(\triangle\) Program Example}
(1) In the following program, the limit control from 500 to 5000 is performed on the data set in BCD values from X20 to X2F when X0 turns ON, and the result is stored to Var_D1.
[Structured ladder/FBD]

[ST]
BINP(X0,K4X20,Var_D0);
LIMITP(X0,500,5000,Var_D0,Var_D1);
[Operation]
- Var_D1 becomes 500 if Var_D0 < 500.

\section*{Example Var_D0 \(=400 \rightarrow\) Var_D1 \(=500\)}
- Var_D1 becomes the value of Var_D0 when \(500 \leqq\) Var_D0 \(\leqq 5000\).

Example Var_D0 \(=1300 \rightarrow\) Var_D1 \(=1300\)
- Var_D1 becomes 5000 when 5000 < Var_D0.

\section*{Example Var_D0 \(=9600 \rightarrow\) Var_D1 \(=5000\)}
(2) In the following program, the limit control from 10000 to 1000000 is performed on the data set in BCD values from X20 to X3F when X0 turns ON, and the result is stored to Var_D10. [Structured ladder/FBD]

[ST]
DBINP(X0,K8X20,Var_D0);
DLIMIT(X0,10000,1000000,Var_D0,Var_D10);
[Operation]
- Var_D10 becomes 10000 if Var_D0 are less than 10000.

\section*{Example Var_D0 \(=400 \rightarrow\) Var_D10 \(=10000\)}
- Var_D10 becomes the value of Var_D0 if \(10000 \leqq\) Var_D0 \(\leqq 1000000\).

Example Var_D0 \(=345678 \rightarrow\) Var_D10 \(=345678\)
- Var_D10 becomes 1000000 if 1000000 < Var_D0.

Example Var_D0 \(=9876543 \rightarrow\) Var_D10 \(=1000000\)

\subsection*{7.13.2 Dead band control of 16 -/32-bit BIN data}


BAND (P)

\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{} \\
\hline instructio & \\
\hline BAND & BANDP \\
\hline DBAND & DBANDP \\
\hline
\end{tabular}

Input argument,
\begin{tabular}{ll} 
EN: & Executing condition \\
s1: & Lower limit value of the dead band (No output band) \\
\(\mathrm{s} 2:\) & Upper limit value of the dead band (No output band) \\
\(\mathrm{s} 3:\) & Input value that controls the output value by the dead band \\
& control values \\
ENO: & Execution result \\
\(\mathrm{d}:\) & Output value controlled by the dead band control values
\end{tabular}
:Bit
:ANY16/32
:ANY16/32
:ANY16/32
:Bit
:ANY16/32
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...1)} & \multirow[b]{2}{*}{U...igal} & \multirow[b]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & \multicolumn{7}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (3) & \multicolumn{7}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (3) & \multicolumn{7}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) & \multicolumn{7}{|c|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

\section*{BAND(P)}
(1) Controls the output value to be stored to the device specified for © by checking the input value (BIN 16 bits) specified for ©3 ) is within the range of dead band upper and lower limit values specified for \((51)\) and ®2 \(^{2}\).
Output value is controlled in the way shown below.

(2) Values between -32768 and 32767 can be specified for (51), (32), and (33).
(3) The output value stored to (d) is a signed 16-bit BIN value. Therefore, if the operation results exceed the range of -32768 to 32767 , the following will take place.
\[
\text { When : }\left\{\begin{array}{l}
\text { Dead band lower limit value © } ₫ 1) . . . . . . . . . . . . . . . . ~ \\
\text { Input value © } 30
\end{array}\right.
\]
\[
\text { Output value }=-32768-10=8000 \mathrm{H}-\mathrm{AH}=7 \mathrm{FF} 6 \mathrm{H}=32758
\]

\section*{DBAND(P)}
(1) Controls the output value to be stored to the device specified for © by checking the input value (BIN 32 bits) specified for ©3) is within the range of dead band upper and lower limit values specified for (51) and (52).
Output value is controlled in the way shown below.


(2) Values between -2147483648 and 2147483647 can be specified for (51), (®2), and (33).
(3) The output value stored to © is a signed 32 -bit BIN value. Therefore, if the operation results exceed the range of -2147483648 to 2147483647 , the following takes place.
\[
\begin{aligned}
& \text { Output value }=-2147483648-1000=80000000 \mathrm{H}-000003 \mathrm{E} 8 \mathrm{H} \\
& =7 \text { FFFFC } 18 \mathrm{H}=2147482648
\end{aligned}
\]

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The lower limit value specified for (①) is greater than the upper limit value specified for (®2).
(Error code: 4100)

\section*{Program Example}
(1) In the following program, the dead band control applying the lower and upper limits from 0 to 1000 is performed on the data set in BCD value at X 20 to X 2 F when X 0 turns ON, and the result is stored to Var_D1.
[Structured ladder/FBD]

[ST]
BINP(X0,K4X20,Var_D0);
BANDP(X0,0,1000,Var_D0,Var_D1);
[Operation]
- 0 is stored to Var_D1 if \(0 \leqq\) Var_D0 \(\leqq 1000\).

Example Var_D0 \(=500 \rightarrow\) Var_D1 \(=0\)
- The value of (Var_D0) - 1000 is stored to Var_D1 if \(1000<\) Var_D0.

Example Var_D0 \(=7000 \rightarrow\) Var_D1 \(=6000\)
(2) In the following program, the dead band control applying the lower and upper limits from -10000 to 10000 is performed on the data set in BCD value at Var_D0 when X0 turns ON, and the result is stored to Var_D10.
[Structured ladder/FBD]

[ST]
DBANDP(X0,-10000,10000,Var_D0,Var_D10);

\section*{[Operation]}
- The value Var_D0 - \((-10000)\) is stored to Var_D10 if Var_D0<(-10000). Example Var_D0 \(=-12345 \rightarrow\) Var_D10 \(=-2345\)
- The value 0 is stored to Var_D10 if \(-10000 \leqq\) Var_D0 \(\leqq 10000\).

Example Var_D0 \(=6789 \rightarrow\) Var_D10 \(=0\)
- The value Var_D0 - 10000 is stored to Var_D10 if 10000 < Var_D0.

Example Var_D0 \(=50000 \rightarrow\) Var_D10 \(=40000\)

\subsection*{7.13.3 Zone control of \(16-/ 32\)-bit BIN data}



Input argument,
\begin{tabular}{ll} 
EN: & Executing condition \\
s1: & Negative bias value that is added to the input value \\
s2: & Positive bias value that is added to the input value \\
s3: & Input value for executing the zone control \\
ENO: & Execution result \\
d: & Output value controlled by the zone control
\end{tabular}
:ANY16/32
:ANY16/32
:ANY16/32
:Bit
:ANY16/32


\section*{ZONE(P)}
(1) Adds a bias value specified for \((51)\) or s2 \(^{2}\) to an input value specified for \(\mathrm{S}_{3}\), and stores the result to the device specified for (d).

Bias values are calculated in the way shown below.
- When (53) Input value < \(0 \ldots \ldots\)..... (3) Input value + (51) Negative bias value \(\rightarrow\) (d) Output value

- When (33 Input value \(>0 \ldots \ldots\). (3) Input value + (52) Positive bias value \(\rightarrow\) (d) Output value Output value ( (d)

(2) Values between -32768 and 32767 can be specified for (51), (32), and (33).
(3) The output value stored to (d) is a signed 16 -bit BIN value. Therefore, if the operation results exceed the range of -32768 to 32767 , the following will take place.

Output value \(=-32768+(-100)=8000 \mathrm{H}+\mathrm{FF9C}=7 \mathrm{F9CH}=32668\)

\section*{DZONE(P)}
(1) Adds a bias value specified for \((51)\) or \((2)\) to an input value specified for \((3)\), and stores the result to the device specified for (d).
Addition of the bias value is performed in the way shown below.

(2) Values between -2147483648 and 2147483647 can be specified for (51), (2), and (33).
(3) The value stored to © is a signed 32 -bit BIN value.

Therefore, if the operation results exceed the range of -2147483648 to 2147483647 , the following takes place.
\[
\begin{aligned}
& \text { When : }\left\{\begin{array}{l}
\text { Negative bias value (31) ............................... }-1000 \\
\text { Input value (3) ................................. }-2147483648
\end{array}\right. \\
& \text { Output value }=-2147483648+(-1000)=80000000 \mathrm{H}+\text { FFFFFC18 } \mathrm{H} \\
& =7 \text { FFFFC18 }=2147482648
\end{aligned}
\]

\section*{O Operation Error}

No operation error occurs in the execution of the \(\operatorname{ZONE}(\mathrm{P})\) and \(\operatorname{DZONE}(\mathrm{P})\) instructions.

\section*{\(\boxed{P r o g r a m}\) Example}
(1) In the following program, the zone control applying negative and positive bias values of -100 to 100 is performed on the data set for Var_DO when X0 turns ON, and the result is stored to Var_D1.
[Structured ladder/FBD]

[ST]
ZONEP(X0,-100,100,Var_D0,Var_D1);
[Operation]
- The value Var_D0 + (-100) is stored to Var_D1 if Var_D0 < 0 .

Example Var_D0 \(=-200 \rightarrow\) Var_D1 \(=-300\)
- The value 0 is stored to Var_D1 if Var_D0 \(=0\).
- The value of Var_D0 +100 is stored to Var_D1 if \(0<\) Var_D0.

Example Var_D0 \(=700 \rightarrow\) Var_D1 \(=800\)
(2) In the following program, the zone control applying negative and positive bias values of -10000 to 10000 is performed on the data set for Var_D0 when X1 turns ON, and the result is stored to Var_D10.
[Structured ladder/FBD]

[ST]
DZONEP(X1,-10000,10000,Var_D0,Var_D10);
[Operation]
- The value Var_D1 + \((-10000)\) is stored to Var_D10 if Var_D0 \(<0\).

Example Var_D0 \(=-12345 \rightarrow\) Var_D10 \(=-22345\)
- The value 0 is stored to Var_D10 if Var_D0 \(=0\).
- The value Var_D0 + 10000 is stored to Var_D10 if \(0<\) Var_D0.

Example Var_D0 \(=50000 \rightarrow\) Var_D10 \(=60000\)

\subsection*{7.13.4 Scaling (coordinate by point data)}

- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported


\section*{SCL(P)}
(1) Processes the scaling on the conversion data (unit of 16-bit data) specified for \(\S_{2}\) ) with the input value specified for (51), and stores the result to the device specified for (d). The scaling conversion is processed in accordance with the conversion data for scaling stored in ©2 and the following devices.

(2) If the result is not an integer value, the fractional part is rounded up.
(3) Set the \(X\) coordinate data of the conversion data for scaling in the ascending order.
(4) Set (51) within the range of the conversion data for scaling (device value of (s2).
(5) If the same X coordinate is specified by the multiple points, the value of Y coordinate with the highest point number is output.
Set the number of coordinate points of the conversion data for scaling within 1 to 32767 .

\section*{DSCL(P)}
(1) Processes the scaling on the conversion data (unit of 32-bit data) specified for \(\Omega_{2}\) with the input value specified for (51), and stores the result to the device specified for (a). The scaling conversion is processed in accordance with the conversion data for scaling stored in \(\Omega_{2}\) and the following devices.

(2) If the result is not an integer value, the fractional part is rounded up.
(3) Set the \(X\) coordinate data of the conversion data for scaling in the ascending order.
(4) Set (51) within the range of the conversion data for scaling (device value of (s2), (32) +1 ).
(5) If the same \(X\) coordinate is specified by the multiple points, the value of \(Y\) coordinate with the highest point number is output.
Set the number of coordinate points of the conversion data for scaling within 1 to 32767 .
1. The search method differs by the ON/OFF status of SM750.
\begin{tabular}{c|c|c} 
SM750 & Search method & Range for number of searches \\
\hline OFF & Linear search & \(1 \leqq\) number of comparisons \(\leqq 32767\) \\
\hline ON & Binary search & \(1 \leqq\) number of comparisons \(\leqq 15\) \\
\hline
\end{tabular}
2. When the conversion data for scaling are sorted in the ascending order, the search method differs by the SM750 status, and thus the processing speed also differs.
The processing speed is fixed according to the number of comparisons, and the less comparison results faster processing speed.
(a) Faster processing speed for the linear search

With the maximum number of coordinate points, when ©1) is set between the coordinate points 1 and 15 , the number of comparisons for linear search becomes less than/equal to 15 , and thus the processing speed of the linear search becomes faster.
(b) Faster processing speed for the binary search

Since the maximum number of comparisons is 15 , when \(\leftrightarrows 1\) is set to the coordinate point higher than 16 , the number of comparisons for binary search becomes less than/equal to the number of comparisons for linear search, and thus the processing speed of the binary search becomes faster.


\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The X coordinate data at the point in front of the conversion data for scaling ©(s) are not set in the ascending order. (However, when SM750 is ON, this error is not detected.)
(Error code: 4100)
- The input value specified for \((₫ 1)\) is outside the range of set conversion data for scaling.
(Error code: 4100)
- The coordinate points from the device \(\Omega_{\Omega}\) is outside the range of 1 to 32767 .
(Error code: 4100)
- The coordinate points from the device \(\circledR_{2}\) is outside of the specified device range.
(Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, the scaling is processed on the conversion data set to D100 and the following devices with the input value specified for DO when MO turns ON, and the result is output to D20.
[Structured ladder/FBD]

[ST]
SCL(M0,D0,D100,D20);
[Operation]
Configuration of conversion data for scaling


\subsection*{7.13.5 Scaling (coordinate by \(X / Y\) data)}

SCL2, DSCL2

- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported


\section*{SCL2(P)}
(1) Processes the scaling on the conversion data (unit of 16-bit data) specified for \(\S_{2}\) ) with the input value specified for (51), and stores the result to the device specified for (d). The scaling conversion is processed in accordance with the conversion data for scaling stored in ©2 and the following devices.

Configuration of conversion data for scaling
\begin{tabular}{|c|c|c|}
\hline \multicolumn{2}{|r|}{Setting item} & Device assignment \\
\hline \multicolumn{2}{|l|}{Coordinate points} & (S2) \\
\hline \multirow{4}{*}{X coordinate} & Point 1 & (52) +1 \\
\hline & Point 2 & (S2) + \\
\hline & ! & ! \\
\hline & Point n & (S2) \({ }^{\text {a }}\) \\
\hline \multirow{4}{*}{Y coordinate} & Point 1 & (S2) \({ }_{\text {n+1 }}\) \\
\hline & Point 2 & (S2) \({ }^{+1+2}\) \\
\hline & & ! \\
\hline & Point n & (32) 22 n \\
\hline
\end{tabular}

(2) If the result is not an integer value, the fractional part is rounded up.
(3) Set the \(X\) coordinate data of the conversion data for scaling in the ascending order.
(4) Set (51) within the range of the conversion data for scaling (device value of (s2).
(5) If the same \(X\) coordinate is specified by the multiple points, the value of \(Y\) coordinate with the highest point number is output.

\section*{DSCL2(P)}
(1) Processes the scaling on the conversion data (unit of 32-bit data) specified for \(\Omega_{2}\) ) with the input value specified for (51), and stores the result to the device specified for (d). The scaling conversion is processed in accordance with the conversion data for scaling stored in \({ }^{2} 2\) and the following devices.

(2) If the result is not an integer value, the fractional part is rounded up.
(3) Set the \(X\) coordinate data of the conversion data for scaling in the ascending order.
(4) Set (s1) within the range of the conversion data for scaling (device value of s2, (s2) +1).
(5) If the same \(X\) coordinate is specified by the multiple points, the value of \(Y\) coordinate with the highest point number is output.
(6) Set the number of coordinate points of the conversion data for scaling within 1 to 32767 .

When the conversion data for scaling are sorted in the ascending order, the search method differs by the SM750 status, and thus the processing speed also differs.
For details, refer to Section 7.13.4.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The \(X\) coordinate data are not set in the ascending order.
(Error code: 4100)
- The input value specified for \((51)\) is outside the range of set conversion data for scaling.
(Error code: 4100)
- The coordinate points specified for ®2 \(^{2}\) is outside the range of 1 to 32767 .
(Error code: 4100)
- The coordinate points from the device (s2) exceeds the specified device range.
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the scaling is processed on the conversion data set to D110 and the following devices with the input value specified for D0 when M0 turns ON, and the result is output to D200.
[Structured ladder/FBD]

[ST]
DSCL2(M0,D0,D110,D200);
[Operation]
\begin{tabular}{|c|c|c|c|}
\hline \multicolumn{2}{|l|}{Setting item} & Device & Setting content \\
\hline \multicolumn{2}{|l|}{Coordinate points} & D110 & K5 \\
\hline \multirow{5}{*}{\[
\underset{\text { coordinate }}{\mathrm{X}}
\]} & Point 1 & D111 & K7 \\
\hline & Point 2 & D112 & K13 \\
\hline & Point 3 & D113 & K15 \\
\hline & Point 4 & D114 & K18 \\
\hline & Point 5 & D115 & K20 \\
\hline \multirow{5}{*}{\[
\begin{gathered}
\mathrm{Y} \\
\text { coordinate }
\end{gathered}
\]} & Point 1 & D116 & K-14 \\
\hline & Point 2 & D117 & K-7 \\
\hline & Point 3 & D118 & K-15 \\
\hline & Point 4 & D119 & K-11 \\
\hline & Point 5 & D120 & K-18 \\
\hline
\end{tabular}


\subsection*{7.14 File Register Switching Instructions}

\subsection*{7.14.1 Switching file register block numbers}

- Q00JCPU: Not supported
- Universal model QCPU: Not supported for Q00UJCPU


\section*{Function}
(1) Changes the file register block number used in the program to the block number stored in the device specified for (s).
Following the block number change, all file registers used in the sequence program are processed to the file register of the block number after the change.

\section*{Example}

When switching block number from block No. 0 to block No. 1


\section*{区POINT}

Be cautious when a file register ( R ) is refreshed and the block No. of the file register is switched with the RSET instruction.
For the restrictions on file registers, refer to Section 3.5.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The block number specified for ©s does not exist.
(Error code: 4100)
- There is no file register for the specified block number.

\section*{\(\square\) Program Example}

In the following program, R0 of the block number 0 and the block number 1 are compared.
[Structured ladder/FBD]

[ST]
RSETP(SM400,0);
MOVP(SM400,R0,Var_D0);
RSETP(SM400,1);
MOVP(SM400,R0,Var_D1);
OUT(Var_D0=Var_D1,Y40);
OUT(Var_D0<Var_D1,Y41);
OUT(Var_D0>Var_D1,Y42);
[Operation]


\subsection*{7.14.2 Setting file register files}

QDRSET

- Universal model QCPU: Not supported for Q00UJCPU


\section*{3 Function}
(1) Changes the file name of the file register used in the program to the file name being stored in the device specified for (s).
After the file names have been changed, all the file registers being used by the sequence program process the file register of the block number 0 of the renamed file. Block number switch is performed by the RSET(P) instruction.

\section*{Example}

When switching from Drive No. 1/File name B to Drive No. 3/File name A

(2) Drive number can be specified from 1 to 4.

The drive number cannot be specified as drive 0 (program memory/internal memory).
(3) It is not necessary to specify the extension (.QDR) with the file name.
(4) A file name setting can be deleted by specifying the NULL character \((00 \mathrm{H})\) for the file name.
(5) File names specified with this instruction will be given priority even if a drive number and file name have been specified in the parameters.
1. If the file name is changed by the QDRSET(P) instruction, the file name returns to the name specified by the parameter when the CPU module is switched from STOP to RUN. To maintain the file name even after the CPU mode is changed from STOP to RUN, execute the QDRSET(P) instruction with the SM402 special relay, which turns ON during one scan when the CPU enters from STOP to RUN mode.
2. Do not change the file name of the file register with the QDRSET(P) instruction when the file register is specified as the refresh device. For the restrictions on file registers, refer to Section 3.5.

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- File name does not exist in the drive number specified for © (
(Error code: 2410)

\section*{\(\square\) Program Example}

In the following program, R0 of the drive number 1 " ABC " and the file name of the drive number 1 "DEF.QDR" are compared.
[Structured ladder/FBD]

[ST]
QDRSETP(X0,"1:ABC");
MOVP(X0,R0,Var_D0);
QDRSETP(X0,"1:DEF");
MOVP(X0,R0,Var_D1);
OUT(Var_D0=Var_D1,Y40);
OUT(Var_D0<Var_D1,Y41);
OUT(Var_D0>Var_D1,Y42);
[Operation]


\subsection*{7.14.3 Setting files for comments}

QCDSET


\section*{Function}
(1) Changes the file name of the file register used in the program to the file name being stored in the device specified for (s).
After the file name change, comment data being used by the sequence program perform processing in relation to the comment data of the file name after the change.

(2) Drive number can be specified from 1 to 4.

The drive number cannot be specified as drive 0 (program memory/internal memory). Drives that can be specified are different according to the CPU module.
Check the drives that can be specified in the manual of the CPU module to be used.
(3) It is not necessary to specify the extension (.QCD) with the file name.
(4) A file name setting can be deleted by specifying the NULL character \((00 \mathrm{H})\) for the file name.
(5) File names specified with this instruction will be given priority even if a drive number and file name have been specified in the parameters.

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- File name does not exist in the drive number specified for (s).
(Error code: 2410)
(1) For Universal model QCPU and LCPU, this instruction is not executed while SM721 (File access in progress) is ON even when the execution command of this instruction is turned ON. Execute this instruction when SM721 is OFF.
(2) For High-speed Universal model QCPU and LCPU, when specifying drive 2 (SD memory card) as drive No., this instruction will not be executed while SM606 (SD memory card forced disable instruction) is on. If executed, no processing is performed.

\section*{\(\triangle\) Program Example}

In the following program, the object file of the comments are switched to the file names "ABC.QCD" in the drive number 1 when X0 turns ON, and to the file names "DEF.QCD" in the drive number 3 when X 1 turns ON .
[Structured ladder/FBD]

[ST]
QCDSETP(X0,"1:ABC");
QCDSETP(X1,"3:DEF");

\section*{®POINT}

If the file name is changed by the QCDSET \((P)\) instruction, the file name returns to the name specified by the parameter when the CPU module is switched from STOP to RUN.
To maintain the file name even after the CPU mode is changed from STOP to RUN, execute the QCDSET(P) instruction with the SM402 special relay, which turns ON during one scan when the CPU enters from STOP to RUN mode.

\subsection*{7.15 Clock Instructions}

\subsection*{7.15.1 Reading clock data}

DATERD

\section*{DATERD(P)}

> P: Executing condition

5


\[
\begin{aligned}
& \text { instructions. } \\
& \text { DATERD DATERDP }
\end{aligned}
\]

Input argument,
Output argument,

EN:
\(\mathrm{d}: \quad\) Clock data that are read, or start number or the array of the :Array of ANY16 (1..7) device that stores clock data
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & & vice & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J等:} & \multirow[b]{2}{*}{U...月Gá...} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{6}{|c|}{-} \\
\hline
\end{tabular}

\section*{\(\left\{\begin{array}{l}\text { Function }\end{array}\right.\)}
(1) Reads year, month, day, minute, second, and day of week from the clock element of the CPU module and stores them in (d) and the following devices in BIN value.
\begin{tabular}{|c|c|c|c|}
\hline \multirow[t]{3}{*}{} & (d) \([0]\) & Year & \multirow[t]{2}{*}{\[
\left\lvert\, \begin{aligned}
& (1980 \text { to } 2079) \\
& (1 \text { to 12) })
\end{aligned}\right.
\]} \\
\hline & (d) [1] & Month & \\
\hline & (d) [2] & Day & (1 to 31) \\
\hline Clock element \(\square\) & (d) [3] & Hour(24.hour clock) & (0 to 23) \\
\hline & (d) [4] & Minute & (0 to 59) \\
\hline & (d) [5] & Second & (0 to 59) \\
\hline & (d) [6] & Day of week & (0 to 6) \\
\hline
\end{tabular}
(2) The year in © [0] is stored as 4-digit year indication.
(3) The day of week in © [6] is stored as 0 to 6 to represent the days from Sunday to Saturday.
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline Day of week & Sun & Mon & Tue & Wed & Thu & Fri & Sat \\
\hline Stored data & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
\end{tabular}
(4) Compensation is made automatically for leap years.

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device specified for (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)

\section*{\(\triangle\) Program Example}

In the following program, the following clock data are output in \(B C D\) values.
Year ..................... Y70 to Y7F
Month .............. Y68 to Y6F
Day ..................... Y60 to Y67
Hour................. Y58 to Y5F
Minute.............. Y50 to Y57
Second .............. Y48 to Y4F
Day of week....... Y44 to Y47
[Structured ladder/FBD]


\section*{[ST]}

DATERD(SM400,Var_DO);
BCDP(SM400,Var_DO[0],K4Y70);
BCDP(SM400,Var_DO[1],K2Y68);
BCDP(SM400,Var_D0[2],K2Y60);
BCDP(SM400,Var_D0[3],K2Y58);
BCDP(SM400,Var_D0[4],K2Y50);
BCDP(SM400,Var_D0[5],K2Y48);
BCDP(SM400,Var_D0[6],K1Y44);
[Operation]


\subsection*{7.15.2 Writing clock data}

DATEWR

DATEWR(P)
P: Executing condition
indicates any of the following

\(\begin{array}{lll}\text { Input argument, } & \text { EN: } & \text { Executing condition } \\ & \mathrm{s}: & \text { Clock data to be written to clock element } \\ \text { Output argument, } & \text { ENO: } & \text { Execution result }\end{array}\)
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & Inter & evice & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{} & \multirow[b]{2}{*}{} & \multirow[b]{2}{*}{Zn} & \multirow[b]{2}{*}{Constant} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{6}{|c|}{-} \\
\hline
\end{tabular}

\section*{IV Function}
(1) Writes clock data stored in ©s and the following devices to the clock element of the CPU module.

(2) Each item is set in BIN value.
(3) The year for © \({ }^{(3)}[0]\) is specified in 4-digit values between 1980 and 2079. (If a particular era name is used for year settings, leap years cannot be automatically adjusted.)
(4) (5) [1] specifies the month in values from 1 to 12 (January to December).
(5) © [2] specifies the day in values from 1 to 31.
(6) © [3] specifies the hour in values from 0 to 23 (using 24-hour clock, from 0 hours to 23 hundred hours). (Uses the 24-hour clock.)
(7) © [4] specifies the minute in values from 0 to 59.
(8) © [5] specifies the second in values from 0 to 59 .
(9) (s) [6] specifies the day of week in values from 0 to 6 (Sunday to Saturday).
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline Day of week & Sun & Mon & Tue & Wed & Thu & Fri & Sat \\
\hline Stored data & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
\end{tabular}

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- Each item of data have been set outside the setting range.
(Error code: 4100)
- The device specified for (s) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the following clock data entered in BCD value are written to the clock element when X40 turns ON.
\begin{tabular}{ll} 
Year ............... X30 to X3F & Hour.................. X18 to X1F \\
Month............ X28 to X2F & Minute............. X10 to X17 \\
Day ............. X20 to X27 & Second .......... X8 to XF
\end{tabular}

Day of week... X 4 to X 7
[Structured ladder/FBD]


\section*{[ST]}

BIN(X40,K4X30,Var_D0[0]);
BIN(X40,K2X28,Var_D0[1]);
BIN(X40,K2X20,Var_D0[2]);
BIN(X40,K2X18,Var_D0[3]);
BIN(X40,K2X10,Var_D0[4]);
BIN(X40,K2X8,Var_D0[5]);
BIN(X40,K1X4,Var_D0[6]);
DATEWRP(X40,Var_D0);
[Operation]


\subsection*{7.15.3 Clock data addition}

DATE+

\section*{DATE \(+(\mathrm{P})\)}
indicates any of the following instructions.
DATE \(+\quad\) DATE +P


Input argument,
\begin{tabular}{lll} 
& s1: & \begin{tabular}{l} 
Start number or the array of the device that stores clock (time) : Array of ANY16 (1..3) \\
data to be added
\end{tabular} \\
Ontput argument, & ENO: \begin{tabular}{l} 
Start number or the array of the device that stores added time :Array of ANY16 (1..3) \\
data
\end{tabular} & \begin{tabular}{l} 
Execution result
\end{tabular} \\
d: & \begin{tabular}{l} 
Start number or the array of the device that stores clock (time) :Array of ANY16 (1..3) \\
data of the addition result
\end{tabular}
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{} & \multirow[b]{2}{*}{U...ig\%} & \multirow[t]{2}{*}{Zn} & \multirow[b]{2}{*}{Constant} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{6}{|c|}{-} \\
\hline (52) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{6}{|c|}{-} \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{6}{|c|}{-} \\
\hline
\end{tabular}

Function
(1) Adds the time data specified for (s2) to the clock data specified for (s1), and stores the result to (d) and the following devices.
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline & & \multicolumn{2}{|l|}{Data range} & \multicolumn{3}{|c|}{Data range} & & \multicolumn{2}{|r|}{Data range} \\
\hline (51) [0] & Hour & (0 to 23) & (32) [0] & Hour & (0 to 23) & & (d) [0] & Hour & (0 to 23) \\
\hline (51) [1] & Minute & (0 to 59) + & (s2) [1] & Minute & (0 to 59) & - & (d) [1] & Minute & (0 to 59) \\
\hline (51) [2] & Second & (0 to 59) & (32) [2] & Second & (0 to 59) & & (d) [2] & Second & (0 to 59) \\
\hline
\end{tabular}

For example, adding the time 7:48:10 to 6:32:40 would result in the following operation.

\begin{tabular}{l|l|} 
& S2) \([0]\) \\
& Hour: 7 \\
& S2) \([1]\) \\
S2 \([2]\) & Minute: 48 \\
\cline { 2 - 3 } & Second: 10 \\
\hline
\end{tabular}
\(\square\)
\begin{tabular}{l|l|} 
(d) \([0]\) & Hour: 14 \\
& (d) \([1]\) \\
(d) \([2]\) & Minute: 20 \\
\cline { 2 - 3 } & Second: 50 \\
\cline { 2 - 3 } &
\end{tabular}
(52) [2] Second: 10
(2) If the addition result of time exceeds 24 hours, 24 hours will be subtracted from the sum to make the final operation result.
For example, if the time 20:20:20 were added to 14:20:30, the result would not be 34:40:50, but would instead be 10:40:50.
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline (S1) [0] & Hour: 14 & \multirow{3}{*}{+} & \multirow[t]{2}{*}{\begin{tabular}{l}
(s) [0] \\
(82) [1]
\end{tabular}} & Hour: 20 & & (d) \([0]\) & Hour: 10 \\
\hline (51) [1] & Minute: 20 & & & Minute: 20 & \multirow[t]{2}{*}{\(\square\)} & (d) [1] & Minute: 40 \\
\hline (51) [2] & Second: 30 & & (52) [2] & Second: 20 & & (d) [2] & Second: 50 \\
\hline
\end{tabular}

\section*{Remark}

For further information regarding the data that can be set for hours, minutes, and seconds, refer to Section 7.15.2.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value set for (s1) and (s2) is outside of the setting range.
(Error code: 4100)
- The device specified for (①), (22) or (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{Program Example}

In the following program, 1 hour is added to the clock data read from the clock element when X20 turns ON, and the result is stored to Var_D100 and the following devices.
[Structured ladder/FBD]


\section*{[Operation]}
- Clock data read by the DATERDP instruction.

- Addition by the DATE+P instruction.


\subsection*{7.15.4 Clock data subtraction}

DATE-(P)
P: Executing condition

\begin{tabular}{|c|c|c|c|}
\hline \multirow[t]{3}{*}{Input argument,} & EN: & Executing condition & :Bit \\
\hline & s1: & Start number or the array of the device that stores clock (time) data to be subtracted & :Array of ANY16 (1..3) \\
\hline & s2: & Start number or the array of the device that stores subtracted time data & :Array of ANY16 (1..3) \\
\hline \multirow[t]{2}{*}{Output argument,} & ENO: & Execution result & :Bit \\
\hline & d: & Start number or the array of the device that stores clock (time) data of the subtraction result & :Array of ANY16 (1..3) \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{小, \%en} & \multirow[b]{2}{*}{U"fgal} & \multirow[b]{2}{*}{Zn} & \multirow[b]{2}{*}{Constant} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & - & & & \multicolumn{6}{|c|}{-} \\
\hline (2) & - & & & \multicolumn{6}{|c|}{-} \\
\hline (d) & - & & & \multicolumn{6}{|c|}{-} \\
\hline
\end{tabular}

\section*{Function}
(1) Subtracts the time data specified for (2) from the clock data specified for (51), and stores the result to (d) and the following devices.


For example, if the clock time 3:50:10 were subtracted from the clock time 10:40:20, the operation would be performed as follows.



(2) If the operation results in a negative number, 24 will be added to the result to make a final operation result.
For example, if the clock time 10:42:12 were subtracted from 4:50:32, the result would not be \(-6: 8: 20\), but rather would be 18:8:20.


\section*{Remark}

For further information regarding the data that can be set for hours, minutes, and seconds, refer to Section 7.15.2.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The value set for \((51)\) and \((52)\) is outside of the setting range.
(Error code: 4100)
- The device specified for (51), (32) or (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the time data stored in Var_D10 and the following devices are subtracted from the clock data read from the clock element when X1C turns ON, and the result is stored to Var_R10 and the following devices.
[Structured ladder/FBD]


\section*{[Operation]}
- Clock data read by the DATERDP instruction.
\begin{tabular}{|c|c|c|c|}
\hline Clock element & Var_D100[0] & 1995 & Year \\
\hline & Var_D100[1] & 4 & Month \\
\hline & Var_D100[2] & 20 & Day \\
\hline & Var_D100[3] & 3 & Hour \\
\hline & Var_D100[4] & 21 & Minute \(\}\) Time data \\
\hline & Var_D100[5] & 20 & Second \\
\hline & Var_D100[6] & 1 & Day of week \\
\hline
\end{tabular}
- Subtraction by the DATE- P instruction (when 10 hours, 40 minutes, and 10 seconds have been specified for Var_D10).


\subsection*{7.15.5 Time data conversion (hour/minute/second format to seconds)}

SECOND

\section*{SECOND(P)}

P: Executing condition


\begin{tabular}{llll} 
Input argument, & EN: & Executing condition & :Bit \\
s: & \begin{tabular}{l} 
Start number or the array of the device that stores clock data \\
before the conversion
\end{tabular} & \(:\) Array of ANY16 (1..3)
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{Ji.al:} & \multirow[b]{2}{*}{U.alalial} & \multirow{2}{*}{Zn} & \multirow{2}{*}{Constant} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \multicolumn{2}{|c|}{-} \\
\hline (d) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & \multicolumn{2}{|c|}{-} \\
\hline
\end{tabular}

\section*{\(\hat{3}\) Function}

Converts the time data stored in (s) and the following devices to the data in seconds and stores the conversion result to the device specified for (d).


For example, if the value were 4 hours, 29 minutes and 31 seconds, the conversion operation would be performed as follows.


\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value set for (s) is outside of the setting range.
(Error code: 4100)
- The device specified for (s) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the clock data read from the clock element are converted to the data in seconds when X20 turns ON, and the result is stored to Var_D100.
[Structured ladder/FBD]

[ST]
DATERDP(X20,Var_D10);
MOVP(X20,Var_D10[3],Var_D13[0]);
MOVP(X20,Var_D10[4],Var_D13[1]);
MOVP(X20,Var_D10[5],Var_D13[2]);
SECONDP(X20,Var_D13,Var_D100);

\section*{[Operation]}
- Clock data read by the DATERDP instruction.

- Conversion to seconds by the SECONDP instruction.
\begin{tabular}{|c|c|c|}
\hline Var_D13[0] & 20 & \\
\hline Var_D13[1] & 21 & \(\square\) Var_D100 78238 \\
\hline Var_D13[2] & 23 & \\
\hline
\end{tabular}

\subsection*{7.15.6 Time data conversion (seconds to hour/minute/second format)}

HOUR

\section*{Basic \\ High
performanc \\ Process \\ Redundant \\ Universal \\ LCPU}

HOUR(P)

> P: Executing condition
\begin{tabular}{|ll|}
\hline HOUR & \\
instructions. & indicates any of the following \\
\\
\\
\\
\\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline \multirow[t]{2}{*}{Input argument,} & EN: & Executing condition & :Bit \\
\hline & s : & Start number of the device that stores clock data before the conversion & :ANY32 \\
\hline \multirow[t]{2}{*}{Output argument,} & ENO: & Execution result & :Bit \\
\hline & d: & Start number or the array of the device that stores clock data after the conversion & :Array of ANY16 (1..3) \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J:..alin} & \multirow[b]{2}{*}{} & \multirow{2}{*}{Zn} & \multirow{2}{*}{Constant} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & - & - \\
\hline
\end{tabular}

\section*{Function}

Converts the data in seconds stored in the device specified for (s) to an hour/minute/second format, and stores the conversion result to (d) and the following devices.


For example, if 45325 seconds were the value specified, the conversion operation would be performed as follows.
\[
7
\]

\begin{tabular}{l|r|} 
& \\
(d) 0\(]\) & 12 \\
\hline (d) 1\(]\) & 35 \\
\hline (d) 2\(]\) & 25 \\
\cline { 2 - 3 } & \\
\hline
\end{tabular}

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The value set for (s) is outside of the setting range.
(Error code: 4100)
- The device specified for (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)

\section*{\(\square\) Program Example}

In the following program, the value in seconds stored in Var_D0 is converted to Hour/Minute/ Second format when X20 turns ON, and the result is stored to Var_D100 and the following devices.
[Structured ladder/FBD]


\section*{[ST]}

HOURP(X20,Var_D0,Var_D100);

\section*{[Operation]}
- Conversion to Hour/Minute/Second format by the HOURP instruction (when the value 40000 seconds has been specified for Var_D0).


\subsection*{7.15.7 Date data comparison}

- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported
\begin{tabular}{|c|c|c|c|c|}
\hline LD & (DT=) & - \(\square\) DT \(=\) & = & ) \\
\hline AND & ( DT <>) & \(\square \mathrm{DT}\) <> & \# & \\
\hline OR & (DT<=) & \(\square \mathrm{D}\) ¢ \(<=\) & \(\leqq\) & \\
\hline & (DT<) & \(\square \mathrm{DT}\) < & < & \\
\hline & (DT>=) & -DT>= & \(\geqq\) & \\
\hline & (DT>) & -DT> & > & ) \\
\hline
\end{tabular}

\begin{tabular}{lll} 
Input argument, & EN: & Executing condition \\
& s1: & Start number of the device that stores the data to be compared:ANY16 \\
& s2: & Start number of the device that stores the data to be compared:ANY16
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J等:} & \multirow[b]{2}{*}{U} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s1) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & - & - \\
\hline (52) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & - & - \\
\hline n & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline
\end{tabular}
(1) Compares the date data specified for (51) and (s2), or compares the date data specified for (51) and the current date data. The comparison target can be selected by specifying n .
(a) Comparing with the specified date data
- Compares the date data specified for \((51)\) and the date data specified for \(\varsigma_{2}\) as a normally open contact according to the condition of \(n\).

(b) Comparing with the current date data
- Compares the date data specified for (s1) and the current date data as a normally open contact according to the condition of \(n\).


\section*{®POINT}

When comparing with the specified date data or the current date data, if any of the following conditions apply to the setting of \((11)\) and \(\Im_{2}\), an operation error (error code: 4101) or malfunction may occur.
- The index setting is specified exceeding the device range.
- A file register is specified without setting file registers.
(2) Each item is set in BIN value.
(3) The year for (s1) and ©22 is specified in 4-digit values between 1980 and 2079.
(4) The month for (51) +1 and \(\left.\S_{2}\right)+1\) is specified in values between 1 and 12 (January to December).
(5) The day for (51) +2 and (32) +2 is specified in values between 1 and 31 .
(6) The comparison target can be specified in detail by specifying following values as shown below. The following shows the bit configuration of \(n\).

0 is specified for b3 to b14. If a value other than 0 is set,
they become non-conduction state regardless of the operation result.

(a) Comparison target date ( b 0 to b 2 )
- 0: No comparison is performed with the comparison target date data (year/month/ day).
- 1: Compares with the comparison target date data (year/month/day). specified for b0 to b2.
(b) Comparison target (b15)
- 0: Compares the date data specified for (s1) and the date data specified for s2).
- 1: Compares the date data specified for ©s1) and the current date data. The date data specified for (®2) are ignored.
(c) The following table shows the bit processing of the comparison target.
\begin{tabular}{|c|c|c|c|}
\hline The value of \(n\) for the comparison with the specified date data & The value of n for the comparison with the current date data & Comparison target date & Processing \\
\hline 0001H & 8001H & Day & Compares the day (\$1 +2 ) only. \\
\hline 0002H & 8002H & Month & Compares the month (S1) +1 ) only. \\
\hline 0003H & 8003H & Month, Day & Compares the month ( \((11)+1)\) and the day \((\) (s1) +2 ). \\
\hline 0004H & 8004H & Year & Compares the year (51) ) only. \\
\hline 0005H & 8005H & Year, Day & Compares the year (S1) and the day (\$1) +2 ). \\
\hline 0006H & 8006H & Year, Month & Compares the year (S1) and the month ( \((11)+1\) ) \\
\hline 0007H & 8007H & Year, Month, Day & Compares the year ((s1)), the month (s1) +1 ), and the day
\[
\text { ( }(11)+2) .
\] \\
\hline Other than 0001 H to 0007 H , & 001H to 8007H & None & No comparison operation on the year (①) ), the month (s1) +1 ), and the day ( \((1)+2)\). (non-conduction state) \\
\hline
\end{tabular}
(7) If the data stored in the comparison target device are not recognized as date data, SM709 turns ON after the instruction execution and the device becomes a non-conduction state. Even when the date are not recognized as date data, if they are within the setting rage, SM709 does not turn ON.

In addition, when the value of \((11)\) to \((51)+2\) or \(\left(s_{2}\right.\) to \(\S_{2}+2\) exceeds the specified device range, SM709 turns ON and it becomes a non-conduction state.
Note that, once SM709 turns ON, it stays ON until the reset or power OFF of CPU. Turn it OFF as required.
(8) The following table shows the comparison result of each instruction.
\begin{tabular}{c|c|c|c|c|c|}
\hline \begin{tabular}{c} 
Instruction \\
symbol in \\
\hline
\end{tabular} & Condition & Comparison result & \begin{tabular}{c} 
Instruction \\
symbol in
\end{tabular} & Condition & Comparison result
\end{tabular}
(a) The following shows the comparison example of date.


The following table shows the comparison result of date \(A, B\), and \(C\) indicated above. Even when the data are compared under the same condition, the comparison result
changes according to the specified comparison target.
\begin{tabular}{c|c|c|c}
\multirow{2}{*}{\begin{tabular}{c} 
Comparison \\
target
\end{tabular}} & y \\
\cline { 2 - 4 } Day & \(\bigcirc\) & \(\mathrm{B}<\mathrm{C}\) & \(\mathrm{A}<\mathrm{C}\) \\
\hline Month & \(\times\) & \(\times\) & \(\times\) \\
\hline Year, Day & \(\times\) & \(\bigcirc\) & \(\times\) \\
\hline Year & \(\bigcirc\) & \(\bigcirc\) & \(\times\) \\
\hline Year, Day & \(\bigcirc\) & \(\bigcirc\) & \(\bigcirc\) \\
\hline Year, Month & \(\bigcirc\) & \(\bigcirc\) & \(\bigcirc\) \\
\hline Year, Month, Day & \(\bigcirc\) & \(\bigcirc\) & \(\bigcirc\) \\
\hline None & \(\times\) & \(\bigcirc\) & \(\bigcirc\) \\
\hline
\end{tabular}
\(\bigcirc\) : Conduction state \(\times\) : Non-conduction state
(b) With the date that do not actually exist, if they are settable date, they are compared according to the following conditions.
- Date A: 2006/02/30 (The date does not actually exist, but it can be set.)
- Date B: 2007/03/29
- Date C: 2008/02/31 (The date does not actually exist, but it can be set.)
\begin{tabular}{c|c|c|c}
\multirow{2}{*}{\begin{tabular}{c} 
Comparison \\
target
\end{tabular}} & \(\mathrm{A}<\mathrm{B}\) & \(\mathrm{B}<\mathrm{C}\) & \(\mathrm{A}<\mathrm{C}\) \\
\cline { 2 - 4 } Day & \(\times\) & \(\bigcirc\) & \(\bigcirc\) \\
\hline Month & \(\bigcirc\) & \(\times\) & \(\times\) \\
\hline Year, Day & \(\bigcirc\) & \(\times\) & \(\bigcirc\) \\
\hline Year & \(\bigcirc\) & \(\bigcirc\) & \(\bigcirc\) \\
\hline Year, Day & \(\bigcirc\) & \(\bigcirc\) & \(\bigcirc\) \\
\hline Year, Month & \(\bigcirc\) & \(\bigcirc\) & \(\bigcirc\) \\
\hline Year, Month, Day & \(\bigcirc\) & \(\bigcirc\) & \(\bigcirc\) \\
\hline None & \(\times\) & \(\times\) & \(\times\) \\
\hline
\end{tabular}
(9) The ORDT=, ORDT<>, ORDT<=, ORDT<, ORDT>=, or ORDT> instruction performs comparison operation between the operation result of "(s1), (s2)" and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400.
When using the ORDT=, ORDT<>, ORDT<=, ORDT<, ORDT>=, or ORDT> instruction, connect EN and ENO in series as shown below.


\section*{Operation Error}

No operation error occurs in the execution of the DT=, DT<>, DT<=, DT<, DT>=, and DT> instructions.

\section*{\(\square\) Program Example}
(1) In the following program, the data in D0 and the data in D10 (year/month/day) are compared, and Y33 turns ON if they are matched.
[Structured ladder/FBD]

(2) In the following program, the data in DO and the current date data (year/month) are compared when M0 turns ON, and Y33 turns ON if they are not matched.
[Structured ladder/FBD]

(3) In the following program, the data in D0 and the data in D10 (year/day) are compared when M0 turns ON, and Y33 turns ON if the value of data in D10 is less than the value of data in DO.
[Structured ladder/FBD]

(4) In the following program, the data in D0 and the current date data (year) are compared, and Y33 turns ON if the value of current date data is equal to/higher than the value of data in D0. [Structured ladder/FBD]


\subsection*{7.15.8 Time data comparison}
- QnU(D)(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "10102" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and QnUDVCPU: Supported
\begin{tabular}{|c|c|c|c|c|}
\hline LD & (TM=) & - वTM= & = & ) \\
\hline AND & (TM<>) & ロTM<> & \# & \\
\hline OR & ( \(\mathrm{TM}<=\) ) & \(\square\) प \(M<=\) & \(\leqq\) & \\
\hline & (TM<) & \(\square\) पM< & : < & \\
\hline & ( \(\mathrm{TM}>=\) ) & \(\square T M>=\) & \(: \geqq\) & \\
\hline & ( \(\mathrm{TM}>\) ) & - वTM> & : > & ) \\
\hline
\end{tabular}


Input argument,

Output argument,
\begin{tabular}{llc} 
EN: & Executing condition \\
s1: & Start number of the device that stores the data to be compared:ANY16 \\
s2: & Start number of the device that stores the data to be compared:ANY16 \\
ENO: & Execution result \\
\(\mathrm{n}:\) & \begin{tabular}{l} 
Value to indicate the comparison target, or number of data to
\end{tabular} & \(:\) ANY16 \\
& which the comparison target is stored
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J\%.....if} & \multirow[b]{2}{*}{U...ig:...} & \multirow[b]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline (2) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline n & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline
\end{tabular}
(1) Compares the time data specified for (51) and ©2 , or compares the time data specified for (51) and the current time data. The comparison target can be selected by specifying n .
(a) Comparing with the specified time data
- Compares the time data specified for (s1) and the time data specified for (s2) as a normally open contact according to the condition of \(n\).

(b) Comparing with the current time data
- Compares the time data specified for (51) and the current time data as a normally open contact according to the condition of \(n\).
- The time data specified for \(\S_{\Omega}\) are treated as dummy data and ignored.


\section*{区POINT}

When comparing with the specified time data or the current time data, if any of the following conditions apply to the setting of (s1) and (s2), an operation error (error code: 4101) or malfunction may occur.
- The index setting is specified exceeding the device range.
- A file register is specified without setting file registers.
(2) Each item is set in BIN value.
(3) The hour for © \(\mathrm{S}_{1}\) and (s2) is specified in values between 0 and 23. (In the 24-hour clock)
(4) The minute for © \(\mathrm{Sl}_{1}+1\) and \(\mathrm{S}_{2}+1\) is specified in values between 0 and 59 .
(5) The second for (s1) +2 and (®2) +2 is specified in values between 0 and 59 .
(6) The comparison target can be specified in detail by specifying following values as shown below. The following shows the bit configuration of \(n\).

0 is specified for b3 to b14. If a value other than 0 is set,
they become non-conduction state regardless of
the operation result.

(a) Comparison target time (b0 to b2)
- 0: No comparison is performed with the comparison target time data (hour/minute/ second).
- 1: Compares with the comparison target time data (hour/minute/second).
(b) Comparison target (b15)
- 0: Compares the time data specified for (s1) and the time data specified for (s2).
- 1: Compares the time data specified for (51) and the current time data. The time data specified for \({ }^{(2)}\) are ignored.
(c) The following table shows the bit processing of the comparison target.
\begin{tabular}{|c|c|c|c|}
\hline The value of n for the comparison with the specified time data & The value of \(n\) for the comparison with the current time data & Comparison target time & Processing \\
\hline 0001H & 8001H & Second & Compares the second (s1) +2) only. \\
\hline 0002H & 8002H & Minute & Compares the minute \((51+1)\) only . \\
\hline 0003H & 8003H & Minute, Second & Compares the minute ( \(51+1\) ) and the second ( \((51)+2\) ). \\
\hline 0004H & 8004H & Hour & Compares the hour (S1) ) only. \\
\hline 0005H & 8005H & Hour, Second & Compares the hour (\$1) ) and the second (\$1)+2). \\
\hline 0006H & 8006H & Hour, Minute & Compares the hour (\$1) and the minute ( 51 +1). \\
\hline 0007H & 8007H & Hour, Minute, Second & Compares the hour (S1) ), the minute (S1) +1 ), and the second ( \((11)+2)\). \\
\hline \multicolumn{2}{|l|}{Other than
\[
0001 \mathrm{H} \text { to } 0007 \mathrm{H}, 8001 \mathrm{H} \text { to } 8007 \mathrm{H}
\]} & None & No comparison operation on the hour (s1) ), the minute (S1) +1 ), and the second \((\) (s1 +2\()\). (non-conduction state) \\
\hline
\end{tabular}
(7) If the data stored in the comparison target device are not recognized as time data, SM709 turns ON after the instruction execution and the device becomes a non-conduction state.

In addition, when the value of (51) to (51) +2 or (s2) to (s2) +2 exceeds the specified device range, SM709 turns ON and it becomes a non-conduction state.
Note that, once SM709 turns ON, it stays ON until the reset or power OFF of CPU. Turn it OFF as required.
(8) The following table shows the comparison result of each instruction.
\begin{tabular}{|c|c|c|c|c|c|}
\hline Instruction symbol in \(\square\) & Condition & Comparison result & Instruction symbol in \(\square\) & Condition & Comparison result \\
\hline \(\square \mathrm{TM}=\) & (51) \(=\) s2 & \multirow{6}{*}{Conduction state} & \(\square \mathrm{TM}=\) & (s1) \(\neq\) (s2) & \multirow{6}{*}{Non-conduction state} \\
\hline \(\square \mathrm{TM}<>\) & (s1) \(\neq\) ( 52 & & \(\square \mathrm{TM}<>\) & (51) \(=\) (s2) & \\
\hline \(\square \mathrm{TM}<=\) & (s1) \(\leqq\) (s2) & & \(\square \mathrm{TM}<=\) & (51) \(>\) (s2) & \\
\hline \(\square \mathrm{TM}<\) & (51) < (s2) & & \(\square\) TM< & (s1) \(\geqq\) ( 22 & \\
\hline \(\square \mathrm{TM}>=\) & (s1) \(\geqq\) (s2) & & \(\square \mathrm{TM}>=\) & (51) < (s2) & \\
\hline \(\square\) TM> & (51) \(>\) (s2) & & \(\square \mathrm{TM}>\) & (51) \(\leqq\) ( 22 & \\
\hline
\end{tabular}
(a) The following shows the comparison example of time.


The following table shows the comparison result of time \(\mathrm{A}, \mathrm{B}\), and C indicated above. Even when the data are compared under the same condition, the comparison result changes according to the specified comparison target.
\begin{tabular}{c|c|c|c}
\(*\) & \multicolumn{3}{|c}{ Comparison condition } \\
\cline { 2 - 4 } target & \(\mathrm{A}<\mathrm{B}\) & \(\mathrm{B}<\mathrm{C}\) & \(\mathrm{A}<\mathrm{C}\) \\
\hline Second & \(\bigcirc\) & \(\times\) & \(\times\) \\
\hline Minute & \(\times\) & \(\bigcirc\) & \(\times\) \\
\hline Minute, Second & \(\times\) & \(\bigcirc\) & \(\times\) \\
\hline Hour & \(\bigcirc\) & \(\bigcirc\) & \(\bigcirc\) \\
\hline Hour, Second & \(\bigcirc\) & \(\bigcirc\) & \(\bigcirc\) \\
\hline Hour, Minute & \(\bigcirc\) & \(\bigcirc\) & \(\bigcirc\) \\
\hline \begin{tabular}{c} 
Hour, Minute, \\
Second
\end{tabular} & \(\bigcirc\) & \(\times\) & \(\times\) \\
\hline None & \(\times\) & & \\
\hline
\end{tabular}
\[
\text { O: Conduction state } \times \text { : Non-conduction state }
\]
(9) The ORTM \(=\), ORTM<>, ORTM<=, ORTM<, ORTM>=, or ORTM \(>\) instruction performs comparison operation between the operation result of "(1), (®2)" and EN. Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400.
When using the ORTM \(=\), ORTM<>, ORTM<=, ORTM<, ORTM>=, or ORTM> instruction, connect EN and ENO in series as shown below.


\section*{O Operation Error}

No operation error occurs in the execution of the \(\mathrm{TM}=, \mathrm{TM}<>, \mathrm{TM}<=, \mathrm{TM}<, \mathrm{TM}>=\), and \(\mathrm{TM}>\) instructions.

\section*{\(\triangle\) Program Example}
(1) In the following program, the data in D0 and the data in D10 (hour/minute/second) are compared, and Y33 turns ON if they are matched.
[Structured ladder/FBD]

(2) In the following program, the data in D0 and the current time data (hour/minute) are compared, and Y33 turns ON if they are not matched.
[Structured ladder/FBD]

(3) In the following program, the data in D0 and the data in D10 (hour/second) are compared, and Y33 turns ON if the value of data in D10 is less than the value of data in D0.
[Structured ladder/FBD]

(4) In the following program, the data in D0 and the current date data (hour) are compared, and Y33 turns ON if the value of current time data is equal to/higher than the value of data in D0. [Structured ladder/FBD]


\subsection*{7.16 Extended Clock Instructions}

\subsection*{7.16.1 Read extended clock data}

\section*{S(P)_DATERD}

- High Performance model QCPU: Supported if first 5 digits of the serial number are "07032" or later
- Process CPU: Supported if first 5 digits of the serial number are " 07032 " or later
- Redundant CPU: Supported if first 5 digits of the serial number are " 07032 " or later

S(P)_DATERD



Input argument, EN:
Output argument,
d:

\section*{Executing condition}

Execution result
Start number of device that stores the read clock data

:Bit
:Bit
:ANY16 (0..7)


Function
(1) Reads year, month, day, minute, second, day of week, and millisecond from the clock element of the CPU module and stores them in (d) and the following devices in BIN value.

(2) The year in © \([0]\) is stored as 4-digit year indication.
(3) The day of week in © [6] is stored as 0 to 6 to represent the days from Sunday to Saturday.
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline Day of week & Sun & Mon & Tue & Wed & Thu & Fri & Sat \\
\hline Stored data & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
\end{tabular}
(4) Compensation is made automatically for leap years.

\section*{OOperation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device specified for (d) exceeds the corresponding device range.
(For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\triangle\) Program Example}

In the following program, the clock data indicated below are output in \(B C D\) values.
Year ..................... Y70 to Y7F
Month .............. Y68 to Y6F
Day ................... Y60 to Y67
Hour.................. Y58 to Y5F
Minute.............. Y50 to Y57
Second .............. Y48 to Y4F
Day of week....... Y44 to Y47
Millisecond.......... Y38 to Y43
[Structured ladder/FBD]

```

[ST]
SP_DATERD(SM400,DO);
BCDP(SM400,D0,K4Y70);
BCDP(SM400,D1,K2Y68);
BCDP(SM400,D2,K2Y60);
BCDP(SM400,D3,K2Y58);
BCDP(SM400,D4,K2Y50);
BCDP(SM400,D5,K2Y48);
BCDP(SM400,D6,K1Y44);
BCDP(SM400,D7,K3Y38);

```
[Operation]


\section*{Caution}
(1) Even when the incorrect clock data are set in the CPU module, this instruction reads the data and stores them to the devices. (Example: February 30)
When setting clock data using the DATEWR instruction or GX Works2, set the correct clock data.
(2) The difference of data when reading the millisecond clock data is 2 ms maximum. (The difference between the data stored at the clock element in the CPU and data read by this instruction.)
(3) Digit specification of bit devices can be used when the following conditions (a) and (b) are satisfied.
(a) Digit specification: K4
(b) Start device: multiple of 16

If the above conditions are not satisfied, an INSTRCT CODE ERR. (error code: 4004) occurs.

\subsection*{7.16.2 Addition of extended clock data}

- High Performance model QCPU: Supported if first 5 digits of the serial number are "07032" or later
- Process CPU: Supported if first 5 digits of the serial number are " 07032 " or later
- Redundant CPU: Supported if first 5 digits of the serial number are "07032" or later


\section*{Function}
(1) Adds the clock data specified for \((2)\) to the clock data specified for \((511\), and stores the result to (d) and the following devices.
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline \multirow[b]{2}{*}{(s1) [0]} & & \multicolumn{3}{|l|}{Setting data} & \multicolumn{3}{|l|}{Setting data} & \multirow[t]{2}{*}{Setting data
\[
(0 \sim 23)
\]} \\
\hline & Hour & (0~23) & (52) [0] & Hour & (0~23) & (d) \([0]\) & Hour & \\
\hline (51) [1] & Minute & \((0 \sim 59)\) & (32) [1] & Minute & (0~59) & (d) \([1]\) & Minute & (0~59) \\
\hline (51) [2] & Second & (0~59) + & (52) [2] & Second & (0~59) & (d) [2] & Second & (0~59) \\
\hline (51) [3] & - & & (52) [3] & - & & (d) [3] & - & \\
\hline (s1) [4] & Millisecond & (0~999) & (52) [4] & Millisecond & (0~999) & (d) [4] & Millisecond & (0~999) \\
\hline
\end{tabular}

For example, adding the time \(7: 48: 10: 500\) to \(6: 32: 40: 875\) results in the following operation.
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{\begin{tabular}{l}
(51) \([0]\) \\
(51) \([1]\)
\end{tabular}} & Hour: 6 & \multirow{5}{*}{+} & \multirow[t]{3}{*}{\begin{tabular}{l}
(s2) [0] \\
(s2) [1] \\
(2) 2\(]\)
\end{tabular}} & Hour: 7 & & \multirow[t]{2}{*}{\begin{tabular}{l}
(d) \([0]\) \\
(d) \([1]\)
\end{tabular}} & Hour: 14 \\
\hline & Minute: 32 & & & Minute: 48 & & & Minute: 20 \\
\hline (51) [2] & Second: 40 & & & Second: 10 & \(\square\) & (d) 22 & Second: 51 \\
\hline (51) [3] & - & & (32) [3] & - & & (d) [3] & - \\
\hline (31) [4] & 875 & & (32) [4] & 500 & & (d) [4] & 375 \\
\hline
\end{tabular}
(2) If the addition result of time exceeds 24 hours, 24 hours is subtracted from the sum to make the final operation result.
For example, if the time 20:20:20:500 is added to 14:20:30:875, the result becomes 10:40:51:375 instead of 34:40:51:375.
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline (51) [0] & Hour: 14 & & (32) [0] & Hour: 20 & & (d) \([0]\) & Hour: 10 \\
\hline (51) [1] & Minute: 20 & & (32) [1] & Minute: 20 & & (d) 11\(]\) & Minute: 40 \\
\hline (31) [2] & Second: 30 & + & (32) [2] & Second: 20 & \(\Rightarrow\) & (d) 2\(]\) & Second: 51 \\
\hline (51) [3] & - & & (32) [3] & - & & (d) [3] & - \\
\hline (51) [4] & 875 & & (32) [4] & 500 & & (d) [4] & 375 \\
\hline
\end{tabular}

\section*{®POINT}

Devices (s1) [3], ©2 [3], and (d) [3] are not used for the operation.
The clock data read by the \(S(P)\) _DATERD can be added as they are.

When the data are read by the \(S(P)\) _DATERD instruction, the data of day of week are stored between the devices that store the data of second and millisecond.
Since the data of day of week are not operated in the
\(S(P) D A T E+\) instruction, the data can be added as they are.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The values set for (51) and (2) are outside of the setting range. (Refer to Function (1))
(Error code: 4100)
- The device specified for (51), (52) or (d) exceeds the corresponding device range.
(For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{Caution}

Digit specification of bit devices can be used when the following conditions (a) and (b) are satisfied.
(a) Digit specification: K4
(b) Start device: multiple of 16

If the above conditions are not satisfied, an INSTRCT CODE ERR. (error code: 4004) occurs.

\section*{\(\square\) Program Example}

In the following program, 1 hour is added to the clock data read from the clock element when X20 turns ON, and the result is stored to D100 and the following devices.
[Structured ladder/FBD]


Reads the clock element data to D0 and the following devices.

Sets the time to D10 and the following devices.

\section*{[Operation]}
- Clock data read by the SP_DATERD instruction.

\begin{tabular}{l|c|l} 
D0 & 2005 & Year \\
D1 & 5 & Month \\
D2 & 17 & Day \\
D3 & 10 & Hour \\
D4 & 23 & Minute \\
D5 & 41 & Second \\
D6 & 2 & Day of week \\
D7 & 100 & Millisecond \(\}\) Clock data
\end{tabular}
- Addition by the SP_DATE+ instruction.
\begin{tabular}{l|c|}
\cline { 2 - 2 } & Hour: 10 \\
\cline { 2 - 2 } & Dinute: 23 \\
D4 & Min \\
Decond: 41 \\
D6 & 2(Tuesday) \\
D7 & 100 \\
\cline { 2 - 3 } &
\end{tabular}
\begin{tabular}{l|c|}
\cline { 2 - 2 } & Hour: 1 \\
\cline { 2 - 2 } & Minute: 0 \\
D12 & Second: 0 \\
D13 & - \\
D14 & 0 \\
\hline
\end{tabular}
\begin{tabular}{l|c|}
\cline { 2 - 2 } & D100 \\
Hour: 11 \\
\cline { 2 - 2 } D101 & Minute: 23 \\
D102 & Second: 41 \\
D103 & - \\
D104 & 100 \\
\cline { 2 - 3 } &
\end{tabular}

\subsection*{7.16.3 Subtraction of extended clock data}

- High Performance model QCPU: Supported if first 5 digits of the serial number are " 07032 " or later
- Process CPU: Supported if first 5 digits of the serial number are "07032" or later
- Redundant CPU: Supported if first 5 digits of the serial number are " 07032 " or later

S(P)_DATE-

\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{3}{*}{Input argument,} & EN: & \multicolumn{5}{|l|}{Executing condition} & \multicolumn{4}{|l|}{:Bit} \\
\hline & s1: & \multicolumn{5}{|l|}{Start number of the device that stores the clock data} & \multicolumn{4}{|l|}{:ANY16 (0..4)} \\
\hline & s2: & \multicolumn{5}{|l|}{Start number of the device that stores the clock data to be subtracted} & \multicolumn{2}{|l|}{:ANY16 (0..4)} & & \\
\hline \multirow[t]{6}{*}{Output argument,} & \[
\begin{aligned}
& \text { ENO: } \\
& \text { d: }
\end{aligned}
\] & \multicolumn{5}{|l|}{\begin{tabular}{l}
Execution result \\
Start number of the device that stores the clock data of the subtraction result
\end{tabular}} & \multicolumn{2}{|l|}{\begin{tabular}{l}
:Bit \\
:ANY16 (0..4)
\end{tabular}} & & \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..alin} & \multirow[t]{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (51) & - & & & & & & & & \\
\hline & (52) & - & & & & & & & & \\
\hline & (d) & - & & & & & & & & \\
\hline
\end{tabular}

\section*{Function}
(1) Subtracts the clock data specified for \(\Im_{2}\) ) from the clock data specified for \((51\), and stores the result to © and the following devices.
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline & & \multicolumn{3}{|l|}{Setting data} & \multicolumn{3}{|l|}{Setting data} & \multirow[t]{2}{*}{Setting data
\[
(0 ~ 23)
\]} \\
\hline (51) [0] & Hour & \((0 \sim 23)\) & (s2) [0] & Hour & (0~23) & (d) [0] & Hour & \\
\hline (31) [1] & Minute & (0~59) & (s2) [1] & Minute & (0~59) & (d) [1] & Minute & (0~59) \\
\hline (51) [2] & Second & (0~59) - & (s2) [2] & Second & (0~59) & (d) [2] & Second & (0~59) \\
\hline (51) [3] & - & & (s2) [3] & - & & (d) [3] & - & \\
\hline (51) [4] & Millisecond & (0~999) & (52) [4] & Millisecond & (0~999) & (d) [4] & Millisecond & (0~999) \\
\hline
\end{tabular}

For example, subtracting the time 3:50:10:500 from 10:40:20:875 results in the following operation.

(2) If the operation results in a negative number, 24 is added to the result to make a final operation result.
For example, if the time 10:42:12:500 is subtracted from 4:50:32:875, the result becomes 18:8:20:375 instead of \(-6: 8: 20: 375\).
\begin{tabular}{|c|c|}
\hline (51) [0] & Hour: 4 \\
\hline (51) [1] & Minute: 50 \\
\hline (51) [2] & Second: 32 \\
\hline (51) [3] & - \\
\hline (51) [4] & 875 \\
\hline
\end{tabular}

\begin{tabular}{r|c|} 
(d) \([0]\) & Hour: 18 \\
\cline { 2 - 2 } (d) \([1]\) & Minute: 8 \\
\cline { 2 - 2 } (d) \([2]\) & Second: 20 \\
(d) \([3]\) & - \\
\cline { 2 - 2 } & - \\
\cline { 2 - 2 } & 375 \\
\hline
\end{tabular}

\section*{XPOINT}

Devices (51) [3], (22 [3], and (c) [3] are not used for the operation.
The clock data read by the \(S(P)\) _DATERD can be added as they are.


\footnotetext{
When the data are read by the \(S(P)\) DATERD instruction, the data of day of week are stored between the devices that store the data of second and millisecond.
Since the data of day of week are not operated in the
S(P)_DATE- instruction, the data can be added as they are.
}

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The values set for (51) and (®2) are outside of the setting range. (Refer to Function (1))
(Error code: 4100)
- The device specified for (51), (s2) or (d) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{© Caution}

Digit specification of bit devices can be used when the following conditions (a) and (b) are satisfied.
(a) Digit specification: K4
(b) Start device: multiple of 16

If the above conditions are not satisfied, an INSTRCT CODE ERR. (error code: 4004) occurs.

\section*{\(\triangle\) Program Example}

In the following program, the clock data stored in D10 and the following devices are subtracted from the clock data read from the clock element when X1C turns ON, and the result is stored to D100 and the following devices.

\section*{[Structured ladder/FBD]}


Reads the clock element data to D0 and the following devices.

Sets the time to D10 and the following devices.

\section*{[Operation]}
- Clock data read by the SP_DATERD instruction.
\(\begin{array}{l|c|l}\text { D0 } & 2005 & \text { Year } \\\)\cline { 2 - 3 } & 2 & \text { Month } \\ \text { D1 } & 2 & \\ \text { D2 } & 23 & \text { Day } \\ \text { D3 } & 8 & \\ \text { D4 } & \text { Hour } \\ \text { D5 } & 42 & \text { Minute } \\ \text { D6 } & 1 & \text { Second }\end{array}\(\}\) Clock data
- Subtraction by the SP_DATE- instruction.
\begin{tabular}{c|c|}
\cline { 2 - 2 } D3 & Hour: 8 \\
\cline { 2 - 3 } D4 & Minute: 42 \\
\cline { 2 - 3 } & Second: 1 \\
D6 & 3(Wednesday) \\
D7 & 997 \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline & D10 & Hour: 10 \\
\hline & D11 & Minute: 40 \\
\hline - & D12 & Second: 10 \\
\hline & D13 & - \\
\hline & D14 & 500 \\
\hline
\end{tabular}

8:42:1:997-10:40:10:500
\begin{tabular}{l|c|}
\cline { 2 - 2 } & D100 \\
D101 & Hour: 22 \\
\cline { 2 - 2 } & Minute: 1 \\
D102 & Second: 51 \\
D103 & - \\
\cline { 2 - 2 } & - \\
\hline
\end{tabular}


\subsection*{7.17 Program Control Instructions}

\subsection*{7.17.1 Program standby}

PSTOP(P)
P: Executing condition


instructions.
PSTOP PSTOPP
\begin{tabular}{lll} 
Input argument, & \(\mathrm{EN}:\) & \begin{tabular}{l} 
Executing condition \\
\(\mathrm{s}:\)
\end{tabular} \\
Character string data of the file name whose program is set to : String \\
the standby type, or start number of the device that stores \\
character string data
\end{tabular}\(\quad:\) Bit
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J:..al} & \multirow[b]{2}{*}{U:...iga} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{\begin{tabular}{l}
Constant \\
\$
\end{tabular}} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline
\end{tabular}

\section*{Function}
(1) Places the program whose file name is stored in the device specified for © in a standby type.
(2) Only the programs stored in the drive number 0 (program memory/built-in RAM) can be set as the standby type.
(3) The specified program is placed in a standby type when END processing is performed.
(4) This instruction will be given priority even in cases when a program execution type has been specified in the parameters.
(5) It is not necessary to specify the extension (.QPG) with the file name.
(Only .QPG files will be acted on.)

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The program with the specified file name does not exist.
(Error code: 2410)
- The program type of the file name specified for (s) is the SFC program.
(Error code: 2412)
- The file name target device (s) exceeds the corresponding device range.
(Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, the program with the file name \(A B C\) is set in a standby type when \(X 0\) turns ON.
[Structured ladder/FBD]


\footnotetext{
[ST]
PSTOPP(X0,"ABC");
}

\subsection*{7.17.2 Program output OFF standby}


Input argument,
\begin{tabular}{ll} 
EN: & Executing condition \(\quad:\) Bit \\
\(\mathrm{s}:\) & File name whose program is set to the standby type by turning :String \\
& OFF the output, or start number of the device that stores file \\
& \begin{tabular}{l} 
name
\end{tabular} \\
ENO: & Execution result \(\quad: B i t\)
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J眔:} & \multirow[b]{2}{*}{} & \multirow[b]{2}{*}{Zn} & \multirow[t]{2}{*}{\begin{tabular}{l}
Constant \\
\$
\end{tabular}} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & \(\bigcirc\) & - \\
\hline
\end{tabular}

\section*{Function}
(1) Changes the execution type of the program whose file name stored in the device specified for ©s.
- Scan execution type: Turns OFF outputs at the next scan (Non-execution processing). Programs are set as the standby type after the subsequent scan.
- Low speed execution type: Stops the execution of the low speed execution type program and turns OFF outputs at the next scan. Programs are set as the standby type after the subsequent scan.
(2) Only the programs stored in the drive number 0 (program memory/built-in RAM) can be set as the standby type.
(3) This instruction will be given priority even in cases when a program execution type has been specified in the parameters.
(4) It is not necessary to specify the extension (.QPG) with the file name. (Only .QPG files will be acted on.)

\section*{OO Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The program with the specified file name does not exist.
(Error code: 2410)
- The file name target device (s) exceeds the corresponding device range.
(Error code: 4101)

\section*{Remark}
1. Non-execution processing is identical to the processing that is performed when the condition settings for each coil instruction are in the OFF state.
2. The operation results for each coil instruction following non-execution processing will be as follows, regardless of the ON/OFF status of the condition settings.
\begin{tabular}{|c|c|c|}
\hline OUT instruction & & Forced OFF \\
\hline \multicolumn{3}{|l|}{SET instruction} \\
\hline \multicolumn{3}{|l|}{RST instruction} \\
\hline SFT instruction & ...... & Maintains status \\
\hline \multicolumn{3}{|l|}{Basic instruction} \\
\hline \multicolumn{3}{|l|}{Application instruction} \\
\hline PLS instruction & & Processing identical to \\
\hline Pulse generation & & when condition contacts \\
\hline instruction (\%P) & & are OFF \\
\hline Current value of low-speed/high-speed timer & ... & 0 \\
\hline Current value of retentive timer & & \\
\hline Current value of counter & ...... & Retains data \\
\hline
\end{tabular}

\section*{\(\square\) Program Example}

In the following program, the program with the file name \(A B C\) is set in the nonexecutable and the standby type when X0 turns ON.
[Structured ladder/FBD]

[ST]
POFFP(X0,"ABC");

\subsection*{7.17.3 Registering program as scan execution type}

PSCAN(P)
P: Executing condition


:Bit
\begin{tabular}{lll} 
Input argument, & \(\mathrm{EN}:\) & Executing condition \\
\(\mathrm{s}:\) & \begin{tabular}{l} 
File name whose program is set to the scan execution type, or :String \\
start number of the device that stores file name
\end{tabular} \\
Output argument, & ENO: & Execution result
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...1:} & \multirow{2}{*}{U} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant \$} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline
\end{tabular}

\section*{IT Function}
(1) Sets the program whose file name is stored in the device specified for (s) in the scan execution type.
(2) Only the programs stored in the drive number 0 (program memory/built-in RAM) can be set in the scan execution type.
(3) Specified programs assume the scan execution type with END processing.

\section*{Example}

When programs \(A, B\), and \(C\) exist and program A performs the PSCAN \((P)\) instruction of program D.

(4) This instruction will be given priority even in cases when a program execution type has been specified in the parameters.
(5) It is not necessary to specify the extension (.QPG) with the file name.
(Only .QPG files will be acted on.)

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The program with the specified file name does not exist.
(Error code: 2410)
- The file name target device © exceeds the corresponding device range.
(Error code: 4101)
- The SFC program file name is specified while the SFC program with another file name is already operating. (Double SFC program activation error)
(For Universal model QCPU and LCPU)
(Error code: 4131)
(For High Performance model QCPU)

\section*{\(\triangle\) Program Example}

In the following program, the program with the file name \(A B C\) is set in the scan execution type when X0 turns ON.
[Structured ladder/FBD]

[ST]
PSCANP(X0,"ABC");

\subsection*{7.17.4 Registering program as low-speed execution type}
\(\operatorname{PLOW}(\mathrm{P}) \quad(\mathrm{P}:\) Executing condition \(\quad: \pm)\)

\begin{tabular}{|l|l|}
\hline PLOW & PLOWP \\
instructions. \\
\\
\\
\\
\\
\hline
\end{tabular}
\begin{tabular}{llll} 
Input argument, & EN: & Executing condition & :Bit \\
s: & File name whose program is set to the low-speed execution \\
type, or start number of the device that stores file name
\end{tabular}\(\quad:\) :Bit
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{} & \multirow[b]{2}{*}{U...ig:} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant \$} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline
\end{tabular}

\section*{Function}
(1) Sets the program whose file name is stored in the device specified for \((s)\) in the low-speed execution type.
(2) Only the programs stored in the drive number 0 (program memory/built-in RAM) can be set in the low-speed execution type.
(3) Specified programs assume the low-speed execution type with END processing.

\section*{Example}

When programs A, B, and C exist and program A performs the PLOW(P) instruction of program D. (Assume that the constant scan has been set.)

Waiting for constant

(4) This instruction will be given priority even in cases when a program execution type has been specified in the parameters.
(5) It is not necessary to specify the extension (.QPG) with the file name.
(Only .QPG files will be acted on.)

\section*{OOperation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The program with the specified file name does not exist.
(Error code: 2410)
- The CHK instruction exists within the program whose file name has been specified.
(Error code: 4235)

\section*{\(\triangle\) Program Example}

In the following program, the program with the file name \(A B C\) is set in the low-speed execution type when X0 turns ON.
[Structured ladder/FBD]


\section*{[ST]}

PLOWP(X0,"ABC");

\subsection*{7.17.5 Checking program execution status}

LDPCHK, ANDPCHK, ORPCHK

LDPCHK
ANDPCHK
ORPCHK

\(\begin{array}{lll}\text { Input argument, } & \mathrm{EN}: & \text { Executing condition } \\ \mathrm{s}: & \begin{array}{l}\text { Start device number that stores file name of a program whose :String } \\ \text { execution status is checked }\end{array} \\ \text { Output argument, } & \text { ENO: } & \begin{array}{l}\text { Execution result }\end{array}\end{array}\)


\section*{IT Function}
(1) Checks whether the program of the specified file name is in execution or not (non-execution).
(2) The instruction is in conduction state when the program of the specified file name is in execution, and the instruction is in non-conduction state when the program is in nonexecution status.
(3) Specify the file name without an extension (.QPG).

For example, specify "ABC" when the file name is ABC.QPG.
(4) The ORPCHK instruction performs comparison operation between the operation result of "(51), ©(5)" and EN.
Therefore, ENO always outputs ON when connecting EN to the left base line directly, or using the bit device to be always set to ON like SM400.
When using the ORPCHK instruction, connect EN and ENO in series as shown below.


\section*{Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The program with the specified file name does not exist.

In the following program, Y 10 is turned ON when the program file "ABC.QPG" is in execution.
[Structured ladder/FBD]


\section*{[ST] \\ Y10 :=LDPCHK(TRUE,"ABC");}

\section*{Remark}

Non-execution indicates that the program execution type is a standby type. Execution indicates that the program execution type is a scan execution type (including during output OFF (during non-execution processing)), low-speed execution type or fixed scan execution type.

\section*{®POINT}

The checking program execution status instruction (PCHK) is in conduction state when the program of the specified file name (target program) is in execution, and the instruction is in non-conduction state when the program is in non-execution. When the target program is set to non-execution (standby type) with the POFF(P) instruction, the PCHK instruction is in conduction state while the non-execution processing of the target program is being performed.
At the END processing of the scan where the non-execution processing is completed, the target program is put into non-execution (standby type), and the PCHK instruction is brought into non-conduction state.
Therefore, note that if the PCHK instruction is executed for the program where the non-execution processing has been completed by the \(\operatorname{POFF}(P)\) instruction, the PCHK instruction may be brought into conduction state.

The following chart shows the operation performed when program A executes the POFF \((P)\) instruction of program \(B\) and program \(C\) executes the PCHK instruction of program \(B\) with the programs being executed in order of program \(A\), program \(B\) and program C .

Program B execution type change (Scan execution type to standby type)


\subsection*{7.18 Other Instructions}

\subsection*{7.18.1 Resetting watchdog timer}

WDT(P)
P: Executing condition

\(\begin{array}{lll}\text { Input argument, } & \text { EN: } & \text { Executing condition } \\ \text { Output argument, } & \text { ENO: } & \text { Execution result } \\ \end{array}\)


\section*{Function}
(1) Resets a watchdog timer during the execution of a sequence program.
(2) Used in cases where the scan time exceeds the value set for the watchdog timer due to prevailing conditions.
If the scan time exceeds the watchdog timer setting value on every scan, change the watchdog timer settings on parameter settings of the programming tool.
(3) Make sure that the setting for t1 from step 0 to the WDT(P) instruction and the setting for t2 from the \(W D T(P)\) instruction to the END processing and FEND instruction not to exceed the setting value of the watchdog timer.

(4) The WDT(P) instruction can be used two or more times during a single scan, but a care should be taken in such cases, because longer time is required for turning OFF the output at the error occurrence.
(5) Scan time values stored in the special register will not be cleared even if the WDT(P) instruction is executed.
Accordingly, there are times when the value for the scan time for the special register is greater than the value of the watchdog timer set in the parameters.

\section*{Operation Error}

No operation error occurs in the execution of the WDT \((P)\) instruction.

\section*{\(\triangle\) Program Example}

The program in which the watchdog timer is set to 200 ms , and the scan time from step 0 to the END processing is 300 ms due to the executing conditions.

Program where scan time is 300 ms .
[When WDT instruction is used]
Program where
scan time is
150 ms .
\(\longrightarrow\)


Program where scan time is
150 ms .

\subsection*{7.18.2 Timing pulse generation}

\section*{DUTY}

\begin{tabular}{|ll|}
\hline DUTY & \\
\\
\\
\\
\\
\\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{3}{*}{Input argument,} & EN: & \multicolumn{5}{|l|}{Executing condition} & \multicolumn{4}{|l|}{:Bit} \\
\hline & n1: & \multicolumn{5}{|l|}{Number of scans to be turned ON} & \multicolumn{4}{|l|}{:ANY16} \\
\hline & n2: & \multicolumn{5}{|l|}{Number of scans to be turned OFF} & \multicolumn{4}{|l|}{:ANY16} \\
\hline \multirow[t]{7}{*}{Output argument,} & ENO: & \multicolumn{5}{|l|}{Execution result} & \multicolumn{4}{|l|}{:Bit} \\
\hline & d: & \multicolumn{5}{|l|}{User timing clock (SM420 to SM424, SM430 to M434)} & \multicolumn{4}{|l|}{:Bit} \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...al} & \multirow{2}{*}{U...iga} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & n1 & \(\bigcirc\) & \multicolumn{7}{|c|}{\(\bigcirc\)} & - \\
\hline & n2 & \(\bigcirc\) & \multicolumn{7}{|c|}{\(\bigcirc\)} & - \\
\hline & (d) & \(\bigcirc{ }^{* 1}\) & \multicolumn{7}{|c|}{-} & - \\
\hline
\end{tabular}
*1 : Only for devices SM420 to SM424 and SM430 to SM434.

\section*{T Function}
(1) Turns ON the user timing clock (SM420 to SM424, SM430 to M434) specified for (d), for the duration equivalent to the number of scans specified for n1, and turns OFF for the duration equivalent to the number of scans specified for n 2 .

(2) Scan execution type programs use from SM420 to SM424, and low-speed execution type programs use from SM430 to SM434.
(3) The following will take place if both n 1 and n 2 have been set to 0 .
(a) \(\mathrm{n} 1=0, \mathrm{n} 2 \geqq 0\) from SM420 to SM424 and from SM430 to SM434 stays OFF.
(b) \(\mathrm{n} 1>0, \mathrm{n} 2=0\) from SM420 to SM424 and from SM430 to SM434 stays ON.
(4) The data specified for \(\mathrm{n} 1, \mathrm{n} 2\), and (d) is registered to the system when the DUTY instruction is executed, and the timing pulse is turned ON and OFF by the END processing.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device specified for (d) is not from SM420 to SM424 or from SM430 to SM434.
(Error code: 4101)
- The values of n 1 and n 2 are less than 0 .
(Error code: 4100)

\section*{\(\triangle\) Program Example}

In the following program, SM420 is turned ON for 1 scan, and OFF for 3 scans when X0 turns ON.
[Structured ladder/FBD]


\section*{[ST]}

DUTY(X0,1,3,SM420);
[Operation]


\subsection*{7.18.3 Time check}

\section*{TIMCHK}


Input argument,

Output argument,
\begin{tabular}{ll} 
EN: & Executing condition \\
s1: & Device that stores the measured current value \\
s2: & Device that stores the set value for measurement \\
ENO: & Execution result \\
d: & Device to be turned ON at the end of the process
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..1:} & \multirow[b]{2}{*}{U} & \multirow[b]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K,H} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{-} & - \\
\hline (2) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{5}{|c|}{\(\bigcirc\)} & - \\
\hline (d) & \(\bigcirc\) & \multicolumn{2}{|c|}{-} & \multicolumn{5}{|c|}{-} & - \\
\hline
\end{tabular}

\section*{Function}
(1) Measures the ON time of the device used as a condition, and turns ON the device specified for \(\Omega_{2}\) if the condition device remains ON for longer than the time set to the device specified for (d).
(2) The current value of the device specified for © \(\subseteq 1\) is cleared to 0 and the device specified for (d) is turned OFF at the rising edge of the execution command.

The current value of the device specified for (51) and the ON status of the device specified for (d) are retained after the execution command turns OFF.
(3) Set the setting value of measurement in unit of 100 ms .

\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The device that cannot be specified has been specified.
(Error code: 4100)
\(\triangle\) Program Example
In the following program, the ON time of XO is set to 5 seconds, the current value storage device is set to D 0 and the device that will turn ON at time out is set to Y 10 .
[Structured ladder/FBD]

[ST]
TIMCHK(X0,D0,50,Y10);

\subsection*{7.18.4 Reading 1 byte directly from file register}

ZRRDB

Process
Redundant
Universal
LCPU

ZRRDB(P)



Input argument,
Output argument,
EN: Executing condition
n : \(\quad\) Serial byte number whose file register is read
ENO: Execution result
\(\mathrm{d}: \quad\) Start number of the device that stores read data
 instructions. ZRRDB ZRRDBP
:Bit
:ANY32
:Bit
:ANY16
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{小...and} & \multirow[b]{2}{*}{U} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline n & \multicolumn{7}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline (d) & \multicolumn{7}{|c|}{\(\bigcirc\)} & - & - \\
\hline
\end{tabular}

Function
(1) Reads the serial byte number specified for \(n\) that does not signify a block number, and stores to the lower 8 bits of the device specified for (d).
The upper 8 bits specified for (d) will become 00 H .

(2) File register numbers correspond to serial byte numbers are as indicated below.

(a) If \(\mathrm{n}=23560\) is specified, the data at the lower 8 bits of ZR 11780 will be read.

(b) If \(\mathrm{n}=43257\) is specified, the data at the upper 8 bits of \(Z R 21628\) will be read.


\section*{Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- A device number (serial byte number) that exceeds the allowable specification range has been specified.
(Error code: 4101)

\section*{Program Example}

In the following program, the lower bits of ZR16000 and the upper bits of R16003 are read when X0 turns ON, and the results are stored to Var_D100 and Var_D101.
[Structured ladder/FBD]


\section*{[ST]}

ZRRDBP(X0,32000,Var_D100); ZRRDBP(X0,32007,Var_D101);
[Operation]


\subsection*{7.18.5 Writing 1 byte directly to file register}

ZRWRB

\section*{Basic \\ High
periormanc \\ ormance Process \\ Redundant Universe Universal \\ LCPU}

ZRWRB(P)
P: Executing condition \(: \underset{\sim}{ }\)


\begin{tabular}{llll} 
Input argument, & EN: & Executing condition & :Bit \\
& \(\mathrm{n}:\) & Serial byte number whose file register is written & :ANY32 \\
& \(\mathrm{s}:\) & Start number of the device that stores data to be written & \(:\) ANY16 \\
Output argument, & ENO: & Execution result & \(:\) Bit
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...al} & \multirow[b]{2}{*}{U...agata} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K,H} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline n & \multicolumn{8}{|c|}{\(\bigcirc\)} & - \\
\hline (s) & \multicolumn{8}{|c|}{\(\bigcirc\)} & - \\
\hline
\end{tabular}

\section*{5 Function}
(1) Writes the lower bits of data stored in the device specified for ©s that does not signify a block number to the file register of the serial byte number specified for \(n\).

The upper 8 bits of data in the device specified for (s) are ignored.

(2) File register numbers correspond to serial byte numbers are as indicated below.


If \(\mathrm{n}=12340\) is specified, the data will be written to the lower 8 bits of ZR6170.


If \(n=43257\) is specified, the data will be written to the upper 8 bits of \(Z R 21628\).


\section*{O Operation Error}

In the following case, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- A device number (serial byte number) that exceeds the allowable specification range has been specified.
(Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, the data at the lower bits of Var_D101 are written to the lower bits of ZR16000 and the upper bits of R16003 when X0 turns ON.
[Structured ladder/FBD]


\section*{[ST]}

ZRWRBP(X0,32000,Var_D100); ZRWRBP(X0,32007,Var_D101);
[Operation]


\subsection*{7.18.6 Reading indirect address}


\section*{Function}
(1) Stores the indirect address of the device specified for (s) in © .

The address stored in the device specified for (d) is used when an indirect device address is performed by the sequence program.

(2) Digit specification of bit devices cannot be set for (s).
(3) Indirect address specification (@D0) cannot be set. To use indirect address specification, use a ladder program.

\section*{Operation Error}

No operation error occurs in the execution of the \(\operatorname{ADRSET}(P)\) instruction.

\subsection*{7.18.7 Numeric input from keyboard}

KEY

\begin{tabular}{|l|l|}
\hline KEY & \\
\\
\\
\\
\\
\\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{3}{*}{Input argument,} & EN: & \multicolumn{5}{|l|}{Executing condition} & \multicolumn{4}{|l|}{} \\
\hline & s: & \multicolumn{5}{|l|}{Start number or the array of the device of \((X)\) to which a numeral is input} & \multicolumn{4}{|l|}{:Array of bit (1..9)} \\
\hline & n : & \multicolumn{5}{|l|}{Number of numeric digits to be input} & \multicolumn{4}{|l|}{:ANY16} \\
\hline \multirow[t]{9}{*}{Output argument,} & ENO: & \multicolumn{5}{|l|}{Execution result} & \multicolumn{4}{|l|}{:Bit} \\
\hline & d1: & \multicolumn{5}{|l|}{Start number or the array of the device where the input numeral to be stored} & \multicolumn{4}{|l|}{:Array of ANY16 (1..3)} \\
\hline & d2: & \multicolumn{5}{|l|}{Bit device number to be turned ON at the input completion} & \multicolumn{4}{|l|}{:Bit} \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...alat} & \multirow[t]{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{\[
\begin{gathered}
\text { Constant } \\
\text { K, H }
\end{gathered}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (s) & \[
(X \text { only })^{* 1}
\] & & & & - & & - & & - \\
\hline & n & \(\bigcirc\) & & & & \(\bigcirc\) & & \(\bigcirc\) & & - \\
\hline & (d1) & - & & & & - & & - & & - \\
\hline & (d2) & \(\bigcirc\) & & & & \(\bigcirc\) & & - & & - \\
\hline
\end{tabular}
*1: Specify the array in which X is set as a device for global label.

\section*{Function}
(1) Fetches ASCII data from the 8 points of input \((X)\) specified for (s), converts it to hexadecimal values and stores the result to ©d1) and the following devices.


For example, in a case where the number of digits ( \(n\) ) has been set to 5 , and the values " 31 ", "33", "35", "37" and "39" have been input through X10 to X18 of the input module, the following will take place.

(2) Numeric input to input (X) specified for ©s) undergoes bit development and inputs to ©s [1] through © [8] as the ASCII code corresponding to the numbers.
ASCII code which can be input is from \(30 \mathrm{H}(0)\) to \(39 \mathrm{H}(9)\), and from \(41 \mathrm{H}(\mathrm{A})\) to \(46 \mathrm{H}(\mathrm{F})\).

(3) After ASCII code is input to © [1] through © [8], the strobe signal at (s) [9] turns ON to incorporate the specified numbers internally.
The strobe signal should be held at its ON or OFF status for more than one scan of the sequence program.
If this time is less than 1 scan, there will be cases when the data are correctly incorporated.

(4) Be sure to keep the execution command (condition contact for the KEY instruction execution) ON until the specified number of digits has been input.
The KEY instruction cannot be executed if the execution command turns OFF.
(5) The digits for the numbers actually fetched to (d1) [1] will be stored to the device specified for (d1), and the ASCII code data input to (d1) [2] and (d1) [3] are converted to hexadecimal BIN values, and stored.

Execution command Condition contact for the execution of KEY instruction
Strobe signal (S) [9])
ASCII code input (S) [1] to S [8])

(6) The number of digits that can be specified for n is from 1 to 8 .
(7) Fetching of the input data is completed when any of the inputs shown below has been made. At the completion, the bit device specified for (d2) is turned ON.
- When the number of digits specified for n has been input
- When the "ODH" code has been input

For example, the operations at the location specified if \(\mathrm{n}=5\) will be as indicated below.

\section*{When the}
specified number of digits are input


To perform input processing for a second time, clear the number of input digits and input data stored in (d1), and turn off the specified device for (d2) at the user program.
If (d1) is not cleared and (d2) not turned OFF, the next input processing cannot be performed.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device specified for (d) is not an input \((X)\) device.
(Error code: 4100)
- The number of digits specified for n is outside the range of 1 to 8 .

\section*{\(\triangle\) Program Example}

In the following program, the data of the 5 or fewer digits are fetched from the numeric keypad connected to the devices from Var_X20 (X20) and the following devices when X0 turns ON, and they are stored to Var_D0 (D0) and the following devices.
[Structured ladder/FBD]


\section*{[ST]}

IF X0 AND NOT(M0) THEN
SET(TRUE,MO);
FMOVP(TRUE,0,3,Var_D0[1]);
END_IF;
MOVP(M0,5,D10);
KEY(M0,Var_X20,D10,Var_D0,M10);
RST(M10,M0);
RST(M10,M10);
[Operation]


\subsection*{7.18.8 Batch save and recovery of index registers}

ZPUSH, ZPOP

\section*{Basic} High
performan

Process
Redundan Universal

LCPU

ZPUSH(P)
ZPOP(P)



Input argument, EN: Executing condition :Bit
Output argument, ENO: Execution result :Bit
\(\mathrm{d}: \quad\) Start number of the device whose index registers are saved or :ANY16
recovered
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{小等:} & \multirow{2}{*}{U...idal} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{6}{|c|}{-} \\
\hline
\end{tabular}

Function

\section*{ZPUSH(P)}
(1) Saves the content of the following index registers to (d) and the following devices.
(When content of an index register are saved, (d) +0 (the number of saves made) is increased by 1.)
- Basic model QCPU: Z0 to Z9
- High Performance model QCPU: Z0 to Z15
- Universal model QCPU and LCPU: Z0 to Z19
(2) The \(\mathrm{ZPOP}(\mathrm{P})\) instruction is used for data recovery. The \(\mathrm{ZPUSH}(\mathrm{P})\) and \(\mathrm{ZPOP}(\mathrm{P})\) instructions are used as a pair and can be nested.
(3) If nesting has been done, each time the \(\mathrm{ZPUSH}(\mathrm{P})\) instruction is executed, the field used following (d) will be added, so a field large enough to accommodate the number of times the instruction will be used should be reserved from the beginning.
(4) The composition of the field used following (d) is as shown below.
- When using Basic model QCPU


When using High Performance model QCPU

- When using Universal model QCPU and LCPU
\begin{tabular}{|c|c|c|}
\hline (d) +0 & Number of saves & \\
\hline +1 & Z0 & \multirow[t]{2}{*}{4} \\
\hline +2 & Z1 & \\
\hline & ! & \multirow[t]{4}{*}{\begin{tabular}{l}
1st nesting \\
(22 words for one nesting)
\end{tabular}} \\
\hline +20 & Z19 & \\
\hline +21 & Reserved by the & \\
\hline +22 & system (2 words) & \\
\hline +23 & Z0 & \(\nabla\) \\
\hline +24 & Z1 & \multirow[b]{2}{*}{2nd nesting} \\
\hline & ! & \\
\hline
\end{tabular}

\section*{ZPOP(P)}

Recovers the contents saved in (d) and the following devices to the index register. (When the saved content is read out to the index register, (d) +0 (the number of saves made) is decreased by 1.)

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The range for the number of points to be used for (d) and later by the ZPUSH(P) instruction exceeds the corresponding device range.
- The content of (d) +0 (number of saves) is 0 in the \(\mathrm{ZPOP}(\mathrm{P})\) instruction.
(Error code: 4100)

\section*{\(\square\) Program Example}

In the following program, the data of the index register which is prior to call the subroutine program are saved to D0 and the following devices, when using the index registers within the subroutine program following P0.
[Structured ladder/FBD]

[ST]
ZPUSH(X0,D0); ZPOP(X1,D0);

\section*{Caution}

When using the ZPUSH/ZPOP instruction, do not branch a line from © .
If branched, the operation result is not output to the devices or labels of branch destination correctly.

For details of branch point of line from destination, refer to Section 3.2.

\subsection*{7.18.9 Reading module information}

UNIRD

Basic
High
Process
Redundan
Universal
LCPU


\section*{3 Function}

Reads the module information whose amount is specified for n 2 from the module specified for n 1 , and stores the information to (d) and the following devices.
(Reads the status of the actually mounted modules instead of the module type specified by I/O assignment.)

The value of n 1 is specified by the higher 3 digits of the start I/O number of the slot from which the module information is read, when it is expressed in 4 digits in hexadecimal.
< QCPU (Q mode) >

< LCPU (L26CPU-BT)>
CPU modules
(L26CPU-BT)


The details of the module information are described as follows.

\begin{tabular}{|c|c|c|c|}
\hline \multirow[b]{2}{*}{Bit} & \multirow[b]{2}{*}{Item} & \multicolumn{2}{|c|}{Description} \\
\hline & & QCPU (Q mode) & LCPU \\
\hline b0 & \multirow{4}{*}{Number of I/O points} & 000: 16 points & 001: 32 points \\
\hline b1 & & 010: 48 points & 011: 64 points \\
\hline & & 100: 128 points & 101: 256 points \\
\hline b2 & & 110: 512 points & 111: 1024 points \\
\hline b3 & \multirow{3}{*}{Module type} & \multirow[t]{3}{*}{\begin{tabular}{l}
000: Input module \\
001: Output module \\
010: I/O mixed module \\
011: Intelligent function module
\end{tabular}} & \multirow[t]{3}{*}{\begin{tabular}{l}
000: Input module \\
001: Output module \\
011: Intelligent function module \\
111: CPU built-in I/O
\end{tabular}} \\
\hline b4 & & & \\
\hline b5 & & & \\
\hline b6 & External supply power status (For future expansion) & \begin{tabular}{l}
1: External supply power is connected. \\
0 : External supply power is not connected.
\end{tabular} & 0 : Fixed \\
\hline b7 & Presence/absence of fuse blown & \begin{tabular}{l}
1: Module's fuse is blown. \\
0 : Normal
\end{tabular} & 0 : Fixed \\
\hline b8 & Online module change status/execution from the standby system & \begin{tabular}{l}
1: An attempt to read module information on the extension base unit during online module change or from the CPU module of standby system in the redundant system. \({ }^{*}\) \\
0 : Other than the above
\end{tabular} & 0 : Fixed \\
\hline b9 & Minor/medium error status & 1: Minor/medium error occurred & 0 : Normal \\
\hline b10 & \multirow[b]{2}{*}{Module error status} & \multirow[t]{2}{*}{\begin{tabular}{l}
00: No module error \\
10: Medium error
\end{tabular}} & \multirow[t]{2}{*}{\begin{tabular}{l}
01: Minor error \\
11: Serious error
\end{tabular}} \\
\hline b11 & & & \\
\hline b12 & Module standby status & 1: Normal & 0: Module error occurred \\
\hline b13 & Empty & \multicolumn{2}{|c|}{0: Fixed} \\
\hline b14 & Series type & 0: Q series module & 0: Fixed \\
\hline b15 & Module mounting status & 1: Module is mounted. & 0 : No module is mounted. \\
\hline
\end{tabular}
*1 : The Universal model QCPU used in the multiple CPU system is turned ON during the online module change of the module controlled by the other CPU.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
[High Performance model QCPU, Process CPU, Redundant CPU, Universal model QCPU, L06CPU, L06CPU-P, L26CPU, L26CPU-P, L26CPU-BT, and L26CPU-PBT]
- When n 1 is other than 0 to FFH .
(Error code: 4100)
- When n 2 is other than 0 to 256.
(Error code: 4100)
- When a total of n 1 and n 2 is equal to or greater than 257.
(Error code: 4100) [Q00/Q01CPU, L02SCPU, L02SCPU-P, L02CPU, and L02CPU-P]
- When n1 is other than 0 to \(3 F H\).
(Error code: 4100)
- When n2 is other than 0 to 64.
(Error code: 4100)
- When a total of n 1 and n 2 is equal to or greater than 65 .
(Error code: 4100)

\section*{[Q00JCPU]}
- When n 1 is other than 0 to FH .
(Error code: 4100)
- When n 2 is other than 0 to 16.
(Error code: 4100)
- When a total of n 1 and n 2 is equal to or greater than 17.
(Error code: 4100)
[QCPU and LCPU]
- For the device number specified for (d) and later, the range whose amount is specified for n 2 exceeds the corresponding device range.
(Error code: 4101)

\section*{/Program Example}

In the following program, the module information of I/O numbers 10 H to 20 H is stored to Var_D0 and the following devices when X10 turns ON.

[Structured ladder/FBD]

[ST]
UNIRD(X10,H1,2,Var_D0);

\section*{Readout result (When read to D0)}
(1) 32-point intelligent function module for \(Q\) series

- With a 48- or 64-point module, the same content as those of D1 are stored to D2 or D2 and D3 respectively.
(2) 32-point module for A series


A module is mounted as latter 16 points of a 32 -point module
- With a 48- or 64-point module, the same contents as those of D1 are stored to D2 or D2 and D3 respectively.
(3) Empty slot

(4) Online module change

(5) Module information on the extension base unit is tried to be read from the standby system of the redundant system in separate mode


Execution from the standby system
(Module information on the extension base unit is tried to be read from the standby system of the redundant system in separate mode.)
(6) 32-point intelligent function module for \(L\) series


\subsection*{7.18.10 Reading module type}

TYPERD

- Universal model QCPU: Supported if first 5 digits of the serial number are "11043" or later


Setting Data/Control Data
\begin{tabular}{|c|c|c|c|c|c|}
\hline Setting data & & Description & Setting range & Setting side & Data type \\
\hline n & \multicolumn{2}{|l|}{The value obtained by dividing "the start I/O number of the module where a module type is read" by 16} & \[
\begin{gathered}
0 \text { to } \mathrm{FFH}, \\
3 \mathrm{E} 0 \text { to } 3 \mathrm{E} 3 \mathrm{H}^{* 1}
\end{gathered}
\] & User & BIN 16 bits \\
\hline \multirow[b]{2}{*}{(d)} & (d) [0] & \multirow[b]{2}{*}{Module type} & \multirow[b]{2}{*}{Within device range} & \multirow[b]{2}{*}{System} & BIN 16 bits \\
\hline & \begin{tabular}{l}
(d) [1] to \\
(d) [9]
\end{tabular} & & & & String \\
\hline
\end{tabular}
*1 : For Universal model QCPU only
(1) Reads the module type of the slot specified for \(n\), and stores data to © and the following devices.

For Universal model QCPU, the target modules are the following six types.
- CPU module
- Input module
- Output module
- I/O combined module
- Intelligent function module
- GOT (when bus is connected)

For LCPU, the target modules are the following four types.
- CPU module
- Input module
- Output module
- Intelligent function module
(2) Apply the upper 3 digits of the start I/O number of the module where a module type is read (when the start I/O number is written in 4 digit hexadecimal) for n .
- When specify the module with 1 slot
< Universal model QCPU >


Specify with H3E0 for the model name of the CPU module.

\section*{XPOINT}

For LCPU, if the start I/O number of built-in I/O or built-in CC-Link module is specified, the CPU module model name is read.
- Specifying a module that occupies two slots

The start I/O number specified for read target module may be different from the start I/O number of the mounted slot.
For the I/O start number to be specified, refer to the manual for the module.
- Specify the upper 3 digits of the start I/O number of the read target module expressed in 4-digit hexadecimal.

\section*{Example When specifying QJ71GP21S-SX}

The start I/O number to be specified is a value to which 0010 H of the mounted module is added.

- Reading CPU module type for multiple CPU system configuration

Specify the upper 3 digits of the start I/O number of each CPU expressed in 4-digit hexadecimal.


The module type can be read by specifying the start I/O number of the module controlled by the other CPU.
(3) Stores instruction execution result to © [0], and module type to © [1] to © [9]. The following shows the values to be stored to (d).
(a) When the read target module obtains its module type internally (Ex. QJ71GP21-SX)


The following table shows the module type example to be stored to (d) [1] to (d) [9].
\begin{tabular}{|c|c|}
\hline Target module & Module type example \\
\hline CPU module & Q06UDEHCPU \\
\hline Intelligent function module & QJ71GP21-SX \\
\hline GOT & GOT1000 \\
\hline
\end{tabular}
(b) When the read target module does not obtain its module type internally (Ex. QX40)


The following table shows the string example to be stored to © [1] to (d) [9].
\begin{tabular}{|c|c|}
\hline Target module & String example \\
\hline Input module & INPUT_16 \\
\hline Output module & OUTPUT_32 \\
\hline I/O combined module & MIXED_64 \\
\hline Intelligent function module & INTELLIGENT_128 \\
\hline
\end{tabular}
[Strings that indicate module type]
- Input module: INPUT
- Output module: OUTPUT
- I/O combined module: MIXED
- Intelligent function module*1: INTELLIGENT

\footnotetext{
*1: Includes QI60 and GOT.
}
[Strings that indicate the number of points]
- 16 points: _16
- 32 points: _32
- 64 points: _64
- 128 points: _128
- 256 points: _256
- 512 points: _512
- 1024 points: _1024
(c) Others
- For an empty slot, or when performing the online program change function
- When n is not a start I/O number of module
- When n is a value within the setting range, but cannot be set for \(\mathrm{I} / \mathrm{O}\) assignment of PLC parameter


\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- When the read target module cannot be communicated because of the failure.
(Error code: 2110)
- When exceeding the device range of 10 words from the device specified for (d).
(Error code: 4101)
- When specifying a value other than 0 to \(F F H, 3 E 0\) to \(3 E 3 H\) for \(n\). (For Universal model QCPU)
(Error code: 4101)
- When specifying a value other than 0 to FFH, 3E3H for n. (For LCPU) (Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, the module type of the module mounted to the slot with the start I/O number 0020 H is stored to D0 and the following devices when X10 turns ON.
[Structured ladder/FBD]


\section*{[ST]}

TYPERD(X0,H2,D0);

\subsection*{7.18.11 Trace set/reset}

TRACE, TRACER
- Universal model QCPU: Not supported for Q00UJCPU

TRACE
TRACER

\begin{tabular}{|l|}
\hline instructions. \\
TRACE \\
TRACER \\
\\
\\
\\
\hline
\end{tabular}
\(\begin{array}{lll}\text { Input argument, } & \text { EN: } & \text { Executing condition } \\ \text { Output argument, } & \text { ENO: } & \text { Execution result }\end{array}\)
Output argument,


\section*{3 Function}

The sampling trace function collects the specified device data of a CPU module consecutively. To execute the sampling trace, turn on SM801 while SM800 is turned on.


\section*{TRACE}
(1) The TRACE instruction turns on SM803. In addition, the instruction stops the sampling trace by latching the sampling trace result, after sampling the number of the sampling trace (of after the set TRACE instruction is executed).
(2) The sampling is stopped if SM801 turns OFF during the trace execution.
(3) After the TRACE instruction is executed and the sampling trace is stopped, SM805 is turned on.
(4) Once the TRACE instruction is executed, the second and the subsequent TRACE instructions are ignored. When the TRACER instruction is executed, the TRACE instruction is enabled again.

\section*{TRACER}
(1) The TRACER instruction resets the TRACE instruction. When the TRACER instruction is executed, the TRACE instruction is enabled again.
(2) When the TRACER instruction is executed, SM803 to SM805 are turned OFF.

\section*{Remark}
1. The target devices for the sampling trace and its timing can be set with a programming tool. For details of the sampling trace, refer to the user's manual (Function Explanation, Program Fundamentals) for the CPU module used.
2. The sampling trace can be executed with a programming tool. For sampling trace execution with a programming tool, refer to the operating manual for the programming tool used.

\section*{Operation Error}

No operation error occurs in the execution of the TRACE and TRACER instructions.

\section*{\(\square\) Program Example}

In the following program, the TRACE instruction is executed when X0 turns ON, and the TRACE instruction is reset with the TRACER instruction when X 1 turns ON.
[Structured ladder/FBD]

[ST]
TRACE (X0);
TRACER(X1);

\subsection*{7.18.12 Writing data to specified file}
- Universal model QCPU: Not supported for Q00UJCPU, Q00UCPU, and Q01UCPU
- Built-in Ethernet port LCPU: Supported
- LCPU: Not supported for L02SCPU and L02SCPU-P

\section*{SP_FWRITE}

SP_FWRITE

Input argument,

Output argument,
\begin{tabular}{ll} 
EN: & Executing condition \\
\(\mathrm{s0}:\) & Drive specification \\
s1: & File name \\
s2: & Number of data to request writing \\
ENO: & Execution result \\
d0: & Control data \\
d1: & Array to be turned on the proces
\end{tabular}

\footnotetext{
Bit
:ANY16
String
:ANY16
:Bit
:Array of ANY16 (0..7)
Array of bit (0..1)
}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J:..alind} & \multirow{2}{*}{U:..aga} & \multirow{2}{*}{Zn} & \multicolumn{2}{|l|}{Constant} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & K, H & \$ & \\
\hline (50) & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & \(\bigcirc\) & - & - \\
\hline (51) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & - & - & - \\
\hline (52) & - & \multicolumn{2}{|c|}{\(\Delta^{* 1}\)} & & & \multicolumn{2}{|l|}{-} & - & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\Delta^{* 2}\)} & & & \multicolumn{2}{|l|}{-} & - & - & - \\
\hline (d1) & \(\Delta^{* 2}\) & \multicolumn{2}{|c|}{\(\Delta^{* 2}\)} & & & \multicolumn{2}{|l|}{-} & - & - & - \\
\hline
\end{tabular}
*1 : For the Universal model QCPU and LCPU, only when the number of request write data exceeds 1024, local devices and file registers set in each program cannot be used.
*2 : Local devices and the devices designated for individual programs cannot be used.
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline Setting data & \multicolumn{3}{|r|}{Description} & Setting range & Setting side & Data type \\
\hline (0) & \multicolumn{3}{|l|}{Drive specification} & 2 & User & \multirow{8}{*}{BIN 16 bits} \\
\hline \multirow[b]{3}{*}{(5)} & \multicolumn{5}{|l|}{Start number of the device that stores a file name. A file name is expressed as follows.} & \\
\hline & Device & Item & Content/Setting data & Setting range & Setting side & \\
\hline & \[
\begin{aligned}
& \text { (51) to } \\
& \text { (51) }+\square
\end{aligned}
\] & File name character string & \begin{tabular}{l}
Specify the character string of a file name. \\
- When omitting an extension, also omit the "." (Period). \\
- Limit the file name within 8 characters + period +3 characters. \\
- When 9 or more characters are used, the extension is ignored regardless of its presence, and "BIN" or "CSV" is automatically assigned as an extension.
\end{tabular} & \begin{tabular}{l}
Character \\
string
\end{tabular} & User & \\
\hline \multirow{4}{*}{(52)} & \multicolumn{5}{|l|}{Start number of the device that stores data. Written data are expressed as follows.} & \\
\hline & Device & Item & Content/Setting data & Setting range & Setting side & \\
\hline & (32) & Number of request write data & \begin{tabular}{l}
Specify the number of data to request writing. (Unit: word) \\
This data should be set in unit of word even when byte is specified for (d0) \([7]\)
\end{tabular} & \[
\begin{gathered}
1 \text { to } 480 \\
1 \text { to } 32707^{* 2}
\end{gathered}
\] & \multirow[t]{2}{*}{User} & \\
\hline & \[
\begin{aligned}
& \text { (s2) }+1 \text { to } \\
& \text { (s2) }+\square
\end{aligned}
\] & Write data & Data to request writing. & \begin{tabular}{l}
0000h to \\
FFFFH
\end{tabular} & & \\
\hline \multirow{6}{*}{(d)} & \multicolumn{5}{|l|}{Start number of the device that stores control data. The following control data are required.} & \multirow{6}{*}{BIN 16 bits} \\
\hline & Device & Item & Content/Setting data & Setting range & Setting side & \\
\hline & (d0) \([0]\) & Execution/ completion type & Specify the execution type.
\(0000 \mathrm{H}:\) Write binary data
\(0100 \mathrm{H}:\) Write data after CSV format conversion & \[
\begin{aligned}
& 0000 \mathrm{H} \\
& 0100 \mathrm{H}
\end{aligned}
\] & User & \\
\hline & (d0) 11\(]\) & (Not used) & Used by system & - & System & \\
\hline & (d0) [2] & Writing result (number of written data) & Contains the number of actually written data against the data specified for (s2) [7]. The unit for the value is determined by the data type specification. & - & System & \\
\hline & (d0) [3] & (Not used) & - & - & - & \\
\hline
\end{tabular}
*3 : Indicates the range applicable for Universal model QCPU and LCPU.
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline Setting data & & & Description & Setting range & Setting side & Data type \\
\hline \multirow[t]{3}{*}{(d)} & \[
\begin{aligned}
& \text { (d0) }[4] \\
& \text { (d0) }[5]
\end{aligned}
\] & File position & \begin{tabular}{l}
When binary data write is specified for \\
- Set the file position. \\
00000000 H :Start of the file 00000001н to FFFFFFFEH \\
: From the specified position The unit for the value is determined by the data type specification. \\
FFFFFFFFFH : Addition starts from the end of the file. \\
When CSV format write is specified for (d0) \\
- For the High Performance model QCPU of which the first 5 digits of the serial number are '01111' or earlier, always set the beginning \((0 \mathrm{H})\) of the file. \\
- For the High Performance model QCPU/ Process CPU/Redundant CPU/Universal model QCPU/LCPU of which the first 5 digits of the serial number are '01112' or later, set the file position. 00000000 H to \(\operatorname{FFFFFFFEH}\) \\
: Start of the file \\
FFFFFFFFH : Addition from the end of the file.
\end{tabular} & 00000000H to FFFFFFFFH & User & \multirow[t]{3}{*}{BIN 16 bits} \\
\hline & (10) [6] & Number of columns specification & \begin{tabular}{l}
When binary write is specified for (d0), always set 0 . \\
When CSV format write is specified for (d0), set the number of columns where data will be written. \\
0 : No columns. Regarded as one row. Other than 0 : Set to the specified number of columns.
\end{tabular} & \begin{tabular}{l}
OH to FFFFH \\
(0 to 65535)
\end{tabular} & User & \\
\hline & (10) [7] & Data type specification & \[
\begin{aligned}
& \text { 0: Word } \\
& \text { 1: Byte }
\end{aligned}
\] & 0,1 & User & \\
\hline \multirow{4}{*}{(d1)} & \begin{tabular}{l}
Bit device \\
(©1) [1] is
\end{tabular} & at turned ON so turned ON & \begin{tabular}{l}
at the completion of the processing. \\
at error completion.)
\end{tabular} & & & \multirow{4}{*}{Bit} \\
\hline & Device & Item & Content/Setting Data & Setting Range & Setting side & \\
\hline & (d1) [0] & Completion signal & Indicates the completion of the processing. ON: Completed OFF: Not completed & - & \multirow[b]{2}{*}{System} & \\
\hline & (11) [1] & Error completion signal & \begin{tabular}{l}
Indicates whether the processing is normally completed or abnormally completed. \\
ON: Error completion \\
OFF: Normal completion
\end{tabular} & - & & \\
\hline
\end{tabular}

\section*{Caution}
(1) For QCPU (Q mode), only the ATA card drive (2) can be set for (50) (drive specification). Note that when the Flash card is installed, the SP_FWRITE instruction cannot be used to perform writing.
The SRAM card, standard RAM or standard ROM drive cannot be set.
For High-speed Universal model QCPU and LCPU, only the SD memory card drive (2) can be set for (50) (drive specification).
(2) For CSV setting, the data written are decimal values.

Example Character "A" \((41 \mathrm{H}) \rightarrow 65\) is written.
Handling range: -32768 to 32767
(3) For binary write, the setting range of the file position with word specification is 00000000 H to 7FFFFFFFH and FFFFFFFFH.
For High-speed Universal model QCPU and LCPU, this instruction will not be executed while SM606 (SD memory card forced disable instruction) is on. If executed, no processing is performed.

\section*{Function}
(1) The specified number of data is written to the specified file.

Set the execution/completion type in the control data to specify whether to write binary data without any conversion or to convert binary data to CSV format data before writing it.
(The writing target is the ATA card only for QCPU (Q mode), and the SD memory card only for High-speed Universal model QCPU and LCPU.)
(2) The execution completion bit device © \({ }^{(11}\) is automatically turned on at the END processing after the completion of the instruction is detected. The bit device is turned off at the END processing in the next scan.
Use this bit device as the execution completion flag for the SP_FWRITE instruction.
When this instruction is completed abnormally, the error completion device ©11][1] is turned
ON/OFF in synchronization with the processing complete ©11[0] device. Use this device as the error completion flag for this instruction.
SM721 is turned ON during the execution of the instruction.
This instruction cannot be executed while SM721 is ON. (If an attempt is made, no processing is performed.)
When an error is detected at the execution of the instruction (before SM721 is turned ON), the processing complete device (d1] [0], the error completion device (d1] [1], and SM721 are not turned ON.
(3) Be sure to use units of word to specify the number of request write data (®2), and the file position (d0 [4] and © [0] [5]).
The following shows the method for writing binary data when number of request write data and file position are specified.

(4) When writing binary data
(a) If the extension of the target file is omitted, ".BIN" is used as an extension.
(b) When the specified file does not exist, a new file is created and the data are additionally saved from the beginning of the file.
The attributes of this new file are set using the archive attributes.
(c) When the size of the data exceeds that of the existing area in the file during the writing, the excess data are additionally saved.
(d) If the file position specified is greater than the existing file size:
- The High Performance model QCPU of which the first 5 digits of the serial number are '01111' or lower results in an error.
- The High Performance model QCPU, Process CPU, Redundant CPU, and Universal model QCPU of which the first 5 digits of the serial number are " 01112 " or later performs writing at point 0 and is completed normally.
(e) An error occurs when the saving space becomes full while data are additionally saved. In such a case, the data that are additionally saved successfully remains in the medium. The error completion is indicated after additionally saving the data as much as possible.
(5) When writing data after CSV format conversion
(a) If the extension is omitted, ".CSV" is used as an extension.
(b) When the existing file is specified:
[High Performance model QCPU of which the first 5 digits of the serial number are '01111' or lower]
File content are all deleted and data are saved from the beginning of the file.
[High Performance model QCPU and Universal model QCPU, Process CPU,
Redundant CPU, and LCPU of which the first 5 digits of the serial number are "01112" or later]
- When other than FFFFFFFFFH is set at (@0 [4], (@0 [5]), file content are all deleted and data are saved from the beginning of the file.
- When FFFFFFFFFH is set at (©0 [4], ©0 [5]), data are saved from the end of the file.
(c) When the specified file does not exist, a new file is created and the data are additionally saved from the beginning of the file.
The attributes of this new file are set using the archive attributes.
(d) An error occurs when the saving space becomes full while data are additionally saved. In such a case, the data that are additionally saved successfully remains in the medium. The error completion is indicated after additionally saving the data as much as possible.
(e) When the specified number of columns is 0 , the data are stored as single-column data in CSV format file.

\section*{Example}

When data are written after CSV format conversion and the specified number of columns is 0 .

(f) When data are written after CSV format conversion and the specified number of columns is other than 0 , the data are stored as table data with specified number of columns in a CSV format file.

\section*{Example}

When data are written after CSV format conversion and the specified number of columns is other than 0 .

(g) When data are added by the High Performance model QCPU, Process CPU, Redundant CPU, Universal model QCPU, and LCPU of which the first 5 digits of the serial number are "01112" or later.
[Specify the file to which data will be written.] (If a file exists, delete it and create a new file again.)
Execution type: CSV format File position: OH (Create a new file)
Column specification: \(4 \mathrm{H}^{* 3 * 5}\)
Start device of the write data: Var_D0
Data type specification: Word
Number of data: \(6 H^{* 3}\)

[In the addition mode, make addition to the end of the file.]

Execution type: CSV format
Column specification: \(3 \mathrm{H}^{* 3 * 5}\)
Data type specification: Word

File position: FFFFFFFH (Continuation mode)
Start device of the write data: Var_D7
Number of data: \(8 H^{* 3}\)
Column Column Column Column

Row 1 \begin{tabular}{|l|l|l|l|l|l|l|l|l|}
\hline 1 &, & 2 &, & 3 &, & 4 & \(C R\) & LF \\
\hline
\end{tabular}

Row 24 \begin{tabular}{|l|l|l|l|l|}
\hline 5 &, & 6 & CR & LF
\end{tabular} \(\begin{aligned} & \text { Device data } \\
& \text { (Data to be written) }\end{aligned}\)

*3: Unless the "number of write points" is set to an integral multiple of "column specification", the column numbers will be random.
*4: Since the last datum is always followed by the row feed code, addition normally starts at the beginning of the new row in the addition mode.
*5: If, in the addition mode, "column specification" is changed from that in the previous writing, the column numbers are shifted.
(h) Do not execute the SP_FWRITE instruction in an interrupt program.
(Otherwise, a malfunction may result.)
(i) The following shows the calculation method of the file size (total bytes) when writing CSV format file to ATA card.

Total bytes = total bytes except the last line + number of bytes of the last line
(Number of bytes of each line \(=\) number of columns* \({ }^{*}+1+\) total bytes of each data value in a line \({ }^{*}\) )
*1: Lines except the last line are specified number of columns. The number of columns of the last line may differ from the number of columns specified in the number of data to be written, therefore, calculated as follows.
(1) Calculate lines except the last line.

Lines except the last line \(=\) required number of data to be written / number of columns (reminder is rounded down)
(2) Calculate the number of columns of the last line.

Number of columns of the last line \(=\) required number of data to be written - (lines except the last line \(\times\) number of columns)
*2: The number of bytes of each data value is calculated as follows.
\begin{tabular}{c|c|c|c}
\hline \begin{tabular}{c} 
Sign of data \\
value
\end{tabular} & \begin{tabular}{c} 
Number of bytes of \\
each data value
\end{tabular} & Range of number of bytes & Example \\
\hline Positive & Digits & \begin{tabular}{l}
1 to 5 (When word is specified) \\
1 to 3 (When byte is specified)
\end{tabular} & \begin{tabular}{l}
\(12345 \ldots 5\) bytes \\
\(67 \quad \ldots 2\) bytes
\end{tabular} \\
\hline Negative & Digits+1 & \begin{tabular}{l}
2 to 6 (When word is specified) \\
2 to 4 (When byte is specified)
\end{tabular} & \begin{tabular}{l}
\(-12345 \ldots 6\) bytes \\
\(-67 \quad \ldots 3\) bytes
\end{tabular} \\
\hline
\end{tabular}

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- Drive specified for drive specification device (50 contains the medium other than the ATA card. (For QCPU (Q mode))
(Error code: 4100)
- Drive specified for drive specification device (50) contains the medium other than the SD memory card. (For Universal model QCPU and LCPU)
(Error code: 4100)
- Values specified for control data (d0) and the following devices are out of the setting range.
(Error code: 4100)
- Value specified for "number of request write data" \({ }_{(22}\) is out of the setting range, or exceeds the device range specified for \((22+1\) and the following devices.
(Error code: 4101)
- Empty space in the ATA card is insufficient. (For QCPU (Q mode)) (Error code: 4100)
- Empty space in the SD memory card is insufficient. (For Universal model QCPU and LCPU)
(Error code: 4100)
- No free space is found when an attempt is made to create a new file. (Error code: 4100)
- Invalid device is specified.
(Error code: 4004)
- Access error occurred in the ATA card. (For QCPU (Q mode))
(Error code: 4100)
- Access error occurred in the SD memory card. (For Universal model QCPU and LCPU)
(Error code: 4100)
- An unusable value is set for a file name (①).
(Error code: 4100)
- The attribute of a file name (①) is "read only".
(Error code: 4100)
- The device specified for (d0) or (d1) exceeds the corresponding device range. (For Universal model QCPU and LCPU)
(Error code: 4101)
- The file being used other functions is accessed by the SP_FWRITE instruction. (For Universal model QCPU and LCPU)
(Error code: 4100)
- Data are written to the SD memory card by the SP_FWRITE instruction, when the writeprotect switch on the card is set to lock (write inhibited). (For Universal model QCPU and LCPU)
(Error code: 4100)

\section*{\(\square\) Program Example}
(1) In the following program, 4 bytes of binary data \((00 \mathrm{H}, 01 \mathrm{H}, 02 \mathrm{H}\), and 03 H\()\) are added to the file "ABCD.BIN" in the memory card inserted to drive 2 when X10 turns ON.
- Assume that 8 points from (d0) are reserved for the control data devices.
[Structured ladder/FBD]

```

[ST]
IF X10 THEN
MOVP(TRUE,H0,Var_D0[0]);
MOVP(TRUE,HFFFF,Var_D0[4]);
MOVP(TRUE,HFFFF,Var_D0[5]);
Var_D10:="ABCD";
MOVP(TRUE,4,Var_D20[0]);
MOVP(TRUE,0,Var_D20[1]);
MOVP(TRUE,1,Var_D20[2]);
MOVP(TRUE,2,Var_D20[3]);
MOVP(TRUE,3,Var_D20[4]);
SP_FWRITE(TRUE,2,Var_D10,Var_D20[0],Var_D0,Var_M0);
END_IF;
IF Var_MO[0] THEN
IF(Var_MO[1]=FALSE)THEN
SET(TRUE, Y10);
RST(TRUE, Y11);
ELSE
SET(TRUE, Y11);
RST(TRUE, Y10);
END_IF;
END_IF;

```
(2) In the following program, 4 bytes of data \((00 \mathrm{H}, 01 \mathrm{H}, 02 \mathrm{H}\), and 03 H\()\) are created as a file name "ABCD.CSV" in the memory card inserted to drive 2 as two-column table data in CSV format, when X10 turns ON.
- The written file is displayed as follows.
[Structured ladder/FBD]

[ST]
IF X10 THEN
MOVP(TRUE,H0,Var_D0[0]);
MOVP(TRUE,2,Var_D0[6]);
MOVP(TRUE,1,Var_D0[7]);
Var_D10:="ABCD";
MOVP(TRUE,4,Var_D20[0]);
MOVP(TRUE,0,Var_D20[1]);
MOVP(TRUE,1,Var_D20[2]);
MOVP(TRUE,2,Var_D20[3]);
MOVP(TRUE,3,Var_D20[4]);
SP_FWRITE(TRUE,2,Var_D10,Var_D20[0],Var_D0,Var_M0);
END_IF;
IF Var_MO[0] THEN
IF(Var_M0[1]=FALSE)THEN
SET(TRUE, Y10); RST(TRUE, Y11);
ELSE
SET(TRUE, Y11); RST(TRUE, Y10);
END_IF;
END_IF;
- Assume that 8 points from (©0) are reserved for the control data devices.


\subsection*{7.18.13 Reading data from specified file}

SP_FREAD
- Universal model QCPU: Not supported for Q00UJCPU, Q00UCPU, and Q01UCPU
- Built-in Ethernet port LCPU: Supported
- LCPU: Not supported for L02SCPU and L02SCPU-P

\section*{SP_FREAD}

*1 : Local devices and file registers per program cannot be used as setting data.
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline Setting data & \multicolumn{3}{|r|}{Description} & Setting range & Setting side & Data type \\
\hline (50) & \multicolumn{3}{|l|}{Drive specification} & 2 & User & \multirow{10}{*}{BIN 16 bits} \\
\hline \multirow[b]{3}{*}{(51)} & \multicolumn{5}{|l|}{Start number of the device that stores a file name. A file name is expressed as follows.} & \\
\hline & Device & Item & Content/Setting data & Setting range & Setting side & \\
\hline & \[
\begin{aligned}
& \text { (51) to } \\
& \text { (51) }+\square
\end{aligned}
\] & File name character string & \begin{tabular}{l}
Specify the character string of a file name. \\
- When omitting an extension, also omit the "." (Period). \\
- Limit the file name within 8 characters + period +3 characters. \\
-When 9 or more characters are used, the extension is ignored regardless of its presence, and "BIN" or "CSV" is automatically assigned as an extension.
\end{tabular} & Character string & User & \\
\hline \multirow{6}{*}{(d)} & \multicolumn{5}{|l|}{Start number of the device that stores control data The following control data are required.} & \\
\hline & Device & Item & Content/Setting data & Setting range & Setting side & \\
\hline & (d0) [0] & Execution completion type & \begin{tabular}{l}
Specify the execution type. 0000H: Read binary data \\
0100H: Read data after CSV format conversion
\end{tabular} & \[
\begin{aligned}
& 0000 \mathrm{H} \\
& 0100 \mathrm{H}
\end{aligned}
\] & User & \\
\hline & (d0) [1] & (Not used) & Used by system & - & System & \\
\hline & (10) [2] & Number of request read data & \begin{tabular}{l}
Specify the number of data to request reading. \\
(Unit: word) \\
This data should be set in unit of word even when byte is specified for (d0) +7 .
\end{tabular} & \[
\begin{gathered}
1 \text { to } 480 \\
1 \text { to } 32767^{* 2}
\end{gathered}
\] & User & \\
\hline & (10) [3] & (Not used) & - & - & - & \\
\hline
\end{tabular}
*2 : Indicates the range applicable for Universal model QCPU and LCPU.
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \begin{tabular}{l}
Setting \\
data
\end{tabular} & & & Description & Setting range & Setting side & Data type \\
\hline \multirow[t]{3}{*}{(d0)} & \begin{tabular}{l}
(d0) 4\(]\) \\
(d0) \([5]\)
\end{tabular} & File position & \begin{tabular}{l}
When binary data read is specified for (d0). \\
- Set the file position. \\
00000000H: Start of the file 00000001н to FFFFFFFEH \\
:From the specified position The unit for the value is determined by the data type specification. \\
FFFFFFFFFH: Setting disabled \\
When CSV format read is specified for (d0) \\
- For the High Performance model QCPU of which the first 5 digits of the serial number are '01111' or lower, always set the beginning \((0 \mathrm{H})\) of the file. \\
- For the High Performance model QCPU/ Process CPU/Redundant CPU/Universal model QCPU/LCPU of which the first 5 digits of the serial number are '01112' or later, set the file position (Row). \\
00000000H: Read from the beginning of the file. \\
00000001н to FFFFFFFEH \\
:Read from the specified row. \\
FFFFFFFFH: Read continues from the previous read position.
\end{tabular} & 00000000H to FFFFFFFFH & User & \multirow[t]{7}{*}{BIN 16 bits} \\
\hline & (d0) [6] & Number of columns specification & \begin{tabular}{l}
When binary read is specified for (d0), always set 0 . \\
When CSV format read is specified for (d0), set the number of columns from where data will be read. \\
0: No columns. Regarded as one row. Other than 0 : Regarded as the specified number of columns.
\end{tabular} & \[
\begin{aligned}
& \text { OH to FFFFH } \\
& (0 \text { to } \\
& 65535)
\end{aligned}
\] & User & \\
\hline & (d0) 7\(]\) & Data type specification & \[
\begin{array}{|l|}
\hline \text { 0: Word } \\
\text { 1: Byte }
\end{array}
\] & 0,1 & User & \\
\hline \multirow{4}{*}{(d1)} & \multicolumn{5}{|l|}{Start number of the device for storing the read data.} & \\
\hline & Device & Item & Content/Setting data & Setting range & Setting side & \\
\hline & (d1) & Reading result (number of read data) & Contains the number of actually read data against the data specified for (d0)[2]. The unit for the value is determined by the data type specification. & 0 to 480 & System & \\
\hline & \[
\begin{aligned}
& \text { (d1) }[1] \text { to } \\
& \text { (d11) }+\square
\end{aligned}
\] & Reading data & Read data & 0000H to FFFFH & System & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline Setting data & & & Description & Setting range & Setting side & Data type \\
\hline \multirow{4}{*}{(d2)} & \multicolumn{5}{|l|}{\begin{tabular}{l}
Bit device that turned ON at the completion of the processing. \\
(d2)[1] is also turned ON at error completion.)
\end{tabular}} & \multirow{4}{*}{Bit} \\
\hline & Device & Item & Content/Setting data & Setting range & Setting side & \\
\hline & (d2) [0] & Completion signal & \begin{tabular}{l}
Indicates the completion of the processing. \\
ON: Completed \\
OFF: Not completed
\end{tabular} & - & \multirow[b]{2}{*}{System} & \\
\hline & (d2) [1] & Error completion signal & \begin{tabular}{l}
Indicates whether the processing is normally completed or abnormally completed. \\
ON: Error completion \\
OFF: Normal completion
\end{tabular} & - & & \\
\hline
\end{tabular}

\section*{Caution}
(1) For QCPU (Q mode), only the ATA card drive (2) can be set for (50) (drive specification). Note that when the Flash card is installed, the SP_FREAD instruction cannot be used to perform reading.
The SRAM card, standard RAM or standard ROM drive cannot be set.
For High-speed Universal model QCPU and LCPU, only the SD memory card drive (2) can be set for (50) (drive specification).
(2) For CSV setting, the data written are decimal values.

\section*{Example}

Character "A" \((41 \mathrm{H}) \rightarrow 65\) is written.
Handling range: -32768 to 32767
(3) For binary read, the setting range of the file position with word specification is 00000000 H to 7FFFFFFFH.
(4) For High-speed Universal model QCPU and LCPU, this instruction will not be executed while SM606 (SD memory card forced disable instruction) is on. If executed, no processing is performed.
(1) Data are read from the specified file.

Set the execution/completion type in the control data to specify whether to read binary data without any conversion or to convert binary data to CSV format data before reading it. (The reading target is the ATA card only for QCPU (Q mode), and the SD memory card only for High-speed Universal model QCPU and LCPU.)
(2) The execution completion bit device (d2) is automatically turned on at the END processing after the completion of the instruction is detected. The bit device is turned off at the END processing in the next scan.
Use this bit device as the execution completion flag for the SP_FREAD instruction.
When this instruction is completed abnormally, the error completion device (d2)[1] is turned ON/OFF in synchronization with the execution completion (d2][0] device. Use this device as the error completion flag for this instruction.
SM721 is turned ON during the execution of the instruction.
This instruction cannot be executed while SM721 is ON. (If an attempt is made, no processing is performed.)
When an error is detected at the execution of the instruction (before SM721 is turned ON), the processing complete device (d1] [0], the error completion device (d1] [1], and SM721 are not turned ON.
(3) Be sure to use units of word to specify the number of request read data (d0 [2], file position (d0 [4], (d0 [5]), and read data device size ©(11).
The following shows how each device data are read in binary data reading operation.

(4) When reading binary data
(a) If the extension of the target file is omitted, ".BIN" is used as an extension.
(b) When the specified file does not exist, an error occurs.
(c) If the position specified is greater than the existing file size:
- The High Performance model QCPU of which the first 5 digits of the serial number are '01111' or lower results in an error.
- The High Performance model QCPU, Universal model QCPU, Process CPU, Redundant CPU, and LCPU of which the first 5 digits of the serial number are "01112" or later performs reading at point 0 and is completed normally.
(5) When reading data after CSV format conversion
(a) The elements in CSV format file (cells for EXCEL) are read by each row. The numeric values and character strings are converted to binary data and stored to the device.
(b) If the extension is omitted, ".CSV" is used as an extension.
(c) When the specified file does not exist, an error occurs.
(d) The elements specified by the number of request read data (d0)2 are read from the beginning of the file.
When the last datum of the file is reached before the specified number of data are read:
- The High Performance model QCPU of which the first 5 digits of the serial number are '01111' or lower results in an error.
- The High Performance model QCPU, Process CPU, Redundant CPU, Universal model QCPU, and LCPU of which the first 5 digits of the serial number are "01112" or later reads the data up to the point where the reading is possible.
(e) When the specified number of columns is 0 , the data are read by ignoring the rows in CSV format file.

Example When data are read after CSV format conversion and the specified number of columns is 0 .

Data created by EXCEL
\begin{tabular}{|r|l|r|r|}
\hline & \multicolumn{1}{|c|}{ A } & B & C \\
\hline \hline 1 & Main / sub item & & Measured value \\
\cline { 1 - 1 } 2 & Length & 1 & \multicolumn{2}{c|}{3} \\
\cline { 1 - 1 } 3 & Temperature & -21 & \\
\cline { 1 - 1 } 1 & & & \\
\end{tabular}

Data saved in the CSV format


Data to be read into devices


File name
Data that was read

Control data


If the number of columns varies in each row, the data are also read by ignoring the rows.

Such file cannot be created using EXCEL. This happens when CSV file is modified by a user.

\section*{Example When the number of columns varies in each row when the data are read.}

(f) When data are read after CSV format conversion and the specified number of columns is other than 0 , the data are read as the table with specified number of columns in CSV format file. The elements outside of the specified columns are ignored.

Example When data are read after CSV format conversion and the specified number of columns is other than 0 .

Data created by EXCEL
\begin{tabular}{|l|l|r|r|r}
\hline & \multicolumn{1}{|c|}{ A } & B & C & \\
\hline 1 & Main / sub item & & Measured value \\
\cline { 1 - 1 } 2 & Length & 1 & \multicolumn{2}{c|}{3} \\
\cline { 1 - 1 } 3 & Temperature & -21 & \\
\cline { 1 - 1 } 1 & & &
\end{tabular}


Data to be read into devices


Control data


If the number of columns varies in each row, the elements outside of the specified columns are ignored and " 0 " is added to the places where elements do not exist.

\section*{Example}

When the number of columns varies in each row when the data are read.


Control data

(g) With the High Performance model QCPU, Process CPU/Redundant CPU, Universal model QCPU, and LCPU whose first 5 digits of the serial number are "01112" or later, they divides read operation into multiple times.
[Specify the row desired to start read.]

Execution type: CSV format
Column specification: 4H
Data type specification: Word

Start row: 2H
Start device of the write data: Var_D0
Number of data: 6H
Device data

[In the continuation mode, read continues from the end of the previous read position.] Execution type: CSV format Start row: FFFFFFFFH (Continuation mode) Column specification: 4 H Data type specification: Word

Start device of the write data: Var_D7
Number of data: 5 H
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline & \multicolumn{2}{|l|}{Column
\[
1
\]} & \multicolumn{2}{|l|}{Column 2} & \multicolumn{2}{|l|}{Column 3} & \multicolumn{2}{|l|}{Column 4} \\
\hline Row 1 & 1 & , & 2 & , & 3 & , & 4 & CR LF \\
\hline
\end{tabular}
Row \(2 \quad\)\begin{tabular}{|l|l|l|l|l|l|l|l|l|}
\hline 5 &, & 6 &, & 7 &, & 8 & \(C R\) & LF \\
\hline
\end{tabular}

Row 4


Row 5
\begin{tabular}{|l|l|l|l|l|l|l|l|l|}
\hline 17 &, & 18 &, & 19 &, & 20 & CR & LF \\
\hline
\end{tabular}
- When read is performed in the continuation mode, the previous addition cannot be made normally if the 'execution type', 'column specification' and 'data type specification' settings differ from those at the previous time.
- The previous addition cannot be made normally if the SP_FREAD instruction or SP_FWRITE instruction with another setting is executed while data are being read continuously in the continuation mode.
(h) When data are read after CSV format conversion, the numeric values that are out of range or the elements other than numeric values in the CSV format file to be read are converted to OH .
(i) When data are read after CSV format conversion, numeric values are read and converted as follows.
\begin{tabular}{|c|c|c|c|c|}
\hline \multicolumn{2}{|c|}{ Numeric values in CSV format } & -32768 to -1 & 0 to 32767 & 32768 to 65535 \\
\hline \multirow{2}{*}{ Word device } & Without a sign & 32768 to 65535 & 0 to 32767 & 32768 to 65535 \\
\cline { 2 - 5 } & With a sign & -32768 to -1 & 0 to 32767 & -32768 to -1 \\
\hline
\end{tabular}
(j) Do not execute this instruction in an interrupt program. (Otherwise, a malfunction may result.)

\section*{0 Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- Drive specified for drive specification device ©0 contains the medium other than the ATA card. (For QCPU (Q mode))
(Error code: 4100)
- Drive specified for drive specification device ©0 contains the medium other than the SD memory card. (For Universal model QCPU and LCPU)
(Error code: 4100)
- Values specified for control data (d0) and the following devices are out of the setting range. (Excluding (d0 [2])
(Error code: 4100)
- Value specified for number of data blocks to be read (d0)[2] is out of the setting range.
(Error code: 4101)
- Invalid device is specified.
(Error code: 4004)
- File name specified for file name character string (51) and the following devices does not exist in the specified drive.
(Error code: 2410)
- Size of read data exceeds the size of reading device.
(Error code: 4101)
- When binary data are read, the number of data in the file is less than the size specified for the number of request read data (d0 [2].
(High Performance model QCPU of which the first 5 digits of the serial number are ' \(01111^{\prime}\) or lower)
(Error code: 4100)
- Access error occurred in the ATA card. (For QCPU (Q mode)) (Error code: 4100)
- Access error occurred in the SD memory card. (For LCPU)
(Error code: 4100)
- The device specified for (d0) or (d1) exceeds the corresponding device range.
(For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, 4 bytes of binary data are read from the beginning of the file "ABCD.BIN" in the memory card inserted to drive 2 when g_bool1 turns ON.
- Assume that 8 points from I_int0 are reserved for the control data devices.
- Assume that 100 bytes from I_int10[0] are reserved for the reading devices.
[Structured ladder/FBD]


\section*{[ST]}

IF g_bool1 THEN
MOVP(TRUE, HO, I_intO[0]);
MOVP(TRUE, 2, I_int0[2]);
MOVP(TRUE, H0, I_intO[4]);
MOVP(TRUE, H0, I_intO[5]);
g_string1 :="ABCD";
SP_FREAD(TRUE,2, g_string1, I_int0, I_int10[0], g_bool2);
END_IF;
IF g_bool2[0] THEN
IF(g_bool3=FALSE)THEN
SET(TRUE, g_bool4);
RST(TRUE,g_bool5);
ELSE
SET(TRUE, g_bool5);
RST(TRUE, g_bool4);
END_IF;
END_IF;
(2) In the following program, the file "ABCD.CSV" is read to the PC card inserted to slot 0 as two-column table data in CSV format when g_bool1 turns ON.
- Assume that 8 points from I_int0 are reserved for the control data devices.
- Assume that 100 bytes from I_int10[0] are reserved for the reading devices.
- Assume that the target CSV format file contains numeric values only.
[Structured ladder/FBD]


\section*{[ST]}

IF g_bool1 THEN
MOVP(TRUE, H100, I_int0[0]);
MOVP(TRUE, 5, I_int0[2]);
MOVP(TRUE, 2, I_int0[4]);
MOVP(TRUE, H0, I_int0[5]);
g_string1:="ABCD";
SP_FREAD(TRUE,2, g_string1, I_int0, I_int10[0], g_bool2);
END_IF;
IF g_bool2[0] THEN
IF(g_bool3=FALSE)THEN
SET(TRUE, g_bool4);
RST(TRUE,g_bool5);
ELSE
SET(TRUE, g_bool5); RST(TRUE, g_bool4);
END_IF;
END_IF;

When using the SP_FREAD instruction, do not branch a line from © \({ }^{(11)}\).
If branched, the operation result is not output to the devices or labels of branch destination correctly.
For details of branch point of line from destination, refer to Section 3.2.

\subsection*{7.18.14 Writing data to standard ROM}

SP_DEVST


\section*{SP_DEVST}



Input argument,
EN: Executing condition
:Bit
:ANY32
s1: Write offset of the device data storage file
(Specified in unit of 16 bits per one point)
s2: Start number of the device to be written to the standard ROM :ANY16
n : Number of device points to be written :ANY16
Output argument,
\(\begin{array}{ll}\text { ENO: } & \text { Execution result } \\ \mathrm{d}: & \text { Array to be turned ON at the execution completion }\end{array}\)
:Bit
:Array of bit (0..1)
-d[0]: Completion
- d[1]: Error completion
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J等:} & \multirow[b]{2}{*}{U言:} & \multirow{2}{*}{Zn} & \multirow[t]{2}{*}{Constant
\[
\mathrm{K}, \mathrm{H}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (51) & - & \(\bigcirc\) & \(\bigcirc\) & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (s2) & - & \(\bigcirc\) & \(\bigcirc\) & \multicolumn{4}{|c|}{-} & - & - \\
\hline n & - & \(\bigcirc\) & \(\bigcirc\) & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d) & \(\triangle^{* 1}\) & - & \(\triangle^{* 1}\) & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}
*1 : Local devices and file registers per program cannot be used as setting data.
(1) Writes \(n\) points of device data specified for \(\overbrace{2}\) to the write offset specified for ©11 of the device data storage file in the standard ROM.
(51) is the offset from the start of device data storage file and specified by word offset (in unit that increments by 1 for each 16 bits).

(2) Since the device data write position completion (d) [0] in the standard ROM automatically turns on at execution of the END processing, which detects the completion of this instruction, and turns off with the END processing of next scan, it is used as an execution completion flag of this instruction.
(3) When this instruction is completed in error, the error completion (d) [1] turns ON/OFF at the same timing with the completion (d) [0]. This is used as an error completion flag of this instruction.
(4) SM721 turns ON during execution of this instruction.

When SM721 has already turned ON, this instruction cannot be executed. (If executed, no processing is performed.)
(5) When an error is detected at execution of this instruction, the completion (d) [0], error completion (d) [1], and SM721 do not turn ON.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The write offset specified for (51) is out of the device data storage file range.
(Error code: 4100)
- The n points from the write offset specified for (s1) is out of the device data storage file range.
(Error code: 4100)
- The range of n points specified for \(\mathrm{S}_{2}\) ) exceeds the corresponding device range.
(Error code: 4101)
- The device data storage file is not set in the PLC file setting of the PLC parameter.
(Error code: 2410)
- The device specified for (d) exceeds the corresponding device range. (Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, 10 points of device data from D100 are written to the device data storage file in the standard ROM when MO turns ON.
[Structured ladder/FBD]

[ST]
SP_DEVST(M0,3,Var_D100,10,Var_M1);

\section*{Caution}
(1) The value written to the standard ROM is the value at execution of this instruction.
(2) The standard ROM data write count (SD687, SD688) increases by the execution of the SP_DEVST instruction.
If the standard ROM data write count exceeds 100,000 , a FLASH ROM ERROR occurs. (Error code: 1610)
(3) In order to prevent the increase of the standard ROM data write count by the improper instruction execution, set an instruction execution count for the standard ROM data write (SD695) to limit the number of times that data are written to the standard ROM in a day. If the set data write count (default: 36) is exceeded, an OPERATION ERROR occurs (Error code: 4113).

\subsection*{7.18.15 Reading data from standard ROM}

\section*{S_DEVLD}


Input argument,
EN: Executing condition
s: \(\quad\) Read offset of the device data storage file
:Bit
:ANY32
:ANY16
ENO: Execution result
:Bit
d: \(\quad\) Start number of the device to be read from the standard ROM :ANY16
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting
data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J慁:} & \multirow[b]{2}{*}{U\%19\%} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K,H} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline ( 5 & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline n & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{Function}

Reads \(n\) points of device data from read offset specified for (s) of the device data storage file in the standard ROM, and stores the data to the device specified for (d).
(s) is the offset from the start of device data storage file and specified by word offset (in unit that increments by 1 for each 16 bits).


\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The address specified for (s) is out of the standard ROM range.
(Error code: 4100)
- The n points from the address specified for (s) is out of the standard ROM range.
(Error code: 4100)
- The range of \(n\) points specified for (d) exceeds the corresponding device range.
(Error code: 4101)
- The device data storage file is not set in the PLC file setting of the PLC parameter.
(Error code: 2410)

\section*{\(\square\) Program Example}

In the following program, 10 points of device data are read from the device data storage file in the standard ROM to from Var_D100 when M0 turns ON.
[Structured ladder/FBD]

[ST]
S_DEVLD(M0,3,10,Var_D100);

\section*{Caution}

When using the SP_DEVLD instruction, do not branch a line from (d).
If branched, the operation result is not output to the devices or labels of branch destination correctly.

For details of branch point of line from destination, refer to Section 3.2.

\subsection*{7.18.16 Program load from memory card}

\section*{PLOADP}

PLOADP

\begin{tabular}{|l:c|}
\hline PLOADP & indicates the following instruction. \\
\\
\\
\\
\\
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline Input argument, & EN: & Executing condition & :Bit \\
\hline & s: & Character string data of the driver number file name that stores : programs to be loaded, or start number of the device that stores character string data* \({ }^{* 1}\) & :String \\
\hline Output argument, & ENO: & Execution result & :Bit \\
\hline & d: & Device to be turned ON for one scan at the instruction completion & :Bit \\
\hline & & : Specified by "<Drive number>:<File Name>". Examp & le 1: \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{\(\mathrm{R}, \mathrm{ZR}\)} & \multicolumn{2}{|c|}{J...al} & \multirow[b]{2}{*}{U...ica} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant \$} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & \(\bigcirc\) & - \\
\hline (d) & \(\triangle^{*}{ }^{2}\) & \multicolumn{2}{|c|}{-} & & & \multicolumn{2}{|l|}{-} & - & - \\
\hline
\end{tabular}
*2 : Local devices cannot be used.

\section*{Function}
(1) The program stored in the memory card or standard ROM is transferred to the program memory (drive 0).
If the transferred program is not registered to the program setting of the PLC parameter dialog box, its program setting in the CPU module is set to the standby type. At this time, the program setting of the PLC parameter dialog box does not change. (To transfer a program with the PLOADP instruction, consecutive free spaces are required in the program memory.)
(2) The program added using the PLOADP instruction is assigned by the lowest number among the unused program numbers.
(To assign a program number manually, store the program number to be assigned to SD720.)
The following example shows when "MAIN6" is added by the PLOADP instruction.
(a) When the program numbers have been set consecutively, the new program is added at the end of the preset program numbers.
When programs number 1 to 5 have been set, a new program is added as program number 6.
\begin{tabular}{|c|c|}
\hline Program No. & Program name \\
\hline 1 & MAIN1 \\
\hline 2 & MAIN2 \\
\hline 3 & MAIN3 \\
\hline 4 & MAIN4 \\
\hline Adds"MAIN6"by the \\
PLOADP instruction. & \\
\hline 5 & MAIN5 \\
\hline
\end{tabular}
\(\leftarrow\) Added at the end.
(b) When there are multiple open program numbers, the program specified by the PLOADP instruction is added to the lowest number among them to be added.
(The empty program numbers are made when programs are deleted by the PUNLOADP instruction.)
When programs number 2 and 4 are empty, a new program is added as program number 2.
\begin{tabular}{|c|c|c|c|c|}
\hline Program No. & Program name & & Program No. & Program name \\
\hline 1 & MAIN1 & & 1 & MAIN1 \\
\hline 2 & Empty & Adds"MAIN6"by the & 2 & MAIN6 \\
\hline 3 & MAIN3 & PLOADP instruction. & 3 & MAIN3 \\
\hline 4 & Empty & \(\checkmark\) & 4 & Empty \\
\hline 5 & MAIN5 & & 5 & MAIN5 \\
\hline
\end{tabular}
\(\leftarrow\) Added to the smallest program number which is empty.
(3) Drive numbers 1, 2, and 4 can be specified. (Drive 3 cannot be specified.)
- Drive 1: Memory card (RAM)
- Drive 2: Memory card (ROM)
- Drive 4: Standard ROM
(4) An extension (.QPG) does not need to be specified for the file name.
(5) The bit device specified for (d) is turned ON during the END processing of the scan where this instruction is completed. The bit device is turned OFF at the next END processing.
(6) The PLC file settings of the loaded program are set as follows.
(a) File usage for each program

All the usage of file registers, device initial values, comments, and local devices of the program transferred by this instruction are set based on PLC file settings.
However, an error will be returned if both of the conditions below are met when the program is transferred using this instruction.
- Setting is made so that local devices are used in the PLC file setting.
- The number of programs in the program memory exceeds the number of programs set in the parameters.
To use local devices in the program transferred by this instruction, register a dummy program file in the parameter, delete the dummy file with the PUNLOADP instruction, and then load the program with the PLOADP instruction.
(b) I/O refresh setting

No settings are applied to both input and output data of the I/O refresh setting for the program transferred by this instruction.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- File name does not exist at the drive number specified for (s).
(Error code: 2410)
- The drive number specified for \({ }^{\text {s }}\) is invalid.
(Error code: 4100)
- There is not enough memory to load the specified program in drive 0. (Error code: 2413)
- The number of files indicated in the table below are registered in the program memory.
(Error code: 4101)
- The program number stored in SD720 is already used, or larger than the largest program number shown in the table below.
(Error code: 4101)
- The program file which has the same name as the program file to be loaded already exists.
(Error code: 2410)
- The file size of the local devices cannot be reserved.
(Error code: 2401)
\begin{tabular}{c|c|c}
\hline CPU model name & Program memory (number of files) & Largest program No. \\
\hline Q02 \((\mathrm{H}) \mathrm{CPU}\) & 28 & 28 \\
\hline Q06HCPU & 60 & 60 \\
\hline Q12HCPU & 124 & 124 \\
\hline Q25HCPU & 124 & 124 \\
\hline Q02PHCPU & 28 & 28 \\
\hline Q06PHCPU & 60 & 60 \\
\hline Q12PHCPU & 124 & 124 \\
\hline Q25PHCPU & 124 & 124 \\
\hline
\end{tabular}

\section*{\(\triangle\) Program Example}

In the following program, "ABCD.QPG" stored in the drive 4 is transferred to drive 0 and set the program in standby type when MO turns ON.
[Structured ladder/FBD]

[ST]
PLOADP(Var_M0,"4:ABCD",Var_M10);

\section*{Caution}
(1) The PLOADP, PUNLOADP and PSWAPP instructions cannot be executed simultaneously. If two or more of the above instructions are executed simultaneously, the instruction executed later will not be executed.
When using the above instructions, provide interlocks manually to avoid simultaneous execution
(2) Do not execute this instruction in an interrupt program.
(Otherwise, a malfunction may result.)
(3) To execute the program that was transferred to the program memory with the PLOADP instruction, execute the scan execution type with the PSCAN instruction (Section 7.17.3).
(4) The "PLOADP instruction" and "Write during RUN" processing cannot be executed simultaneously.
(a) When a write during RUN request is given during processing of the PLOADP instruction, write during RUN is delayed. Write during RUN is started after the processing of the PLOADP instruction is completed.
(b) When the PLOADP instruction is executed during write during RUN, the processing of the PLOADP instruction is delayed.
The processing of the PLOADP instruction is started after completion of write during RUN.
(5) Do not execute "Read from PLC" or "Verify with PLC" and the instruction simultaneously. If the instruction is executed, "Read from PLC" or "Verify with PLC" is not complete normally because the program file status stored in the program memory is changed. If executed, execute "Read from PLC" or "Verify with PLC" again after the instruction completion.

\subsection*{7.18.17 Program unload from program memory}

\section*{PUNLOADP}

\section*{PUNLOADP}

\begin{tabular}{lll} 
Input argument, & EN: & Executing condition \\
\(\mathrm{s}:\) & \begin{tabular}{l} 
Character string data of the program file name to be unloaded, : String \\
or start number of the device that stores character string data
\end{tabular} \\
Output argument, & ENO: & \begin{tabular}{l} 
Execution result
\end{tabular} \\
& \(\mathrm{d}:\) & \begin{tabular}{l} 
Device to be turned ON for one scan at the instruction \\
completion
\end{tabular} \\
& & :Bit
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...al} & \multirow[b]{2}{*}{U...iga} & \multirow[b]{2}{*}{Zn} & \multirow[t]{2}{*}{\begin{tabular}{l}
\[
\begin{gathered}
\text { Constant } \\
\$
\end{gathered}
\] \\
\$
\end{tabular}} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d) & \(\triangle^{* 1}\) & \multicolumn{2}{|c|}{-} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}
*1 : Local devices cannot be used.

\section*{Function}
(1) The standby type program stored in the program memory (drive 0 ) is deleted from the program memory.
(The program set as the 'scan execution type' with the PSCAN instruction or the program set as the 'low speed execution type' with the PLOW instruction cannot be deleted.)
(2) The program number deleted by the PUNLOADP instruction becomes 'Empty'.

When programs number 1 to 5 have been set in the program setting of the PLC parameter dialog box, deleting program number 2 with this instruction makes program number 2 empty.
\begin{tabular}{|c|c|c|c|c|c|}
\hline Program No. & Program name & & Program No. & Program name & \multirow{6}{*}{\(\leftarrow\) Program No. 2 is deleted.} \\
\hline 1 & MAIN1 & & 1 & MAIN1 & \\
\hline 2 & MAIN2 & \multirow[t]{4}{*}{Deletes"MAIN2"by the PUNLOADP instruction} & 2 & Empty & \\
\hline 3 & MAIN3 & & 3 & MAIN3 & \\
\hline 4 & MAIN4 & & 4 & MAIN4 & \\
\hline 5 & MAIN5 & & 5 & MAIN5 & \\
\hline
\end{tabular}
(3) An extension (.QPG) does not need to be specified for the file name.
(4) The bit device specified for © ( is turned ON during the END processing of the scan where this instruction is completed. The bit device is turned OFF at the next END processing.
(5) When the programmable controller CPU is powered ON or the CPU module is reset after execution of the PUNLOADP instruction, the following operation is performed.
(a) When boot setting has been set in the PLC parameter dialog box, the program where the boot setting has been set is transferred to the program memory.
When the program deleted by the PUNLOADP instruction is not to be executed, delete the corresponding program name from the boot setting and program setting of the PLC parameter dialog box.
(b) When boot setting has not been set in the PLC parameter dialog box, "FILE SET ERROR (error code: 2400)" occurs.
1) When the program deleted by the PUNLOADP instruction is not to be executed, delete the corresponding program name from the program setting of the PLC parameter dialog box.
2) When the program deleted by the PUNLOADP instruction is to be executed again, write the corresponding program to the CPU module.

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The file name specified for (s) does not exist. (Error code: 2410)
- The program specified for ©s is not a standby program, or is a program in execution. (Error code: 4101)

\section*{Program Example}

In the following program, "ABCD.QPG" stored in the drive 0 is deleted from the memory when MO turns from OFF to ON.
[Structured ladder/FBD]


\section*{[ST]}

PUNLOADP(M0,"ABCD",Var_M10);

\section*{Caution}
(1) The PLOADP, PUNLOADP and PSWAPP instructions cannot be executed simultaneously. If two or more of the above instructions are executed simultaneously, the instruction executed later will not be executed.
When using the above instructions, provide interlocks manually to avoid simultaneous execution.
(2) Do not execute this instruction in an interrupt program.
(Otherwise, a malfunction may result.)
(3) The program to be deleted from the program memory by this instruction should be set to the "standby execution type" with the PSTOP instruction beforehand. (Section 7.17.1)
(4) The "PUNLOADP instruction" and "Write during RUN" processing cannot be executed simultaneously.
(a) When a write during RUN request is given during processing of the PUNLOADP instruction, write during RUN is delayed.
Write during RUN is started after the processing of the PUNLOADP instruction is completed.
(b) When the PUNLOADP instruction is executed during write during RUN, the processing of the PUNLOADP instruction is delayed.
The processing of the PUNLOADP instruction is started after completion of write during RUN.
(5) Do not execute "Read from PLC" or "Verify with PLC" and the instruction simultaneously. If the instruction is executed, "Read from PLC" or "Verify with PLC" is not complete normally because the program file status stored in the program memory is changed.
If executed, execute "Read from PLC" or "Verify with PLC" again after the instruction completion.

\subsection*{7.18.18 Load and unload}

\section*{PSWAPP}

PSWAPP

\begin{tabular}{|c|}
\hline \(\square\) indicates the following instruction. PSWAPP \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline \multirow[t]{3}{*}{Input argument,} & EN: & Executing condition & :Bit \\
\hline & s1: & Character string data of the program file name to be unloaded, or start number of the device that stores character string data & :String \\
\hline & s2: & Character string data of the driver number file name that stores programs to be loaded, or start number of the device that stores character string data* \({ }^{* 1}\) & :String \\
\hline \multirow[t]{3}{*}{Output argument,} & \multirow[t]{3}{*}{\[
\begin{aligned}
& \text { ENO: } \\
& \text { d: }
\end{aligned}
\]} & Execution result & :Bit \\
\hline & & Device to be turned ON for one scan at the instruction completion & :Bit \\
\hline & & 1 : Specified by "<Drive number>:<File Name>". Exam & 1 : \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..nlin} & \multirow{2}{*}{U:...ican} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{\[
\begin{gathered}
\text { Constant } \\
\$
\end{gathered}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s1) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & \(\bigcirc\) & - \\
\hline (s2) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & \(\bigcirc\) & - \\
\hline (d) & \(\triangle^{* 2}\) & \multicolumn{2}{|c|}{-} & & & \multicolumn{2}{|l|}{-} & - & - \\
\hline
\end{tabular}
*2 : Local devices cannot be used.

\section*{Function}
(1) The standby type program stored in the program memory (drive 0 ) specified for \((51)\) is deleted from the program memory, and at the same time, the program stored in the memory card or standard ROM specified for \(\Im_{2}\) is transferred to the program memory and placed in standby type.
(When the program is transferred to the program memory, the program must have a continuous free space.)
The program set as the 'scan execution type' with the PSCAN instruction or the program set as the 'low speed execution type' with the PLOW instruction cannot be deleted.
(2) The program to be transferred to the program memory by the PSWAPP instruction will have the program number of the program to be deleted from the program memory.
(If there is an empty program number before the program to be deleted from the program memory, the program to be transferred to the program memory will not have the empty program number.)
When program number 2 is "Empty", the program transferred to the program memory is registered as program number 3 by the program swapping of program number with this instruction.
\begin{tabular}{|c|c|c|c|c|}
\hline Program No. & Program name & \multirow{6}{*}{Swaps"MAIN3"with"MAIN6" by the PSWAPP instruction} & Program No. & Program name \\
\hline 1 & MAIN1 & & 1 & MAIN1 \\
\hline 2 & Empty & & 2 & Empty \\
\hline 3 & MAIN3 & & 3 & MAIN6 \\
\hline 4 & MAIN4 & & 4 & MAIN4 \\
\hline 5 & MAIN5 & & 5 & MAIN5 \\
\hline
\end{tabular}
(3) Drive numbers 1, 2, and 4 can be specified. (Drive 3 cannot be specified.)
- Drive 1: Memory card (RAM)
- Drive 2: Memory card (ROM)
- Drive 4: Standard ROM
(4) An extension (.QPG) does not need to be specified for the file name.
(5) The bit device specified for (d) is turned ON during the END processing of the scan where this instruction is completed. The bit device is turned OFF at the next END processing.
(6) When the programmable controller CPU is powered ON or the CPU module is reset after execution of the PSWAPP instruction, the following operation is performed.
(a) When boot setting has been set in the PLC parameter dialog box, the program where the boot setting has been set is transferred to the program memory.
When the program replaced by the PSWAPP instruction is to be executed, change the boot setting and program setting of the PLC parameter dialog box for the corresponding program name.
(b) When boot setting has not been set in the PLC parameter dialog box, "FILE SET ERROR (error code: 2400)" occurs.
1) When the program replaced by the PSWAPP instruction is to be executed, change the program setting of the PLC parameter dialog box for the corresponding program name.
2) To execute the program set in the program setting of the PLC parameter dialog box, write the corresponding program to the CPU module again.
(7) The PLC file settings of the program on which the PSWAPP instruction has been executed are set as follows.
(a) File usage for each program

All the usage of file register, device initial value, comment, and local device of the program after the execution of the PSWAPP instruction are set based on PLC file settings.
(b) I/O refresh setting

No settings are applied to both input and output data of the I/O refresh setting for the program after the PSWAPP instruction has been executed.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- The drive number or the file name specified for \((51)\) or \((2)\) does not exist.
(Error code: 2410)
- The drive number specified for (s1) is invalid.
(Error code: 4100)
- There is not enough memory to load the specified program in drive 0. (Error code: 2413)
- The program specified for (s) is not a standby program, or is a program in execution.
(Error code: 4101)

\section*{\(\boxed{\square}\) Program Example}

In the following program, as "EFGH.QPG" stored in drive 0 is deleted from the memory, "ABCD.QPG" stored in drive 4 is transferred to drive 0 , and set the program in standby type when M0 turns from OFF to ON.
[Structured ladder/FBD]

[ST]
PSWAPP(M0,"EFGH","4:ABCD",Var_M10);

\section*{Caution}
(1) The PLOADP, PUNLOADP and PSWAPP instructions cannot be executed simultaneously. If two or more of the above instructions are executed simultaneously, the instruction executed later will not be executed.
When using the above instructions, provide interlocks manually to avoid simultaneous execution.
(2) Do not execute this instruction in an interrupt program.
(Execution of this instruction in an interrupt program can cause a malfunction.)
(3) The "PSWAPP instruction" and "Write during RUN" processing cannot be executed simultaneously.
(a) When a write during RUN request is given during processing of the PSWAPP instruction, write during RUN is delayed.
Write during RUN is started after the processing of the PSWAPP instruction is completed.
(b) When the PSWAPP instruction is executed during write during RUN, the processing of the PSWAPP instruction is delayed.
The processing of the PSWAPP instruction is started after completion of write during RUN.
(4) Do not execute "Read from PLC" or "Verify with PLC" and the instruction simultaneously. If the instruction is executed, "Read from PLC" or "Verify with PLC" is not complete normally because the program file status stored in the program memory is changed. If executed, execute "Read from PLC" or "Verify with PLC" again after the instruction completion.

\subsection*{7.18.19 File register high-speed block transfer}

\begin{tabular}{|lc|}
\hline RBMOV & \\
\hline RBMOVP \\
\\
\\
\\
\\
\\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{3}{*}{Input argument,} & EN: & \multicolumn{5}{|l|}{Executing condition} & \multicolumn{4}{|l|}{:Bit} \\
\hline & s : & \multicolumn{5}{|l|}{Start number of the device that stores data to be transferred} & \multicolumn{4}{|l|}{:ANY16} \\
\hline & n : & \multicolumn{5}{|l|}{Number of transfers} & \multicolumn{4}{|l|}{:ANY16} \\
\hline \multirow[t]{7}{*}{Output argument,} & ENO: & \multicolumn{5}{|l|}{Execution result} & \multicolumn{4}{|l|}{:Bit} \\
\hline & d: & \multicolumn{5}{|l|}{Start number of the device at the transfer destination} & \multicolumn{4}{|l|}{:ANY16} \\
\hline & \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...1:} & \multirow[t]{2}{*}{} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant
\[
\mathrm{K}, \mathrm{H}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & & Bit & Word & & Bit & Word & & & & \\
\hline & (5) & \multicolumn{6}{|c|}{\(\bigcirc\)} & \multicolumn{2}{|c|}{-} & - \\
\hline & n & \multicolumn{6}{|c|}{\(\bigcirc\)} & \multicolumn{2}{|c|}{\(\bigcirc\)} & - \\
\hline & (d) & \multicolumn{6}{|c|}{\(\bigcirc\)} & \multicolumn{2}{|c|}{-} & - \\
\hline
\end{tabular}

\section*{Function}
(1) Batch transfers \(n\) points of 16-bit data from the device specified for ©s to the \(n\) points of devices from ©

(2) The transfer is available even if there is an overlap between the source and destination devices.
For the transmission to the smaller number of device, the data are transferred from © . For the transmission to the larger number of device, the data are transferred from \((5)+(n-1)\).
However, as shown in the example below, when transferring data from \(R\) to \(Z R\), or from \(Z R\) to \(R\), the range to be transferred (source) and the range of destination must not overlap.
- ZR transfer range ((specified start number of \(Z R\) ) to (specified start number of \(Z R+\) the number of transfers -1))
- R transfer range ((specified start number of \(R+\) file register block number \(\times 32768\) ) to (specified start number of \(R+\) file register block number \(\times 32768+\) the number of transfers-1))

\section*{Example}

Transfer ranges of ZR and R overlap when transferring 10000 points of data from ZR30000 (source) to R10 (block No. 1 of the destination).
- ZR transfer range \(\rightarrow\) (30000) to (30000+10000-1) \(\rightarrow\) (30000) to (39999)
- R transfer range \(\rightarrow(10+(1 \times 32768))\) to \((10+(1 \times 32768)+10000-1)\)
\[
\rightarrow(32778) \text { to (42777) }
\]

Therefore, the range from 32778 to 39999 overlaps.

(3) When (s) is a word device and (d) is a bit device, the number of bits specified by the digit specification for bit devices will be transferred. If K1Y30 has been specified for (d), the lower 4 bits of the word device specified for (s) will be transferred.


\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The device range of \(n\) points from (s) or (d) exceeds the corresponding device range.
(Error code: 4101)
- The file register is not specified for either (s) or © .
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the lower 4 bits of data in R66 to R69 are output to the devices from Y30 to Y3F in units of 4 points.
[Structured ladder/FBD]

[ST]
RBMOVP(SM402,R66,4,K1Y30);
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multicolumn{6}{|r|}{Before execution (source of transfer)} \\
\hline \multicolumn{6}{|r|}{b15- - b4b3--b0} \\
\hline R66 & 1 & 1 & 1 & 0 & 1 \\
\hline R67 & 0 & 0 & 0 & & 0 \\
\hline R68 & 1 & 0 & 0 & & 1 \\
\hline R69 & 0 & 1 & 1 & 0 & \\
\hline
\end{tabular}

After execution (destination of transfer)
\begin{tabular}{|c|c|c|c|c|}
\hline 1 & 1 & 0 & 1 & Y33 to Y30 \\
\hline 0 & 0 & 0 & 0 & Y37 to Y34 \\
\hline 0 & 0 & 1 & 1 & Y3B to Y38 \\
\hline 1 & 1 & 0 & 1 & Y3F to Y3C \\
\hline
\end{tabular}

Ignored
(2) In the following program, the data in X20 to X2F are output to the devices from R100 to R103 in units of 4 points.
[Structured ladder/FBD]

[ST]
RBMOVP(SM402,K1X20,4,R100);


\section*{Caution}

When using the RBMOV instruction, do not branch a line from © (
If branched, the operation result is not output to the devices or labels of branch destination correctly.

For details of branch point of line from destination, refer to Section 3.2.

\section*{XPOINT}

The RBMOV \((P)\) instruction is useful when batch transferring a large quantity of file register data with QnHCPU/QnPHCPU/QnPRHCPU.
The comparison of processing speed with the BMOV instruction is as follows.
(1)Transferring data from file registers to internal devices or from internal devices to file registers
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline \multirow{2}{*}{CPU} & \multirow{2}{*}{Instruction} & \multirow[t]{2}{*}{Target memory where file register is stored} & \multicolumn{2}{|l|}{1 word} & \multicolumn{2}{|l|}{1000 words} & \multicolumn{2}{|r|}{10000 words} \\
\hline & & & \multicolumn{2}{|l|}{Min. \(\quad\) Max.} & Min. & Max. & Min. & Max. \\
\hline \multirow{6}{*}{\begin{tabular}{l}
QnHCPU \\
QnPHCPU \\
QnPRHCPU
\end{tabular}} & \multirow{3}{*}{RBMOV} & Standard RAM & \multicolumn{2}{|c|}{\(20.0 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{91.0رs} & \multicolumn{2}{|c|}{\(775.0 \mu \mathrm{~s}\)} \\
\hline & & SRAM card & \multicolumn{2}{|c|}{22.0رs} & \multicolumn{2}{|c|}{\(305.0 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(2900.0 \mu \mathrm{~s}\)} \\
\hline & & Flash card* \({ }^{\text {1 }}\) & \multicolumn{2}{|c|}{\(22.5 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(405.0 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(3950.0 \mu \mathrm{~s}\)} \\
\hline & \multirow{3}{*}{BMOV} & Standard RAM & \multicolumn{2}{|c|}{\(7.5 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(76.2 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(720.0 \mu \mathrm{~s}\)} \\
\hline & & SRAM card & \multicolumn{2}{|c|}{\multirow[b]{2}{*}{\(8.0 \mu \mathrm{~s}\)}} & \multicolumn{2}{|c|}{\(384.0 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(3900.0 \mu \mathrm{~s}\)} \\
\hline & & Flash card*1 & & & \multicolumn{2}{|c|}{\(418.0 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(4250.0 \mu \mathrm{~s}\)} \\
\hline \multirow{6}{*}{QnCPU} & \multirow{3}{*}{RBMOV} & Standard RAM & \multicolumn{2}{|c|}{\(45.5 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(215.0 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(1850.0 \mu \mathrm{~s}\)} \\
\hline & & SRAM card & \multicolumn{2}{|c|}{\multirow[t]{2}{*}{\(49.5 \mu \mathrm{~s}\)}} & \multicolumn{2}{|c|}{\multirow[b]{2}{*}{540.0رs}} & \multicolumn{2}{|c|}{\multirow[t]{2}{*}{\(5150.0 \mu \mathrm{~s}\)}} \\
\hline & & Flash card*1 & & & & & & \\
\hline & \multirow{3}{*}{BMOV} & Standard RAM & \multicolumn{2}{|c|}{17.5 \(\mu \mathrm{s}\)} & \multicolumn{2}{|c|}{\(177.0 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{1700.0رs} \\
\hline & & SRAM card & \multicolumn{2}{|c|}{\multirow[t]{2}{*}{\(18.0 \mu \mathrm{~s}\)}} & \multicolumn{2}{|c|}{\(500.0 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(5050.0 \mu \mathrm{~s}\)} \\
\hline & & Flash card* \({ }^{*}\) & & & \multicolumn{2}{|c|}{\(572.0 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(5800.0 \mu \mathrm{~s}\)} \\
\hline \multirow{6}{*}{\begin{tabular}{l}
QOOUCPU \\
Q01UCPU
\end{tabular}} & \multirow{3}{*}{RBMOV} & Standard RAM & \(12.2 \mu \mathrm{~s}\) & \(34.9 \mu \mathrm{~s}\) & \(121.5 \mu \mathrm{~s}\) & \(145.1 \mu \mathrm{~s}\) & \(1111.5 \mu \mathrm{~s}\) & \(1135.1 \mu \mathrm{~s}\) \\
\hline & & SRAM card \({ }^{*}{ }^{2}\) & - & - & - & - & - & - \\
\hline & & Flash card*2 & - & - & - & - & - & - \\
\hline & \multirow{3}{*}{BMOV} & Standard RAM & \(7.3 \mu \mathrm{~s}\) & 13.8 \(\mu \mathrm{s}\) & \(116.5 \mu \mathrm{~s}\) & \(124.2 \mu \mathrm{~s}\) & \(1106.5 \mu \mathrm{~s}\) & 1114.2 \(\mu \mathrm{s}\) \\
\hline & & SRAM card \({ }^{*}{ }^{2}\) & - & - & - & - & - & - \\
\hline & & Flash card*2 & - & - & - & - & - & - \\
\hline \multirow{6}{*}{Q02UCPU} & \multirow{3}{*}{RBMOV} & Standard RAM & \(9.4 \mu \mathrm{~s}\) & \(31.3 \mu \mathrm{~s}\) & \(118.5 \mu \mathrm{~s}\) & \(141.3 \mu \mathrm{~s}\) & \(1108.5 \mu \mathrm{~s}\) & 1131.3 \(\mu \mathrm{s}\) \\
\hline & & SRAM card & \(9.4 \mu \mathrm{~s}\) & 31.4 \(\mathrm{\mu s}\) & \(178.5 \mu \mathrm{~s}\) & \(201.3 \mu \mathrm{~s}\) & \(1708.5 \mu \mathrm{~s}\) & \(1731.3 \mu \mathrm{~s}\) \\
\hline & & Flash card* \({ }^{*}\) & \(9.4 \mu \mathrm{~s}\) & \(32.1 \mu \mathrm{~s}\) & \(278.5 \mu \mathrm{~s}\) & \(301.3 \mu \mathrm{~s}\) & \(2708.5 \mu \mathrm{~s}\) & \(2731.3 \mu \mathrm{~s}\) \\
\hline & & Standard RAM & \(5 \mu \mathrm{~s}\) & \(11.6 \mu \mathrm{~s}\) & \(114.5 \mu \mathrm{~s}\) & \(122.3 \mu \mathrm{~s}\) & \(1104.5 \mu \mathrm{~s}\) & \(1112.3 \mu \mathrm{~s}\) \\
\hline & \multirow[t]{2}{*}{BMOV} & SRAM card & \(5.1 \mu \mathrm{~s}\) & \(11.7 \mu \mathrm{~s}\) & \[
174.5 \mu \mathrm{~s}
\] & \(182.3 \mu \mathrm{~s}\) & \[
1704.5 \mu \mathrm{~s}
\] & \(1712.3 \mu \mathrm{~s}\) \\
\hline & & Flash card*1 & \(5 \mu \mathrm{~s}\) & \(11.6 \mu \mathrm{~s}\) & \[
274.5 \mu \mathrm{~s}
\] & \(282.3 \mu \mathrm{~s}\) & \(2704.5 \mu \mathrm{~s}\) & 2712.3 \(\mu \mathrm{s}\) \\
\hline \multirow{6}{*}{Q03UD(E)CPU} & \multirow{3}{*}{RBMOV} & Standard RAM & \(11.3 \mu \mathrm{~s}\) & \(16.8 \mu \mathrm{~s}\) & 120.7 s & \(127.1 \mu \mathrm{~s}\) & \(1110.7 \mu \mathrm{~s}\) & \(1117.1 \mu \mathrm{~s}\) \\
\hline & & SRAM card & \(11.2 \mu \mathrm{~s}\) & \(16.7 \mu \mathrm{~s}\) & \[
180.7 \mu \mathrm{~s}
\] & \(187.1 \mu \mathrm{~s}\) & \[
1710.7 \mu \mathrm{~s}
\] & \[
1717.1 \mu \mathrm{~s}
\] \\
\hline & & Flash card*1 & \(11.3 \mu \mathrm{~s}\) & \(16.8 \mu \mathrm{~s}\) & \(280.7 \mu \mathrm{~s}\) & \(287.1 \mu \mathrm{~s}\) & \(2710.7 \mu \mathrm{~s}\) & \(2717.1 \mu \mathrm{~s}\) \\
\hline & \multirow{3}{*}{BMOV} & Standard RAM & \[
4.8 \mu \mathrm{~s}
\] & \(6.6 \mu \mathrm{~s}\) & \[
114.7 \mu \mathrm{~s}
\] & \(117.1 \mu \mathrm{~s}\) & \[
1104.7 \mu \mathrm{~s}
\] & \(1107.1 \mu \mathrm{~s}\) \\
\hline & & SRAM card & \(4.8 \mu \mathrm{~s}\) & \(6.6 \mu \mathrm{~s}\) & \[
174.7 \mu \mathrm{~s}
\] & \(177.1 \mu \mathrm{~s}\) & \[
1704.7 \mu \mathrm{~s}
\] & \[
1707.1 \mu \mathrm{~s}
\] \\
\hline & & Flash card*1 & \(4.8 \mu \mathrm{~s}\) & \(6.5 \mu \mathrm{~s}\) & 274.7 us & \(277.1 \mu \mathrm{~s}\) & \(2704.7 \mu \mathrm{~s}\) & \(2707.1 \mu \mathrm{~s}\) \\
\hline \multirow[t]{6}{*}{Q04UD(E)HCPU Q06UD(E)HCPU Q10UD(E)HCPU Q13UD(E)HCPU Q20UD(E)HCPU Q26UD(E)HCPU Q50UDEHCPU Q100UDEHCPU} & \multirow{3}{*}{RBMOV} & Standard RAM & \(9.2 \mu \mathrm{~s}\) & 15.1 \% & 61.0رs & \(68.6 \mu \mathrm{~s}\) & \(531.0 \mu \mathrm{~s}\) & \(538.6 \mu \mathrm{~s}\) \\
\hline & & SRAM card & \[
9.4 \mu \mathrm{~s}
\] & \(15.6 \mu \mathrm{~s}\) & \[
165.0 \mu \mathrm{~s}
\] & \(172.6 \mu \mathrm{~s}\) & \[
1576.0 \mu \mathrm{~s}
\] & \[
1583.6 \mu \mathrm{~s}
\] \\
\hline & & Flash card* \({ }^{*}\) & \(9.4 \mu \mathrm{~s}\) & 15.7 s & \(260.0 \mu \mathrm{~s}\) & \(267.6 \mu \mathrm{~s}\) & \[
2526.0 \mu \mathrm{~s}
\] & \(2533.6 \mu \mathrm{~s}\) \\
\hline & \multirow{3}{*}{BMOV} & Standard RAM & \(4.1 \mu \mathrm{~s}\) & \(5.6 \mu \mathrm{~s}\) & \(56.0 \mu \mathrm{~s}\) & \(58.6 \mu \mathrm{~s}\) & \[
526.0 \mu \mathrm{~s}
\] & \(528.6 \mu \mathrm{~s}\) \\
\hline & & SRAM card & \(4.5 \mu \mathrm{~s}\) & \(6.1 \mu \mathrm{~s}\) & \(160.0 \mu \mathrm{~s}\) & \(162.6 \mu \mathrm{~s}\) & \[
1571.0 \mu \mathrm{~s}
\] & \(1573.6 \mu \mathrm{~s}\) \\
\hline & & Flash card*1 & \(4.3 \mu \mathrm{~s}\) & \(6.2 \mu \mathrm{~s}\) & \[
255.0 \mu \mathrm{~s}
\] & \(257.6 \mu \mathrm{~s}\) & \[
2521.0 \mu \mathrm{~s}
\] & \(2523.6 \mu \mathrm{~s}\) \\
\hline \multirow{4}{*}{Q03UDVCPU} & \multirow[t]{2}{*}{RBMOV} & Standard RAM & \(3.7 \mu \mathrm{~s}\) & 21.0رs & \(80.6 \mu \mathrm{~s}\) & \(89.3 \mu \mathrm{~s}\) & \(822.2 \mu \mathrm{~s}\) & \(831.4 \mu \mathrm{~s}\) \\
\hline & & Extended SRAM cassette & \(3.7 \mu \mathrm{~s}\) & 21.0رs & \(102.6 \mu \mathrm{~s}\) & \(118.1 \mu \mathrm{~s}\) & 1056.4 \(\mu \mathrm{s}\) & \(1072.0 \mu \mathrm{~s}\) \\
\hline & \multirow[b]{2}{*}{BMOV} & Standard RAM & \(1.9 \mu \mathrm{~s}\) & 7.9 s & \(79.5 \mu \mathrm{~s}\) & 82.0رs & \(820.6 \mu \mathrm{~s}\) & \(823.1 \mu \mathrm{~s}\) \\
\hline & & Extended SRAM cassette & \(1.9 \mu \mathrm{~s}\) & \(7.9 \mu \mathrm{~s}\) & \(102.6 \mu \mathrm{~s}\) & \(107.9 \mu \mathrm{~s}\) & \(1055.5 \mu \mathrm{~s}\) & \(1057.5 \mu \mathrm{~s}\) \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline \multirow{2}{*}{CPU} & \multirow{2}{*}{Instruction} & \multirow[t]{2}{*}{Target memory where file register is stored} & \multicolumn{2}{|c|}{1 word} & \multicolumn{2}{|r|}{1000 words} & \multicolumn{2}{|r|}{10000 words} \\
\hline & & & Min. & Max. & Min. & Max. & Min. & Max. \\
\hline \multirow{4}{*}{Q04UDVCPU Q06UDVCPU Q13UDVCPU Q26UDVCPU} & \multirow{2}{*}{RBMOV} & Standard RAM & \(3.7 \mu \mathrm{~s}\) & \(21.0 \mu \mathrm{~s}\) & 42.1 us & \(57.7 \mu \mathrm{~s}\) & \(413.0 \mu \mathrm{~s}\) & 428.6 \({ }^{\text {s }}\) \\
\hline & & Extended SRAM cassette & \(3.7 \mu \mathrm{~s}\) & 21.0رs & \(102.6 \mu \mathrm{~s}\) & 118.1 / & 1056.4 \(\mu \mathrm{s}\) & 1072.0رs \\
\hline & \multirow{2}{*}{BMOV} & Standard RAM & \(1.9 \mu \mathrm{~s}\) & \(7.9 \mu \mathrm{~s}\) & \(41.0 \mu \mathrm{~s}\) & 47.1 \% & \(411.6 \mu \mathrm{~s}\) & \(417.7 \mu \mathrm{~s}\) \\
\hline & & Extended SRAM cassette & \(1.9 \mu \mathrm{~s}\) & 7.9 s & \(102.6 \mu \mathrm{~s}\) & \(107.9 \mu \mathrm{~s}\) & 1055.4 s & 1060.9 \({ }^{\text {s }}\) \\
\hline
\end{tabular}
*1: When file registers are stored in the Flash card, no processing is performed for transfer from internal devices to file registers.
*2: Memory card cannot be used on Q00UCPU and Q01UCPU.
(2) Transferring data from file registers to file registers
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline \multirow{2}{*}{CPU} & \multirow{2}{*}{Instruction} & \multirow[t]{2}{*}{Target memory where file register is stored} & \multicolumn{2}{|c|}{1 word} & \multicolumn{2}{|l|}{1000 words} & \multicolumn{2}{|r|}{10000 words} \\
\hline & & & Min. & Max. & Min. & Max. & Min. & Max. \\
\hline \multirow{4}{*}{\begin{tabular}{l}
QnHCPU \\
QnPHCPU \\
QnPRHCPU
\end{tabular}} & \multirow{2}{*}{RBMOV} & Standard RAM & \multicolumn{2}{|c|}{20.0رs} & \multicolumn{2}{|c|}{91.0رs} & \multicolumn{2}{|c|}{\(775.0 \mu \mathrm{~s}\)} \\
\hline & & SRAM card & \multicolumn{2}{|c|}{\(22.5 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(545.0 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(5300.0 \mu \mathrm{~s}\)} \\
\hline & \multirow{2}{*}{BMOV} & Standard RAM & \multicolumn{2}{|c|}{\(7.5 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{77.0رs} & \multicolumn{2}{|c|}{720.0رs} \\
\hline & & SRAM card & \multicolumn{2}{|c|}{\(8.5 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{692.0رs} & \multicolumn{2}{|c|}{7050.0رs} \\
\hline \multirow{4}{*}{QnCPU} & \multirow{2}{*}{RBMOV} & Standard RAM & \multicolumn{2}{|c|}{\(45.5 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(215.0 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(1850.0 \mu \mathrm{~s}\)} \\
\hline & & SRAM card & \multicolumn{2}{|c|}{\(50.0 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(870.0 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(8350.0 \mu \mathrm{~s}\)} \\
\hline & \multirow{2}{*}{BMOV} & Standard RAM & \multicolumn{2}{|c|}{\(17.5 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(179.0 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(1700.0 \mu \mathrm{~s}\)} \\
\hline & & SRAM card & \multicolumn{2}{|c|}{\(18.5 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(839.0 \mu \mathrm{~s}\)} & \multicolumn{2}{|c|}{\(8600.0 \mu \mathrm{~s}\)} \\
\hline \multirow{4}{*}{Q00UCPU Q01UCPU} & \multirow{2}{*}{RBMOV} & Standard RAM & \(12.6 \mu \mathrm{~s}\) & \(35.3 \mu \mathrm{~s}\) & \(232.5 \mu \mathrm{~s}\) & 256.1 \(\mu \mathrm{s}\) & \(2211.5 \mu \mathrm{~s}\) & 2235.1 us \\
\hline & & SRAM card* \({ }^{\text {1 }}\) & - & - & - & - & - & - \\
\hline & \multirow{2}{*}{BMOV} & Standard RAM & 7.7 \(\mu \mathrm{s}\) & \(14.2 \mu \mathrm{~s}\) & \(227.5 \mu \mathrm{~s}\) & \(234.2 \mu \mathrm{~s}\) & \(2206.5 \mu \mathrm{~s}\) & 2214.2 \({ }^{\text {s }}\) \\
\hline & & SRAM card \({ }^{* 1}\) & - & - & - & - & - & - \\
\hline \multirow{4}{*}{Q02UCPU} & \multirow{2}{*}{RBMOV} & Standard RAM & \(9.6 \mu \mathrm{~s}\) & \(31.5 \mu \mathrm{~s}\) & \(228.5 \mu \mathrm{~s}\) & \(252.3 \mu \mathrm{~s}\) & \(2208.5 \mu \mathrm{~s}\) & 2231.3 \({ }^{\text {m }}\) \\
\hline & & SRAM card & \(9.6 \mu \mathrm{~s}\) & \(31.5 \mu \mathrm{~s}\) & \(378.5 \mu \mathrm{~s}\) & 401.3 \(\mu \mathrm{s}\) & \(3708.5 \mu \mathrm{~s}\) & 3731.3 \({ }^{\text {s }}\) \\
\hline & \multirow{2}{*}{BMOV} & Standard RAM & \(5.2 \mu \mathrm{~s}\) & \(11.8 \mu \mathrm{~s}\) & \(224.5 \mu \mathrm{~s}\) & \(232.3 \mu \mathrm{~s}\) & \(2204.5 \mu \mathrm{~s}\) & \(2212.3 \mu \mathrm{~s}\) \\
\hline & & SRAM card & \(5.2 \mu \mathrm{~s}\) & 11.8 \(\mu \mathrm{s}\) & \(374.5 \mu \mathrm{~s}\) & \(382.3 \mu \mathrm{~s}\) & \(3704.5 \mu \mathrm{~s}\) & 3712.3 s \\
\hline \multirow{4}{*}{Q03UD(E)CPU} & \multirow[t]{2}{*}{RBMOV} & Standard RAM & \(11.2 \mu \mathrm{~s}\) & \(16.7 \mu \mathrm{~s}\) & \(230.7 \mu \mathrm{~s}\) & \(237.1 \mu \mathrm{~s}\) & \(2210.7 \mu \mathrm{~s}\) & 2217.1 ¢ \\
\hline & & SRAM card & 11.6 \(\mu \mathrm{s}\) & 16.7 \(\mu \mathrm{s}\) & 380.7 \(\mu \mathrm{s}\) & 387.1 / & \(3710.7 \mu \mathrm{~s}\) & 3717.1 \(\mathrm{\mu}\) \\
\hline & \multirow[t]{2}{*}{BMOV} & Standard RAM & \(4.9 \mu \mathrm{~s}\) & \(6.7 \mu \mathrm{~s}\) & 224.7 / s & \(227.1 \mu \mathrm{~s}\) & 2204.7 / s & 2207.1 ¢ \\
\hline & & SRAM card & \(5.2 \mu \mathrm{~s}\) & \(6.7 \mu \mathrm{~s}\) & \(374.7 \mu \mathrm{~s}\) & 377.1 \(\mu \mathrm{s}\) & 3704.7 m & 3707.1 s \\
\hline \multirow[t]{4}{*}{Q04UD(E)HCPU Q06UD(E)HCPU Q10UD(E)HCPU Q13UD(E)HCPU Q20UD(E)HCPU Q26UD(E)HCPU Q50UDEHCPU Q100UDEHCPU} & \multirow{2}{*}{RBMOV} & Standard RAM & \(9.3 \mu \mathrm{~s}\) & \(15.5 \mu \mathrm{~s}\) & \(118.0 \mu \mathrm{~s}\) & \(124.6 \mu \mathrm{~s}\) & 1102.0رs & \(1107.6 \mu \mathrm{~s}\) \\
\hline & & SRAM card & \(9.7 \mu \mathrm{~s}\) & \(15.5 \mu \mathrm{~s}\) & \(365.0 \mu \mathrm{~s}\) & \(371.6 \mu \mathrm{~s}\) & \(3571.0 \mu \mathrm{~s}\) & 3578.6 \(\mu \mathrm{s}\) \\
\hline & \multirow{2}{*}{BMOV} & Standard RAM & \(4.3 \mu \mathrm{~s}\) & \(6.2 \mu \mathrm{~s}\) & \(113.0 \mu \mathrm{~s}\) & \(115.6 \mu \mathrm{~s}\) & 1096.0رs & 1098.6 \(\mu \mathrm{s}\) \\
\hline & & SRAM card & \(4.5 \mu \mathrm{~s}\) & 6.1 ¢ & \(360.0 \mu \mathrm{~s}\) & \(362.6 \mu \mathrm{~s}\) & 3566.0رs & 3568.6 \({ }^{\text {us }}\) \\
\hline \multirow{4}{*}{Q03UDVCPU} & \multirow{2}{*}{RBMOV} & Standard RAM & \(3.7 \mu \mathrm{~s}\) & \(20.7 \mu \mathrm{~s}\) & \(162.0 \mu \mathrm{~s}\) & \(171.2 \mu \mathrm{~s}\) & 1637.7 \(\mu \mathrm{s}\) & 1646.4 \({ }^{\text {s }}\) \\
\hline & & Extended SRAM cassette & \(3.7 \mu \mathrm{~s}\) & \(20.7 \mu \mathrm{~s}\) & \(216.7 \mu \mathrm{~s}\) & 232.1 \% & 2197.4 s & \(2212.5 \mu \mathrm{~s}\) \\
\hline & \multirow{2}{*}{BMOV} & Standard RAM & 1.9 \({ }^{\text {us }}\) & 8.0رs & \(161.1 \mu \mathrm{~s}\) & 163.7 \(\mu \mathrm{s}\) & 1636.2 \(\mu \mathrm{s}\) & 1638.8 \({ }^{\text {s }}\) \\
\hline & & Extended SRAM cassette & \(1.9 \mu \mathrm{~s}\) & \(8.3 \mu \mathrm{~s}\) & \(216.4 \mu \mathrm{~s}\) & \(221.7 \mu \mathrm{~s}\) & 2197.4 \(\mu \mathrm{s}\) & 2201.7 \(\mu \mathrm{s}\) \\
\hline \multirow[t]{4}{*}{\begin{tabular}{l}
Q04UDVCPU \\
Q06UDVCPU \\
Q13UDVCPU \\
Q26UDVCPU
\end{tabular}} & \multirow[t]{2}{*}{RBMOV} & Standard RAM & \(3.5 \mu \mathrm{~s}\) & \(20.7 \mu \mathrm{~s}\) & \(84.6 \mu \mathrm{~s}\) & \(99.5 \mu \mathrm{~s}\) & \(836.3 \mu \mathrm{~s}\) & \(851.7 \mu \mathrm{~s}\) \\
\hline & & Extended SRAM cassette & \(3.6 \mu \mathrm{~s}\) & 20.7 \(\mu \mathrm{s}\) & \(216.7 \mu \mathrm{~s}\) & \(232.1 \mu \mathrm{~s}\) & 2197.4 4 s & \(2212.5 \mu \mathrm{~s}\) \\
\hline & \multirow[b]{2}{*}{BMOV} & Standard RAM & \(1.8 \mu \mathrm{~s}\) & \(8.0 \mu \mathrm{~s}\) & 83.1 ¢ & \(89.0 \mu \mathrm{~s}\) & 835.0رs & \(840.9 \mu \mathrm{~s}\) \\
\hline & & Extended SRAM cassette & \(1.8 \mu \mathrm{~s}\) & \(8.3 \mu \mathrm{~s}\) & \(216.4 \mu \mathrm{~s}\) & \(221.7 \mu \mathrm{~s}\) & 2197.4 \(\mu \mathrm{s}\) & 2201.7 \(\mu \mathrm{s}\) \\
\hline
\end{tabular}
*1: Memory card cannot be used on Q00UCPU and Q01UCPU.

\subsection*{7.18.20 User message instruction}

UMSG

- Built-in Ethernet port LCPU: Supported
- LCPU: Not supported for L02SCPU and L02SCPU-P

\section*{UMSG}


:Bit
\(r\) :String
:Bit
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[b]{2}{*}{Setting data} & Inte & evice & \multirow[b]{2}{*}{R, ZR} & \multirow[b]{2}{*}{Indirect specification} & \multicolumn{2}{|c|}{Jalaly} & \multirow[b]{2}{*}{U...idat...} & \multirow[b]{2}{*}{Zn} & \multicolumn{2}{|c|}{Constant} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & & Bit & Word & & & K, H & Real string & \\
\hline (s) & - & & & \(\bigcirc\) & & & - & & & \(\triangle^{* 1}\) & - \\
\hline
\end{tabular}
*1 : Only for character string data.

\section*{Function}
(1) Character string data specified for (s) are displayed on the display module as a user message.
Character string data directly specified for © (specified by enclosing with " " (double quotation)), or character string data between the device number specified for (s) and the device number that stores 00 H are displayed.

(2) Up to 128 characters can be displayed on the display module.

\section*{区POINT}

The upper bytes and the lower bytes are reversed when character string data are stored to devices by using the character string data transfer instruction (\$MOV). For the \$MOV instruction, refer to Section 6.4.4.
(3) The user message is displayed at the rising edge of the UMSG instruction command. If the character string data are changed while the command is ON, the changed user message is displayed on the display module.
(4) The specified character string data for the UMSG instruction are displayed at the END processing. When the multiple UMSG instructions are executed, the UMSG instruction executed directly prior to the End processing is validated. When the multiple programs are running, the UMSG instruction executed at last is validated.
(5) No processing is performed if the instruction is executed without the display module mounted.
(6) Press "ESC" on the display module to clear the displayed message.

Execute "User Message" on the menu screen of the display module to redisplay the message.
(7) The displayed message is cleared if the NULL code \((00 \mathrm{H})\) is specified for the argument of the instruction.
The following explains the method for specifying a NULL code \((00 \mathrm{H})\) for the argument of the instruction.


For details of the display module, refer to MELSEC-L CPU Module User's Manual (Function Explanation, Program Fundamentals).

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- The NULL code \((00 \mathrm{H})\) does not exist within the specified range of the devices following the device number specified for (s).
(Error code: 4101)
- The number of characters of (s) exceeds 128 characters.
(Error code: 4100)

\section*{\(\triangle\) Program Example}
(1) In the following program, the character string data in D10 and the following devices are displayed when X10 turns ON.
[Structured ladder/FBD]

[ST]
UMSG(X10,D10);
[Operation]
\begin{tabular}{l|c:c|} 
& B15 to b8 & B7 to b0 \\
\cline { 2 - 3 } D10 & \(41_{\mathrm{H}}(\mathrm{A})\) & \(4 \mathrm{D}_{\mathrm{H}}(\mathrm{M})\) \\
\cline { 2 - 3 } D11 & \(4 \mathrm{E}_{\mathrm{H}}(\mathrm{N})\) & \(49_{\mathrm{H}}(\mathrm{I})\) \\
\cline { 2 - 3 } D12 & \(45_{\mathrm{H}}(\mathrm{E})\) & \(54_{\mathrm{H}}(\mathrm{T})\) \\
D13 & \(41_{\mathrm{H}}(\mathrm{A})\) & \(4 \mathrm{E}_{\mathrm{H}}(\mathrm{N})\) \\
\cline { 2 - 3 } D14 & \(43_{\mathrm{H}}(\mathrm{C})\) & \(4 \mathrm{E}_{\mathrm{H}}(\mathrm{N})\) \\
D15 & \(20_{\mathrm{H}}(\) space \()\) & \(45 \mathrm{H}(\mathrm{E})\) \\
\cline { 2 - 3 } D16 & \multicolumn{2}{|c|}{\(0000_{\mathrm{H}}\)} \\
\cline { 2 - 3 } &
\end{tabular}


Execution of UMSG
instruction
(2) In the following program, "A line in operation" is displayed when M0 turns ON.
[Structured ladder/FBD]

[ST]
UMSG(M0,"A line in operation");
[Operation]
\begin{tabular}{|c|c|}
\hline B15 to b 8 & B 7 to b 0 \\
\hline \(20_{\mathrm{H}}(\) space \()\) & \(41_{\mathrm{H}}(\mathrm{A})\) \\
\hline \(69_{\mathrm{H}}(\mathrm{i})\) & \(6 \mathrm{C}_{\mathrm{H}}(\mathrm{I})\) \\
\hline \(65_{\mathrm{H}}(\mathrm{e})\) & \(6 \mathrm{E}_{\mathrm{H}}(\mathrm{n})\) \\
\hline \(69_{\mathrm{H}}(\mathrm{i})\) & \(20_{\mathrm{H}}(\) space \()\) \\
\hline \(20_{\mathrm{H}}(\) space \()\) & \(6 \mathrm{E}_{\mathrm{H}}(\mathrm{n})\) \\
\hline \(70_{\mathrm{H}}(\mathrm{p})\) & \(6 \mathrm{~F}_{\mathrm{H}}(\mathrm{o})\) \\
\hline \(72_{\mathrm{H}}(\mathrm{r})\) & \(65_{\mathrm{H}}(\mathrm{e})\) \\
\hline \(74_{\mathrm{H}}(\mathrm{t})\) & \(61_{\mathrm{H}}(\mathrm{a})\) \\
\hline \(6 \mathrm{~F}_{\mathrm{H}}(\mathrm{o})\) & \(69_{\mathrm{H}}(\mathrm{i})\) \\
\hline \(20_{\mathrm{H}}(\) space \()\) & \(6 \mathrm{E}_{\mathrm{H}}(\mathrm{n})\) \\
\hline \multicolumn{2}{|c|}{\(0000_{\mathrm{H}}\)} \\
\hline
\end{tabular}


A line in operation
(3) In the following program, "B line in suspension" is displayed when X 10 turns ON , and the message is cleared when X10 turns OFF.
[Structured ladder/FBD]

[ST]
UMSG(X10, "B line in suspension");
UMSG(NOT(X10), "");

\section*{[Operation]}


MEMO

\section*{DATA LINK INSTRUCTIONS}
8.1 Network Refresh Instructions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8-2
8.2 Reading/Registering Routing Information. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8-78-7

\subsection*{8.1 Network Refresh Instructions}

\subsection*{8.1.1 Refresh instruction for specified module}

\section*{S(P)_ZCOM_J(U)}
P: Executing condition


Andicates any of the following instructions.
S_ZCOM_J S_ZCOM_U

SP_ZCOM_J SP_ZCOM_U
:Bit
EN: Executing condition
Input argument,
\[
\mathrm{Jn}^{*}: \quad \text { Network number of the host station (For QCPU (Q mode) only):ANY16 }
\]
Un*: Start I/O number of the network module of the host station :ANY16
( 00 to FE: Higher two digits when expressing the I/O number in
three digits)
Output argument, ENO: Execution result
:Bit
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..ala} & \multirow[b]{2}{*}{} & \multirow{2}{*}{Zn} & \multirow{2}{*}{Constant} & \multirow{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline - & \multicolumn{9}{|c|}{-} \\
\hline
\end{tabular}

Function
The ZCOM instruction is used to perform refresh at any timing during execution of a sequence program.

The targets of refresh performed by the ZCOM instruction are indicated below.
- Refresh of CC-Link IE Controller Network (when refresh parameters are set) (For QCPU (Q mode) only)
- Refresh of CC-Link IE Field Network (when refresh parameters are set)
(For Universal model QCPU with a serial number whose first five digits are '12012' or higher and LCPU with a serial number whose first five digits are '13012' or higher only)
- Refresh of MELSECNET/H (when refresh parameters are set) (For QCPU (Q mode) only)
- Auto refresh of CC-Link (when refresh device is set)
- Auto refresh of intelligent function module (when auto refresh is set)
(1) When the ZCOM instruction is executed, the CPU module temporarily suspends processing of the sequence program and conducts refresh processing of the network modules specified by Jn/Un. (For LCPU with a serial number whose first five digits are '13011' or lower, a module cannot be specified by Jn.)

(2) The ZCOM instruction does not perform the following processing.
(a) Communication processing between CPU module and programming tool
(b) Monitor processing of other station
(c) Read processing of buffer memory of other intelligent function module by serial communication module
(d) Low-speed cyclic data transmission of MELSECNET/H
(3) CC-Link IE Controller Network and MELSECNET/H (PLC to PLC network)
(a) When the scan time for the sequence program of host station is longer than the scan time for the other station, the ZCOM instruction is used to ensure the data reception from the other station.
1) Example of data communications when the ZCOM instruction is not used

2) Example of data communications when the ZCOM instruction is used


For details of the transmission delay time on the CC-Link IE Controller Network and MELSECNET/H (PLC to PLC network), refer to the following manuals.
- CC-Link IE Controller Network Reference Manual
- Q Corresponding MELSECNET/H Network System Reference Manual (PLC to PLC network)
(b) When the link scan time is longer than the sequence program scan time, data communications will not be faster even if the ZCOM instruction is used.

(4) MELSECNET/H (remote I/O network)

The link refresh of the remote master station is performed by the "END processing" of the CPU module.
Since link scan is performed at completion of link refresh, link scan 'synchronizes' with the program of the CPU module.
When the ZCOM instruction is used at the remote master station, link refresh is performed at the point of ZCOM instruction execution, and link scan is performed at completion of link refresh.
Hence, use of the ZCOM instruction at the remote master station speeds up send/receive processing to/from the remote I/O station.
1) When the ZCOM instruction is not used

2) When the ZCOM instruction is used


For details of the transmission delay time on the MELSECNET/H (remote I/O network), refer to the following manual.
- Q Corresponding MELSECNET/H Network System Reference Manual (Remote I/O network)
(5) The ZCOM instruction can be used as many times as desired in sequence programs. However, note that each execution of a refresh operation will lengthen the sequence program scan time by the amount of time required for the refresh operation.
(6) Designating "Un" in the argument enables the target designation of the intelligent function as well as the network modules.
In this case, the auto refresh is performed for the buffer memory of the intelligent function modules. (It replaces the FROM/TO instructions.)
(7) For Universal model QCPU and LCPU, an interrupt is permitted during the execution of the ZCOM instruction. However, note that the data may be separated if the refresh data are used in an interrupt program.

\section*{XPOINT}
1. The ZCOM instruction cannot be used in a fixed cycle execution type program or interrupt program.
2. For Redundant CPU, there are restrictions on the use of ZCOM instruction. For details on the restrictions, refer to the following manual.
- QnPRHCPU User's Manual (Redundant System)

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- When the module specified with the start I/O number is not a network module or intelligent function module. (For Basic model QCPU, High Performance model QCPU, Process CPU, and Redundant CPU)
(Error code: 2111)
- When the specified network number is not connected to the host station. (For LCPU, first 5 digits of the serial number are "13012" or higher)
(Error code: 4102)
- When the module specified with the start I/O number is not a network module or intelligent function module. (For Universal model QCPU and LCPU)
(Error code: 4102)

\section*{XPOINT}

To conduct only service processing, use the COM instruction. (Refer to Section 7.6.5, 7.6.6)

\section*{\(\square\) Program Example}
(1) In the following program, a link refresh is executed on the network module of network No. 6 while XO is ON .
[Structured ladder/FBD]


\section*{[ST]}

S_ZCOM_J(X0,6);
(2) In the following program, a link refresh is executed on the network module mounted to the position whose start I/O number is \(\mathrm{X} / \mathrm{Y} 30\) to \(\mathrm{X} / \mathrm{Y} 4 \mathrm{~F}\) while X 0 is ON .
[Structured ladder/FBD]

[ST]
S_ZCOM_U(X0,3);

\subsection*{8.2 Reading/Registering Routing Information}
8.2.1 Reading routing information

\section*{RTREAD}
- LCPU: Supported if first 5 digits of the serial number are "13012" or later

\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Input argument,} & EN: & \multicolumn{5}{|l|}{Executing condition} & \multicolumn{4}{|l|}{} \\
\hline & n: & \multicolumn{5}{|l|}{Transfer destination network number (1 to 239)} & \multicolumn{4}{|l|}{:ANY16} \\
\hline \multirow[t]{5}{*}{Output argument,} & \[
\begin{aligned}
& \text { ENO: } \\
& \text { d: }
\end{aligned}
\] & \multicolumn{5}{|l|}{\begin{tabular}{l}
Execution result \\
Start number or the array of the device that stores the read data
\end{tabular}} & \multicolumn{4}{|l|}{\begin{tabular}{l}
:Bit \\
:Array of ANY16 (0..2)
\end{tabular}} \\
\hline & Setting & Inter & vice & R, ZR & & & O & Zn & Constant & Others \\
\hline & data & Bit & Word & & Bit & Word & U...IG... & & K, H & \\
\hline & n & \(\bigcirc\) & & & & & - & & \(\bigcirc\) & - \\
\hline & (d) & - & & & & & - & & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Reads data from transfer destination network number specified for \(n\), using routing information set by the routing parameters, and stores it to (d) and the following devices.
(2) If no data for the transfer destination network number specified for n is set at the routing parameters, stores 0 to (d) and the following devices.
(3) The content of the data stored in the area starting from (d) is as indicated below.
(Data ranges)
\begin{tabular}{l|l|l} 
& \multicolumn{2}{|c}{ (d) \([0]\)} \\
& Relay network number & (1 to 239\()\) \\
(d) \([1]\) & Relay station number & Refer to the following table \\
& &
\end{tabular}
[Relay station number]
\begin{tabular}{l|l}
\multicolumn{1}{c|}{ Network Type } & \multicolumn{1}{c}{ Specified range } \\
\hline MELSECNET/H & 1 to 64 \\
\hline CC-Link IE Controller Network & 1 to 120 \\
\hline CC-Link IE Field Network & \begin{tabular}{l} 
Master Station: 125 Fixed (Fixed value is stored) \\
Local station: 1 to 120 (Station number is stored)
\end{tabular} \\
\hline
\end{tabular}

\section*{OOperation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
- When the device which cannot be used the argument is specified (Error code: 4004)
- When data specified for n is other than 1 to 239. (For High Performance model QCPU, Process CPU, Redundant CPU, and Universal model QCPU)
(Error code: 4100)
- The device specified for \(n\) and (d) exceeds the range of the corresponding device. (For Universal model QCPU and LCPU)
(Error code: 4101)

\section*{\(\triangle\) Program Example}
(1) In the following program, the routing information of the network number specified for D0 is read when X0 turns ON.
[Structured ladder/FBD]


\section*{[ST]}

S_RTREAD(X0,D0,D1);
[Operation]
[Content of routing parameter setting]
D0
D1
D1
D1
D2
D3
Dummy

\subsection*{8.2.2 Registering routing information}
- LCPU: Supported if first 5 digits of the serial number are " 13012 " or later

\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{小)} & \multirow[b]{2}{*}{U.fic:} & \multirow[b]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline n & \(\bigcirc\) & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (5) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Registers routing data of © ( and the following devices in the area for the transfer destination network number specified for n in the routing parameters.
\begin{tabular}{l|l}
\multicolumn{1}{c|}{ CPU module } & Registerable data \\
\begin{tabular}{l} 
The High Performance model QCPU, Process CPU, Redundant CPU, Universal \\
model QCPU whose serial number (first five digits) is "14111" or earlier, High-speed \\
Universal model QCPU whose serial number (first five digits) is "15041" or earlier, and \\
LCPU
\end{tabular} & Up to 64 \\
\hline • The Universal model QCPU whose serial number (first five digits) is "14112" & \\
\begin{tabular}{l} 
or later, (except the High-speed Universal model QCPU) \\
- High-speed Universal model QCPU whose serial number (first five digits) is \\
"15042" or later
\end{tabular} & Up to 238 \\
\hline
\end{tabular}
(2) The following shows the content of data to be set for (s) and the following devices.
(Data ranges)
\begin{tabular}{|c|c|c|}
\hline & \multicolumn{2}{|r|}{(Data ranges)} \\
\hline (s) [0] & Relay network number & (1 to 239) \\
\hline (s) [1] & Relay station number & Refer to the following table. \\
\hline (s) [2] & Dummy & \\
\hline
\end{tabular}
[Relay station number]
\begin{tabular}{l|l}
\multicolumn{1}{c|}{ Network Type } & \multicolumn{1}{c}{ Specified range } \\
\hline MELSECNET/H & 1 to 64 \\
\hline CC-Link IE Controller Network & 1 to 120 \\
\hline CC-Link IE Field Network & \begin{tabular}{l} 
Master Station: 125 Fixed (Fixed value is stored) \\
Local station: 1 to 120 (Station number is stored)
\end{tabular} \\
\hline
\end{tabular}
(3) If data of the transfer destination network number specified for \(n\) are set in the routing parameters, they are changed to the data in (5) and the following devices.
(4) When data in both of © +0 and (s) +1 are 0 , the data of the transfer destination network number specified for n is deleted from the routing parameters.

\section*{0 Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
- When the device which cannot be used the argument is specified (Error code: 4004) [High Performance model QCPU, Process CPU, Redundant CPU, and Universal model QCPU]
- When data specified for n is other than 1 to 239.
(Error code: 4100)
- When the data of ©s and the following devices exceed each setting ranges.
(Error code: 4100)
- When the total number of routing data registered in the routing parameter of the network parameters and routing data registered with the RTWRITE instruction exceeds the maximum number of registerable data.
(Error code: 4100)
- When the transfer destination network number which is not registered in the routing parameter is attempted to delete.
(Error code: 4100)
[Universal model QCPU and LCPU]
- The device specified for n and (d) exceeds the range of the corresponding device.
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, the routing information specified for D1 to D3 is written to the network module whose network number is specified for DO when X0 turns ON.
[Structured ladder/FBD]

[ST]
S_RTWRITE(X0,D0,D1);
[Operation] [Content of routing parameter setting]


\subsection*{8.3 Refresh Device Write/Read Instruction}

\subsection*{8.3.1 Refresh device write (in 1-bit units)}
- QnUD(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "14072" or later
- Built-in Ethernet port LCPU: Supported
- Not supported for Q00UJCPU, Q00UCPU, Q01UCPU, Q02UCPU, QnUDVCPU, L02SCPU, and L02SCPU-P

S(P)_REFDVWRB

P: Executing condition
indicates any of the following instructions.

S_REFDVWRB SP_REFDVWRB

Bit
ANY16
:Array of ANY16 (0..3)
:Bit
:ANY32
:Bit

Start number or the array of the bit device which turns on for 1 :Array of Bit (0..1)
scan by the instruction completion. (d1) +1 also turns on at the
error completion (bit).
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{\(\mathrm{R}, \mathrm{ZR}\)} & \multicolumn{2}{|c|}{} & \multirow[b]{2}{*}{U} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{\[
\begin{gathered}
\text { Constant } \\
\text { K, H }
\end{gathered}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline n1 & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & \(\bigcirc\) & - \\
\hline (51) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & - & - \\
\hline (s2) & \(\Delta^{*}{ }^{2}\) & \multicolumn{2}{|c|}{-} & & & \multicolumn{2}{|l|}{-} & - & - \\
\hline n2 & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & \(\bigcirc\) & - \\
\hline (d1) & \(\triangle^{* 2}\) & \multicolumn{2}{|c|}{-} & & & \multicolumn{2}{|l|}{-} & - & - \\
\hline
\end{tabular}
*1: The first 3 digits of the hexadecimal 4 digits which represent the start I/O number.
*2: Local devices and the devices specified for individual programs cannot be used.
(1) The contents of the data stored in the area starting from ©51 are as indicated below
\begin{tabular}{|c|c|c|c|}
\hline Device & Item & Setting contents & Setting range \\
\hline (s1) +0 & Station number & Station number for the station assigned the refresh device which writes data. When the link special relay (SB) is specified as type of (s1) +1 , the setting is disabled. & 1 to 120 \\
\hline (51) +1 & Type & \begin{tabular}{l}
Type of the refresh device which writes data \\
1: Remote input (RX) \\
2: Remote output (RY) \\
3: Link special relay (SB)
\end{tabular} & 1 to 3 \\
\hline \[
\begin{aligned}
& \text { (51) }+2 \\
& \text { (s1) }+3
\end{aligned}
\] & Offset & Offset from the head of the refresh device assigned the device specified in (s1) +0 and (s1) +1 & 0 to 2147483647 \\
\hline
\end{tabular}
(2) Instruction execution possibility of an execution type for each program.
(a) Enabled: Initial program and scan execution type program
(b) Disabled: Fixed scan execution type program and interrupt program
(3) To write data to a refresh device, the data reflection to the station number specified by the instruction is executed at the timing for the auto refresh.
(4) Number of points specified in n 2 is written from the device specified in \({ }^{2} 2\) to the offset specified in (51)+2 of the refresh device assigned for the device specified in (51)+1 of the target station specified in n 1 and \((11)+0\).
[Structure]


Refresh device areas assigned for station No. 4
At the above configuration, number of points specified in n 2 is written from the device specified in (52) to the offset (Y1078) specified in (51)+2 of the device assigned for the station number 4.


\section*{区POINT}

When a refresh range per station is assigned in transfer settings, specify number of write points so that the range written data from the specified offset is within the range assigned in the same transfer setting. An error occurs if the number of write points over the range assigned in each transfer setting is specified.
(5) The station type which can and cannot specify with the start I/O number is as follows.
\begin{tabular}{l|l}
\multicolumn{1}{c|}{\begin{tabular}{c} 
Specification \\
possibility
\end{tabular}} & \multicolumn{1}{c}{ Station type } \\
Enabled & \begin{tabular}{l} 
CC-Link master station, CC-Link master station (compatible with redundant \\
function), CC-Link IE Field Network master station
\end{tabular} \\
\hline Disabled & \begin{tabular}{l} 
CC-Link local station, CC-Link standby master station, CC-Link IE Field Network \\
local station, CC-Link IE Field Network submaster station
\end{tabular} \\
\hline
\end{tabular}
(6) Because the available range of the station number is 1 to 120 , the station number for the master station in n 1 cannot be specified to \(\left(\mathrm{sil}^{2}+0\right.\). If the station number is specified, the "OPERATION ERROR" (error code: 4102) occurs.
(7) SM739 (Refresh device write/read instruction in execution flag) turns on during the instruction execution. When SM739 is on, the following instructions cannot be executed.
- S(P)_REFDVWRB
- S(P)_REFDVWRW
- S(P)_REFDVRDB
- S(P)_REFDVRDW

If these instructions are executed, no processing is performed. When an error is detected at the instruction execution (before SM739 turns on), the completion device ( ( \(11+0\) ), the completion device ( \((11+1)\), and SM739 do not turn on.
(8) The instruction completion can be checked in the completion device ( \((11+0\) and (d1) +1 ).
(a) Completion device \(\left(@_{1}+0\right)\)

The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.
(b) Completion device \(\left({ }^{(d 1)}+1\right)\)

The device turns on or off by the status when the instruction is completed.
- Normal completion: No change from off
- Error completion: The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.

(9) A module set parameters by the dedicated instruction and a CC-Link module operating by the automatic CC-Link startup cannot be specified with the instruction.
(10) The write source (points in \(n 2\) from \({ }_{(22}\) ) and write destination (points in \(n 2\) from a device specified in control data) are overlapped, data can be written. Write data starting from (s2) when data are written to the smaller device number. Write data starting from \(\Omega_{2}+((\mathrm{n} 2)-1)\) when data are written to the larger device number.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) turns on, and an error code is stored into SD0.
- When the instruction is used with the CPU module whose serial number (first five digits) is "14071" or earlier
(Error code: 4002)
- When the module cannot be specified the start I/O number in n1
(Error code: 4002)
- When the device which cannot be specified is specified
(Error code: 4004)
- When the specified device exceeds the range of the number of device points
(Error code: 4101)
- When the start I/O number in n 1 is out of the specified range
(Error code: 4101)
- When the device type number in \((51+1\) is out of the specified range
(Error code: 4101)
- When the write offset in \(\$ 1+2\) is out of the specified range
(Error code: 4101)
- When the number of write points in n 2 is out of the specified range
(Error code: 4101)
- When the number of write points in n 2 exceeds the range of the number of device points
(Error code: 4101)
- When the station number in \((51+0\) is out of the specified range
(Error code: 4102)
- When the station number in © \(51+0\) does not exist
(Error code: 4102)
- When the station number in ⑪ \(^{2} 0\) is the master station specified in \(n 1\) (Error code: 4102)
- When the start I/O number in n 1 is the station type which cannot be specified
(Error code: 4150)
- When the start I/O number in n 1 does not exist in the network parameter
(Error code: 4150)
- When the device in (51)+1 of the station number specified in (51)+0 is not assigned the refresh device
(Error code: 4151)
- When the write offset in (51)+2 exceeds the refresh device range assigned for the device in (51)+1 of the station number specified in (51)+0
(Error code: 4151)
- When the number of write points in n2 exceeds the assignment range of the setting for one transfer from the write offset in (51)+2
(Error code: 4151)

\section*{\(\square\) Program Example}

The following program writes 16 device values in B100 to the head of the refresh device (offset: 0) assigned for the remote output (RY) in the remote I/O station on the station number 32 controlled by the CC-Link master station of the start I/O number \(0080_{\mathrm{H}}\) when X1C is turned on.
[Structured ladder/FBD]

```

[ST]
MOV( LDP( TRUE , X1C ), K32, D100 );
MOV( LDP( TRUE , X1C ), K2, D101 );
DMOV( LDP( TRUE , X1C ), H0 , D102 );
SP_REFDVWRB( LDP( TRUE , X1C ), H8 , D100, B100, K16, M100 );
IF M100=TRUE THEN
IF M101=FALSE THEN
SET( TRUE , M150 );
ELSE
SET( TRUE , M151 );
END_IF;
END_IF;

```

\section*{Caution}
(1) Do not execute the instruction in an interrupt program. If the instruction is executed, no processing is performed. In addition, the completion device (@1)+0), the completion device \(\left({ }^{(d 1)}+1\right)\), and SM739 do not turn on. If the instruction is executed in a fixed scan execution type program, they also do not turn on.
(2) When the instruction is executed, do not rewrite the device data in \(\AA_{2}\) until the completion device turns on.

\subsection*{8.3.2 Refresh device write (in 16-bit units)}
- QnUD(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "14072" or later
- Built-in Ethernet port LCPU: Supported
- Not supported for Q00UJCPU, Q00UCPU, Q01UCPU,

Q02UCPU, QnUDVCPU, L02SCPU, and L02SCPU-P

instructions.
S_REFDVWRW

Input argument,
n1: \(\quad\) Start I/O number ( \(0_{\mathrm{H}}\) to \(\mathrm{FE}_{\mathrm{H}}\) ) (BIN 16-bit) \()^{* 1}\) of the master
ANY16
station controlling the station assigned the refresh device which writes data
s1: Start number or the array of the device stored control data :Array of ANY16 (0..3)
(device name)
s2: \(\quad\) Start number of the device stored write data to the refresh
ANY16
device assigned the device specified in (51 +0 and (51) +1
(device name)
n2: Number of write points (1 to 2147483647) :ANY32
Output argument,
d1: Start number or the array of the bit device which turns on for 1 :Array of Bit (0..1)
scan by the instruction completion. (d1) +1 also turns on at the
error completion (bit).
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[t]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..nlan} & \multirow[b]{2}{*}{U} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline n1 & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (51) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline (s2) & - & \multicolumn{2}{|c|}{\(\Delta^{*}\)} & \multicolumn{4}{|c|}{-} & - & - \\
\hline n2 & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d1) & \(\Delta^{*}{ }^{2}\) & \multicolumn{2}{|c|}{-} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\footnotetext{
*1: The first 3 digits of the hexadecimal 4 digits which represent the start I/O number.
*2: Local devices and the devices designated for individual programs cannot be used.
}
(1) The contents of the data stored in the area starting from © 51 are as indicated below.
\begin{tabular}{|c|c|c|c|}
\hline Device & Item & Setting contents & Setting range \\
\hline (51) +0 & Station number & Station number for the station assigned the refresh device which writes data. When the link special register (SW) is specified as type of \((\mathrm{s} 1)+1\), the setting is disabled. & 1 to 120 \\
\hline (51) +1 & Type & \begin{tabular}{l}
Type of the refresh device which writes data \\
1: Remote register (RWr) \\
2: Remote register ( RWw ) \\
3: Link special register (SW)
\end{tabular} & 1 to 3 \\
\hline \[
\begin{aligned}
& \text { (s1) }+2 \\
& \text { (51) }+3
\end{aligned}
\] & Offset & Offset from the head of the refresh device assigned the device specified in (51 +0 and \((51)+1\) & 0 to 2147483647 \\
\hline
\end{tabular}
(2) Instruction execution possibility of an execution type for each program.
(a) Enabled: Initial program and scan execution type program
(b) Disabled: Fixed scan execution type program and interrupt program
(3) To write data to a refresh device, the data reflection to the station number specified by the instruction is executed at the timing for the auto refresh.
(4) Number of points specified in n 2 is written from the device specified in ©2) to the offset specified in \((51)+2\) of the refresh device assigned for the device specified in \((51)+1\) of the target station specified in n 1 and (51)+0.

\section*{[Structure]}


Refresh device areas assigned for station No. 4
At the above configuration, number of points specified in n 2 is written from the device specified in (32) to the offset (W1063) specified in (51) +2 of the device assigned for the station number 4.

Start number of the device
where target data is stored:


\section*{POINT}

When a refresh range per station is assigned in transfer settings, specify number of write points so that the range written data from the specified offset is within the range assigned in the same transfer setting. An error occurs if the number of write points over the range assigned in each transfer setting is specified.
(5) The station type which can and cannot specify with the start I/O number is as follows.
\begin{tabular}{l|l}
\multicolumn{1}{c|}{\begin{tabular}{c} 
Specification \\
possibility
\end{tabular}} & \multicolumn{1}{c}{ Station type } \\
\hline Enabled & \begin{tabular}{l} 
CC-Link master station, CC-Link master station (compatible with redundant function), \\
CC-Link IE Field Network master station
\end{tabular} \\
\hline Disabled & \begin{tabular}{l} 
CC-Link local station, CC-Link standby master station, CC-Link IE Field Network local \\
station, CC-Link IE Field Network submaster station
\end{tabular} \\
\hline
\end{tabular}
(6) Because the available range of the station number is 1 to 120 , the station number for the master station in n 1 cannot be specified to (s1) +0 . If the station number is specified, the "OPERATION ERROR" (error code: 4102) occurs.
(7) SM739 (Refresh device write/read instruction in execution flag) turns on during the instruction execution. When SM739 is on, the following instructions cannot be executed.
- S(P)_REFDVWRB
- S(P)_REFDVWRW
- S(P)_REFDVRDB
- S(P)_REFDVRDW

If these instructions are executed, no processing is performed. When an error is detected at the instruction execution (before SM739 turns on), the completion device (@1)+0), the completion device ( \({ }^{(11)+1) \text {, and SM739 do not turn on. }}\)
(8) The instruction completion can be checked in the completion device ( (d1) +0 and (d1) +1 ).
(a) Completion device \(\left(@_{11}+0\right)\)

The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.
(b) Completion device \(\left({ }^{(d 1)}+1\right)\)

The device turns on or off by the status when the instruction is completed.
- Normal completion: No change from off
- Error completion: The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.

(9) A module set parameters by the dedicated instruction and a CC-Link module operating by the automatic CC-Link startup cannot be specified with the instruction.
(10) The write source (points in n 2 from (®2) and write destination (points in n 2 from a device specified in control data) are overlapped, data can be written. Write data starting from \(\Omega_{2}\) when data are written to the smaller device number. Write data starting from ©2)+((n2)-1) when data are written to the larger device number.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) turns on, and an error code is stored into SD0.
- When the instruction is used with the CPU module whose serial number (first five digits) is "14071" or earlier
(Error code: 4002)
- When the module cannot be specified the start I/O number in n 1
(Error code: 4002)
- When the device which cannot be specified is specified
(Error code: 4004)
- When the specified device exceeds the range of the number of device points
(Error code: 4101)
- When the start I/O number in n 1 is out of the specified range
(Error code: 4101)
- When the device type number in (51)+1 is out of the specified range
(Error code: 4101)
- When the write offset in \({ }^{(51)+2}\) is out of the specified range
(Error code: 4101)
- When the number of write points in n 2 is out of the specified range
(Error code: 4101)
- When the number of write points in n2 exceeds the range of the number of device points
(Error code: 4101)
- When the station number in \((51)+0\) is out of the specified range
(Error code: 4102)
- When the station number in (51)+0 does not exist
(Error code: 4102)
- When the station number in ©11 +0 is the master station specified in n1 (Error code: 4102)
- When the start I/O number in n 1 is the station type which cannot be specified
(Error code: 4150)
- When the start I/O number in n1 does not exist in the network parameter
(Error code: 4150)
－When the device in（51）＋1 of the station number specified in（51）＋0 is not assigned the refresh device
－When the write offset in（51）＋2 exceeds the refresh device range assigned for the device in（51）＋1 of the station number specified in（51）＋0
（Error code：4151）
－When the number of write points in n2 exceeds the assignment range of the setting for one transfer from the write offset in（51）＋2
（Error code：4151）

\section*{\(\triangle\) Program Example}

The following program writes 16 device values in W100 to the head of the refresh device（offset： 0 ）assigned for the remote register（RWw）in the remote device station on the station number 32 controlled by the CC－Link master station of the start I／O number \(0080_{\mathrm{H}}\) when X1C is turned on．
［Structured ladder／FBD］

［ST］
MOV（ LDP（ TRUE ，X1C ），K32，D100 ）；
MOV（ LDP（ TRUE ，X1C ），K2 ，D101 ）；
DMOV（ LDP（ TRUE ，X1C ），H0 ，D102 ）；
S＿REFDVWRW（ LDP（ TRUE ，X1C ），H8 ，D100 ，W100 ，K16，M100 ）；
IF M100＝TRUE THEN
IF M101＝FALSE THEN
SET（ TRUE ，M150 ）；
ELSE
SET（ TRUE ，M151 ）；
END＿IF；
END＿IF；

\section*{Caution}
(1) Do not execute the instruction in an interrupt program. If the instruction is executed, no processing is performed. In addition, the completion device ( \((11+0)\), the completion device (d1) +1 ), and SM739 do not turn on. If the instruction is executed in a fixed scan execution type program, they also do not turn on.
(2) When the instruction is executed, do not rewrite the device data in \(\Im_{2}\) until the completion device turns on.
(3) Specifying digit for the bit device can be used only when the following conditions (a) and (b) are met.
(a) Digit specification: K4
(b) Head of device: multiple of 16

When the above conditions (a) and (b) are not met, INSTRCT CODE ERR. (error code: 4004) will occur.

\subsection*{8.3.3 Refresh device read (in 1-bit units)}

REFDVRDB
- QnUD(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "14072" or later
- Built-in Ethernet port LCPU: Supported
- Not supported for Q00UJCPU, Q00UCPU, Q01UCPU, Q02UCPU, QnUDVCPU, L02SCPU, and L02SCPU-P

S(P)_REFDVRDB

indicates any of the following

instructions.
S_REFDVRDB SP_REFDVRDB
\begin{tabular}{|c|c|c|c|}
\hline Input argument, & \[
\begin{aligned}
& \mathrm{EN} \text { : } \\
& \mathrm{n} 1:
\end{aligned}
\] & \begin{tabular}{l}
Execution condition \\
Start I/O number ( \(0_{\mathrm{H}}\) to \(\mathrm{FE}_{\mathrm{H}}\) ) (BIN 16 -bit) \()^{* 1}\) of the master station controlling the station assigned the refresh device which reads data
\end{tabular} & :Bit :ANY16 \\
\hline & s1: & Start number or the array of the device stored control data (device name) & :Array of ANY16 (0..3) \\
\hline & d1: & Start number of the device stored read data to the refresh device assigned the device specified in +0 and (s1) +1 (device name) & :Bit \\
\hline & n2: & Number of read points (1 to 2147483647) & :ANY32 \\
\hline Output argument, & ENO: & Execution result & :Bit \\
\hline & d2: & Start number or the array of the bit device which turns on for 1 scan by the instruction completion. (d1) +1 also turns on at the error completion (bit) & :Array of Bit (0..1) \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...)} & \multirow[b]{2}{*}{U} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K, H} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline n1 & - & & & & & - & & \(\bigcirc\) & - \\
\hline (s1) & - & & & & & - & & - & - \\
\hline (d1) & \(\Delta^{* 2}\) & & & & & - & & - & - \\
\hline n2 & - & & & & & - & & \(\bigcirc\) & - \\
\hline (d2) & \(\triangle^{* 2}\) & & & & & - & & - & - \\
\hline
\end{tabular}
*1: The first 3 digits of the hexadecimal 4 digits which represent the start I/O number.
*2: Local devices and the devices designated for individual programs cannot be used.
(1) The contents of the data stored in the area starting from (51) are as indicated below.
\begin{tabular}{|c|c|c|c|}
\hline Device & Item & Setting contents & Setting range \\
\hline (51) +0 & Station number & Station number for the station assigned the refresh device which reads data. When the link special relay (SB) is specified as type of (s1) +1 , the setting is disabled. & 1 to 120 \\
\hline (51) +1 & Type & \begin{tabular}{l}
Type of the refresh device which reads data \\
1: Remote input (RX) \\
2: Remote output (RY) \\
3: Link special relay (SB)
\end{tabular} & 1 to 3 \\
\hline \begin{tabular}{l}
(51) +2 \\
(s1) +3
\end{tabular} & Offset & Offset from the head of the refresh device assigned the device specified in (s1) +0 and (s1) +1 & 0 to 2147483647 \\
\hline
\end{tabular}
(2) Instruction execution possibility of an execution type for each program.
(a) Enabled: Initial program and scan execution type program
(b) Disabled: Fixed scan execution type program and interrupt program
(3) To read data to a refresh device, the data reflection to the station number specified by the instruction is executed at the timing for the auto refresh.
(4) Number of points specified in n 2 is read from the device specified in ©2 to the offset specified in (51) +2 of the refresh device assigned for the device specified in (51)+1 of the target station specified in n 1 and (s1)+0.


Refresh device areas assigned for station No. 4
At the above configuration, number of points specified in n 2 is read starting from the offset (X1078) specified in ©11+2 of the device assigned for the station number 4, then the number of points is read to devices starting from the device specified in (d1).

Start number of the device
where target data is stored:


\section*{XPOINT}

When a refresh range per station is assigned in transfer settings, specify number of read points so that the range read data from the specified offset is within the range assigned in the same transfer setting. An error occurs if the number of read points over the range assigned in each transfer setting is specified.
(5) The station type which can and cannot specify with the start I/O number is as follows.
\begin{tabular}{l|l}
\multicolumn{1}{c|}{\begin{tabular}{c} 
Specification \\
possibility
\end{tabular}} & \multicolumn{1}{c}{ Station type } \\
\hline Enabled & \begin{tabular}{l} 
CC-Link master station, CC-Link master station (compatible with redundant function), CC-Link \\
IE Field Network master station
\end{tabular} \\
\hline Disabled & \begin{tabular}{l} 
CC-Link local station, CC-Link standby master station, CC-Link IE Field Network local station, \\
CC-Link IE Field Network submaster station
\end{tabular} \\
\hline
\end{tabular}
(6) Because the available range of the station number is 1 to 120 , the station number for the master station in n 1 cannot be specified to \((91+0\). If the station number is specified, the "OPERATION ERROR" (error code: 4102) occurs.
(7) SM739 (Refresh device write/read instruction in execution flag) turns on during the instruction execution. When SM739 is on, the following instructions cannot be executed.
- S(P)_REFDVWRB
- S(P)_REFDVWRW
- S(P)_REFDVRDB
- S(P)_REFDVRDW

If these instructions are executed, no processing is performed. When an error is detected at the instruction execution (before SM739 turns on), the completion device ( (d2)+0), the completion device ( (d2) +1 ), and SM739 do not turn on.
(8) The instruction completion can be checked in the completion device (d2) +0 and (d2) +1 ).
(a) Completion device (ब2)+0)

The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.
(b) Completion device (ब2) +1 )

The device turns on or off by the status when the instruction is completed.
- Normal completion: No change from off
- Error completion: The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.

(9) A module set parameters by the dedicated instruction and a CC-Link module operating by the automatic CC-Link startup cannot be specified with the instruction.
(10) The read source (points in n 2 from a device specified in control data) and read destination (points in n 2 from (d1)) are overlapped, data can be read. Read data starting from a device specified in control data when data are read to the smaller device number. Read data starting from a device specified in control data (d1)+((n2)-1) when data are read to the larger device number.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) turns on, and an error code is stored into SD0.
- When the instruction is used with the CPU module whose serial number (first five digits) is "14071" or earlier
(Error code: 4002)
- When the module cannot be specified the start I/O number in n1
(Error code: 4002)
- When the device which cannot be specified is specified
(Error code: 4004)
- When the specified device exceeds the range of the number of device points
(Error code: 4101)
- When the start I/O number in n 1 is out of the specified range
(Error code: 4101)
- When the device type number in \((11)+1\) is out of the specified range
(Error code: 4101)
- When the read offset in ©1)+2 is out of the specified range
(Error code: 4101)
- When the number of read points in n 2 is out of the specified range
(Error code: 4101)
- When the number of read points in \(n 2\) exceeds the range of the number of device points
(Error code: 4101)
- When the station number in (51) +0 is out of the specified range
(Error code: 4102)
- When the station number in (51)+0 does not exist
(Error code: 4102)
- When the station number in \(\mathrm{S1}^{1}+0\) is the master station specified in n 1 (Error code: 4102)
- When the start I/O number in n 1 is the station type which cannot be specified
(Error code: 4150)
- When the start I/O number in n1 does not exist in the network parameter
(Error code: 4150)
- When the device in (51)+1 of the station number specified in ©(1) +0 is not assigned the refresh device
(Error code: 4151)
- When the read offset in (51)+2 exceeds the refresh device range assigned for the device in (51) +1 of the station number specified in (s1)+0
(Error code: 4151)
- When the number of read points in n 2 exceeds the assignment range of the setting for one transfer from the read offset in (51)+2
(Error code: 4151)

\section*{\(\triangle\) Program Example}

The following program reads 16 device values from the head of the refresh device (offset: 0 ) assigned for the remote input (RX) in the remote I/O station on the station number 32 controlled by the CC-Link master station of the start I/O number \(0080_{\mathrm{H}}\) to B 100 when X 1 C is turned on.
[Structured ladder/FBD]

[ST]
MOV( LDP( TRUE , X1C ) , K32 , D100 );
MOV( LDP( TRUE , X1C ) , K1, D101 );
DMOV( LDP( TRUE , X1C ) , H0 , D102 );
S_REFDVRDB( LDP( TRUE , X1C ) , H8, D100, B100, K16, M100 );
IF M100=TRUE THEN
IF M101=FALSE THEN
SET( TRUE , M150 );
ELSE
SET( TRUE , M151 );
END_IF;
END_IF;

\section*{Caution}

Do not execute the instruction in an interrupt program. If the instruction is executed, no processing is performed. In addition, the completion device ( (d2) +0 ), the completion device ( (d2 +1 ), and SM739 do not turn on. If the instruction is executed in a fixed scan execution type program, they also do not turn on.

\subsection*{8.3.4 Refresh device read (in 16-bit units)}

\section*{REFDVRDW}

- QnUD(H)CPU and QnUDE(H)CPU: Supported if first 5 digits of the serial number are "14072" or later
- Built-in Ethernet port LCPU: Supported
- Not supported for Q00UJCPU, Q00UCPU, Q01UCPU, Q02UCPU, QnUDVCPU, L02SCPU, and L02SCPU-P

S(P)_REFDVRDW
P: Execution condition

\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{\multirow[t]{2}{*}{\begin{tabular}{l}
\(\square\) indicates any of the following instructions. \\
S_REFDVRDW \\
SP_REFDVRDW
\end{tabular}}} \\
\hline & \\
\hline
\end{tabular}

Input argument,
n1: \(\quad\) Start I/O number \(\left(0_{\mathrm{H}}\right.\) to \(\left.\mathrm{FE}_{\mathrm{H}}\right)(\mathrm{BIN} 16 \text {-bit })^{* 1}\) of the master
:Bit
:ANY16
station controlling the station assigned the refresh device which reads data
s1: Start number or the array of the device stored control data :Array of ANY16 (0..3)
(device name)
d1: Start number of the device stored read data to the refresh :ANY16
device assigned the device specified in (51 +0 and (51) +1
(device name)
n2: Number of read points (1 to 2147483647) :ANY32
Output argument,
ENO: Execution result :Bit
d2: Start number or the array of the bit device which turns on for 1 :Array of Bit (0..1)
scan by the instruction completion. (d2) +1 also turns on at the
error completion (bit).
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{} & \multirow[b]{2}{*}{U:..agam} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant
\[
\mathrm{K}, \mathrm{H}
\]} & \multirow[t]{2}{*}{Others} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline n1 & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & \(\bigcirc\) & - \\
\hline (51) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & - & - \\
\hline (d1) & - & \multicolumn{2}{|c|}{\(\triangle^{* 2}\)} & & & \multicolumn{2}{|l|}{-} & - & - \\
\hline n2 & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & & & \multicolumn{2}{|l|}{-} & \(\bigcirc\) & - \\
\hline (d2) & \(\Delta^{* 2}\) & \multicolumn{2}{|c|}{-} & & & \multicolumn{2}{|l|}{-} & - & - \\
\hline
\end{tabular}

\footnotetext{
*1: The first 3 digits of the hexadecimal 4 digits which represent the start I/O number.
*2: Local devices and the devices designated for individual programs cannot be used.
}

\section*{\(\sqrt{2}\) Function}
(1) The contents of the data stored in the area starting from ©11 are as indicated below.
\begin{tabular}{|c|c|c|c|}
\hline Device & Item & Setting contents & Setting range \\
\hline (51) +0 & Station number & \begin{tabular}{l}
Station number for the station assigned the refresh device which reads data. \\
When the link special register (SW) is specified as type of \\
(s1) +1 , the setting is disabled.
\end{tabular} & 1 to 120 \\
\hline (51) +1 & Type & \begin{tabular}{l}
Type of the refresh device which reads data \\
1: Remote register (RWr) \\
2: Remote register (RWw) \\
3: Link special register (SW)
\end{tabular} & 1 to 3 \\
\hline \[
\begin{aligned}
& \text { (s1) }+2 \\
& \text { (51) }+3
\end{aligned}
\] & Offset & Offset from the head of the refresh device assigned the device specified in (s1) +0 and (s1) +1 . & 0 to 2147483647 \\
\hline
\end{tabular}
(2) Instruction execution possibility of an execution type for each program.
(a) Enabled: Initial program and scan execution type program
(b) Disabled: Fixed scan execution type program and interrupt program
(3) To read data to a refresh device, the data reflection to the station number specified by the instruction is executed at the timing for the auto refresh.
(4) Number of points specified in \(n 2\) is read from the device specified in \({ }^{2}\) ) to the offset specified in \((51)+2\) of the refresh device assigned for the device specified in ©1 +1 of the target station specified in n 1 and (s1) +0 .


Refresh device areas assigned for station No. 4
At the above configuration, number of points specified in n 2 is read starting from the offset (W1063) specified in \(\$ 1_{1}+2\) of the device assigned for the station number 4, then the number of points is read to devices starting from the device specified in (d1).

Start number of the device where target data is stored:


\section*{XPOINT}

When a refresh range per station is assigned in transfer settings, specify number of read points so that the range read data from the specified offset is within the range assigned in the same transfer setting. An error occurs if the number of read points over the range assigned in each transfer setting is specified.
(5) The station type which can and cannot specify with the start I/O number is as follows.
\begin{tabular}{l|l}
\multicolumn{1}{c|}{\begin{tabular}{c} 
Specification \\
possibility
\end{tabular}} & \multicolumn{1}{c}{ Station type } \\
\hline Enabled & \begin{tabular}{l} 
CC-Link master station, CC-Link master station (compatible with redundant function), CC-Link \\
IE Field Network master station
\end{tabular} \\
\hline Disabled & \begin{tabular}{l} 
CC-Link local station, CC-Link standby master station, CC-Link IE Field Network local station, \\
CC-Link IE Field Network submaster station
\end{tabular} \\
\hline
\end{tabular}
(6) Because the available range of the station number is 1 to 120 , the station number for the master station in n 1 cannot be specified to \(\left(\mathrm{sil}^{2}+0\right.\). If the station number is specified, the "OPERATION ERROR" (error code: 4102) occurs.
(7) SM739 (Refresh device write/read instruction in execution flag) turns on during the instruction execution. When SM739 is on, the following instructions cannot be executed.
- S(P)_REFDVWRB
- S(P)_REFDVWRW
- S(P)_REFDVRDB
- S(P)_REFDVRDW

If these instructions are executed, no processing is performed. When an error is detected at the instruction execution (before SM739 turns on), the completion device (d2)+0), the completion device ( \(\mathrm{d} 22^{2}+1\) ), and SM739 do not turn on.
(8) If these instructions are executed, no processing is performed. When an error is detected at the instruction execution (before SM739 turns ON), the completion device (©2)+0), the completion device ( (d2) +1 ), and SM739 do not turn on.
(a) Completion device (d2)+0)

The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.
(b) Completion device \((\) (12) +1\()\)

The device turns on or off by the status when the instruction is completed.
- Normal completion: No change from off
- Error completion: The device turns on at the END processing in a scan where the instruction is completed and turns off at the next END processing.

(9) A module set parameters by the dedicated instruction and a CC-Link module operating by the automatic CC-Link startup cannot be specified with the instruction.
(10) The read source (points in n2 from a device specified in control data) and read destination (points in n 2 from (d1) are overlapped, data can be read. Read data starting from a device specified in control data when data are read to the smaller device number. Read data starting from a device specified in control data @11+((n2)-1) when data are read to the larger device number.

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) turns on, and an error code is stored into SD0.
- When the instruction is used with the CPU module whose serial number (first five digits) is "14071" or earlier
(Error code: 4002)
- When the module cannot be specified the start I/O number in n1
(Error code: 4002)
- When the device which cannot be specified is specified
(Error code: 4004)
- When the specified device exceeds the range of the number of device points
(Error code: 4101)
- When the start I/O number in n 1 is out of the specified range
(Error code: 4101)
- When the device type number in (51)+1 is out of the specified range
(Error code: 4101)
- When the read offset in \(\mathrm{S} 11^{\text {+ }}+2\) is out of the specified range
(Error code: 4101)
- When the number of read points in n 2 is out of the specified range
(Error code: 4101)
- When the number of read points in n 2 exceeds the range of the number of device points
(Error code: 4101)
- When the station number in (51)+0 is out of the specified range
(Error code: 4102)
- When the station number in (51)+0 does not exist
(Error code: 4102)
- When the station number in §11 \(^{2} 0\) is the master station specified in n 1 (Error code: 4102)
- When the start I/O number in n 1 is the station type which cannot be specified
(Error code: 4150)
- When the start I/O number in n1 does not exist in the network parameter
(Error code: 4150)
- When the device in \(51+1\) of the station number specified in \(\subseteq 51+0\) is not assigned the refresh device
(Error code: 4151)
- When the read offset in \((51+2\) exceeds the refresh device range assigned for the device in (51) +1 of the station number specified in (51) +0
(Error code: 4151)
- When the number of read points in n 2 exceeds the assignment range of the setting for one transfer from the read offset in ©11 +2
(Error code: 4151)

\section*{\(\square\) Program Example}

The following program reads 16 device values from the head of the refresh device (offset: 0 ) assigned for the remote register ( RWr ) in the remote device station on the station number 32 controlled by the CC-Link master station of the start I/O number \(0080_{\mathrm{H}}\) to W 100 when X 1 C is turned on.
[Structured ladder/FBD]

```

[ST]
MOV( LDP( TRUE , X1C ), K32 , D100 );
MOV( LDP( TRUE , X1C ) , K1 , D101 );
DMOV( LDP( TRUE , X1C ), H0, D102 );
S_REFDVRDW( LDP( TRUE , X1C ), H8, D100,W100, K16, M100 );
IF M100=TRUE THEN
IF M101=FALSE THEN
SET( TRUE , M150 );
ELSE
SET( TRUE , M151 );
END_IF;
END_IF;

```
(1) Do not execute the instruction in an interrupt program. If the instruction is executed, no processing is performed. In addition, the completion device ( ( \(12+0)\), the completion device (d2) +1 ), and SM739 do not turn on. If the instruction is executed in a fixed scan execution type program, they also do not turn on.
(2) Specifying digit for the bit device can be used only when the following conditions (a) and (b) are met.
(a) Digit specification: K4
(b) Head of device: multiple of 16

When the above conditions (a) and (b) are not met, INSTRCT CODE ERR. (error code: 4004) will occur.

MEMO

\section*{MULTPLECPU DEDCATED NSTRUCTIONS}
9.1 Writing Data to Host CPU Shared Memory ..... 9-2
9.2 Reading Data from Other CPU Shared Memory ..... 9-14

\subsection*{9.1 Writing Data to Host CPU Shared Memory}

The \(S(P)_{-}\)TO or \(T O(P)\) instruction is used to write to the host CPU shared memory in the multiple CPU system.

The following table indicates the usability of the \(S(P)\) _TO and \(T O(P)\) instructions.
\begin{tabular}{|c|c|c|c|}
\hline \multicolumn{2}{|r|}{CPU module model} & S(P) TO instruction & TO(P) instruction \\
\hline \multirow{2}{*}{Basic model QCPU} & Q00JCPU & Unusable & Unusable \\
\hline & Q00CPU and Q01CPU & Usable & Usable \\
\hline High Performance model QCPU & Q02CPU, Q02HCPU, Q06HCPU, Q12HCPU, and Q25HCPU & Usable & Unusable \\
\hline Process CPU & Q02PHCPU, Q06PHCPU, Q12PHCPU and Q25PHCPU & Usable & Unusable \\
\hline Redundant CPU & Q12PRHCPU and Q25PRHCPU & Unusable & Unusable \\
\hline & Q00UJCPU & Unusable & Unusable \\
\hline Universal model QCPU & Q00UCPU, Q01UCPU, Q02UCPU, Q03UDCPU, Q03UDVCPU, Q03UDECPU, Q04UDHCPU, Q04UDVCPU, Q04UDEHCPU, Q06UDHCPU, Q06UDVCPU, Q06UDEHCPU, Q10UDHCPU, Q10UDEHCPU, Q13UDHCPU, Q13UDVCPU, Q13UDEHCPU, Q20UDHCPU, Q20UDEHCPU, Q26UDHCPU, Q26UDVCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU & Usable & Usable \\
\hline LCPU & L02SCPU, L02SCPU-P, L02CPU, L02CPU-P, L06CPU, L06CPU-P, L26CPU, L26CPU-P, L26CPU-BT, L26CPU-PBT & Unusable & Unusable \\
\hline
\end{tabular}
(1) Operation of \(\mathrm{S}(\mathrm{P})_{-} \mathrm{TO}\) instruction

The \(S(P)\) _TO instruction can write data to the CPU shared memory of the host CPU module. The following figure shows the processing performed when the \(S(P)_{-}\)TO instruction is executed in CPU No. 1.

(2) Operation of \(\mathrm{TO}(\mathrm{P})\) instruction

The \(\mathrm{TO}(\mathrm{P})\) instruction can write device memory data to the following memory.
- CPU shared memory of the host CPU module
- Buffer memory of the intelligent function module

The following figure shows the processing performed when the \(\mathrm{TO}(\mathrm{P})\) instruction is executed in CPU No. 1.


\section*{XPOINT}

For Basic model QCPU (Q00CPU, Q01CPU) and Universal model QCPU, both \(S(P)_{-}\)TO and \(T O(P)\) instructions can be used to write data to the CPU shared memory. However, use of the \(\mathrm{TO}(\mathrm{P})\) instruction is recommended in order to reduce the number of steps and shorten the processing time for writing data to the host CPU shared memory.

\section*{Remark}

For writing to the buffer memory of the intelligent function module by the \(T(P)\) instruction, refer to Section 7.8.2.

\subsection*{9.1.1 Writing data to host CPU shared memory}

- Q00CPU and Q01CPU: Supported if first 5 digits of the serial number are "04122" or later
- High Performance model QCPU: Supported if the function version B or later

S(P)_TO
\(P\) : Executing condition


\section*{Function}
(1) Writes (54) words of device data from host CPU module \(\circledR_{3}\) to the CPU shared memory address specified for (®2) and the following addresses of host CPU module.
When writing is completed, the completion bit specified for (a) turns ON.
Host CPU

(a) CPU shared memory address of the Basic model QCPU

(b) CPU shared memory address of the High Performance model QCPU, Process CPU, and Universal model QCPU*2

*1: Usable as a user setting area when auto refresh setting is not set.
In addition, even when auto refresh setting is set, the auto refresh send range or later is usable as a user setting area.
*2: Data cannot be written to the multiple CPU high speed transmission area of the Universal model QCPU using the \(S(P)\) _TO instruction.
(2) When the number of write points is 0 , no processing is performed and the completion device does not turn ON.
(3) The \(S(P)\) TO instruction can be executed once per one scan for each CPU.

When execution condition is established at two or more places at the same time, the \(S(P)\) _TO instruction executed later is not processed since handshake is established automatically.
(4) The number of data that can be written varies depending on the target CPU module.
\begin{tabular}{c|c} 
CPU module & \begin{tabular}{c} 
Number of write \\
points
\end{tabular} \\
\hline Basic model QCPU & 1 to 320 \\
\hline \begin{tabular}{c} 
High Performance model QCPU, \\
Process CPU
\end{tabular} & 1 to 256 \\
\hline Universal model QCPU & 1 to 2048 \\
\hline
\end{tabular}

XPOINT
Data Write to CPU shared memory can be performed using the intelligent function module device.
For the intelligent function module device, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
(1) When the specified value is outside of the following range.
(Error code: 4101)
- When the number of write points (54) is outside the specified range of set data.
- When the start of the CPU shared memory address (s2) of the write destination host CPU exceeds the CPU shared memory address range.
- When the CPU shared memory address (®2 + the number of write points (s4) exceeds the CPU shared memory address range.
- When the start number of the devices \(\overbrace{3}\) where the data to be written is stored + the number of write points (54) exceeds the device range.
(2) When the host CPU operation information area, restricted system area or host CPU refresh area is specified to the CPU shared memory address ©2 of the write destination.
(For High Performance model QCPU, Process CPU)
(Error code: 4101)
(For Basic model QCPU and Universal model QCPU)
(Error code: 4111)
(3) When the start I/O number (s1) of the host CPU is other than that of the host CPU.
(For High Performance model QCPU, Process CPU)
(Error code: 2107)
(For Basic model QCPU and Universal model QCPU)
(Error code: 4112)
(4) The CPU module does not exist in the position specified by the start I/O number of the CPU module.
(Error code: 2110)
(5) When the start I/O number (51) of the host CPU is other than \(3 \mathrm{E} 0 \mathrm{H}, 3 \mathrm{E} 1 \mathrm{H}, 3 \mathrm{E} 2 \mathrm{H}\) or 3 E 3 H .
(Error code: 4100)
(6) When the specified instruction is improper.
(Error code: 4002)
(7) When the specified number of devices is wrong.
(Error code: 4003)
(8) When the unusable device is specified.
(Error code: 4004)

\section*{\(\square\) Program Example}

In the following program, 10 points of data from Var_D0 are stored to the address 800 H of the CPU shared memory of CPU No. 1 when X0 turns ON.
[Structured ladder/FBD]

[ST]
SP_TO(X0,H3E0,H800,Var_D0,10,Var_M0);

\section*{Remark}

The ©s1) is specified by the first 3 digits of the hexadecimal 4 digits which represent the start I/O number of the slot mounted to the CPU module.
\begin{tabular}{c|c|c|c|c|}
\hline & CPU Slot & Slot 0 & Slot 1 & Slot 2 \\
\hline Start I/O number & 3 E 00 & 3 E 10 & 3 E 20 & 3 E 30 \\
\hline (11) & 3E0 & 3 E 1 & 3 E 2 & 3 E 3 \\
\hline
\end{tabular}

\subsection*{9.1.2 Writing data to host CPU shared memory}

- Q00CPU and Q01CPU: Supported if first 5 digits of the serial number are " 04122 " or later


\begin{tabular}{|lc|}
\hline instructions. & \\
TO & TOP \\
DTO & DTOP \\
\\
& \\
\\
& \\
\hline
\end{tabular}

Input argument,

Output argument,

EN: Executing condition
s: Write data, or start number of the device that stores write data :ANY16/32
n1: Start I/O number of the host CPU*1
:ANY16
Basic model QCPU: 3EOH
Universal model QCPU: 3E0H to 3 E 3 H
n2: Host CPU shared memory address of the write destination :ANY16
Basic model QCPU: 192 to 511
Universal model QCPU: 2048 to 4095, 10000 to \(24335^{* 2}\)
n3: \(\quad\) Number of write points 1 to 320
:ANY16
Basic model QCPU: TO(P) : 1 to 320
DTO(P) : 1 to 160
Universal model QCPU: TO(P) : 1 to \(14336^{* 2}\)
DTO(P) : 1 to \(7168^{* 2}\)
ENO: Execution result
*1 : Specified by the upper 3 digits of start I/O number expressed in 4-digit hexadecimal.
*2 :The setting range varies depending on the auto refresh setting range of the multiple CPU high speed transmission function.
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{小...alin} & \multirow[b]{2}{*}{U...igal} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K,H} & \multirow[t]{2}{*}{Others U} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (s) & \multicolumn{3}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline n1 & \multicolumn{3}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & \(\bigcirc\) \\
\hline n2 & \multicolumn{3}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline n3 & \multicolumn{3}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{\(\bigcirc\)} & \(\bigcirc\) & - \\
\hline
\end{tabular}

\section*{TO}
(1) Writes n3 words of device data from host CPU module ©s to the CPU shared memory address specified for n 2 and the following addresses of host CPU module.


When a constant is specified for (s), writes the same data (value specified for (s) to the area of n3 words from the specified CPU shared memory.

(a) CPU shared memory address of the Basic model QCPU

(b) CPU shared memory address of the Universal model QCPU*3

*2: Usable as a user setting area when auto refresh setting is not set.
In addition, even when auto refresh setting is set, the auto refresh send range or later is usable as a user setting area.
*3: With Q02UCPU, data cannot be written to the multiple CPU high speed transmission area.
(2) No processing is performed when the number of write points is 0 .
(3) The number of data that can be written varies depending on the target CPU module.
\begin{tabular}{c|c} 
CPU module & Number of write points \\
Basic model QCPU & 1 to 320 \\
\hline Universal model QCPU & 1 to 14336 \\
\hline
\end{tabular}

\section*{DTO}
(1) Writes ( \(\mathrm{n} 3 \times 2\) ) words of device data from host CPU module © to the CPU shared memory address specified for n 2 and the following addresses of host CPU module.

Host CPU


When a constant is specified for © ©, writes the same data (value specified for ©s) to the area of ( \(\mathrm{n} 3 \times 2\) ) words from the specified CPU shared memory.

(2) No processing is performed when the number of write points is 0 .
(3) The number of data that can be written varies depending on the target CPU module.
\begin{tabular}{c|c} 
CPU module & Number of write points \\
Basic model QCPU & 1 to 160 \\
\hline Universal model QCPU & 1 to 7168 \\
\hline
\end{tabular}

\section*{XPOINT}

Data Write to CPU shared memory can be performed using the intelligent function module device.
For the intelligent function module device, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SDO.
(1) When the specified value is outside of the following range.
(Error code: 4101)
- When the number of write points (n3) is outside the specified range of set data.
- When the beginning of the CPU shared memory address (n2) + number of write points (n3) of the write destination host CPU exceeds the CPU shared memory range.
- When the start number of the devices that stores the data to be written (ⓢ) + the number of write points (n3) exceeds the device range.
- When the value that exceeds the write specification permitted area is specified for the start CPU shared memory address ( n 2 ) of the write destination.
(2) When the value for the start of the CPU shared memory address ( n 2 ) of the write destination is not valid.
(Error code: 4111)
(3) When the I/O number specified for ( n 1 ) is other than that of the host station. (Except when the multiple CPU high speed transmission area of the other station is specified.)
(Error code: 4112)
(4) When the CPU module does not exist at the position where the start I/O number is specified.
(Error code: 2110)

\section*{\(\square\) Program Example}
(1) In the following program, 10 points of data from g_int1 are stored to the address 10000 of the CPU shared memory of CPU No. 1 when g_bool 1 turns ON.
[Structured ladder/FBD]

[ST]
TOP(g_bool1, g_int1, H3E0, 10000, 10);
(2) In the following program, 20 points of data from g_dint1 are stored to the address 10000 of the CPU shared memory of CPU No. 4 when g_bool1 turns ON.
[Structured ladder/FBD]

[ST]
DTOP(g_bool1, g_dint1, H3E3, 10000, 20);

\section*{Remark}

The n 1 is specified by the first 3 digits of the hexadecimal 4 digits which represent the start I/O number of the slot mounted to the CPU module.
\begin{tabular}{c|c|c|c|c}
\hline & CPU Slot & Slot 0 & Slot 1 & Slot 2 \\
\hline Start I/O number & 3 E 00 & 3 E 10 & 3 E 20 & 3 E 30 \\
\hline n 1 & 3 E 0 & 3 E 1 & 3 E 2 & 3 E 3 \\
\hline
\end{tabular}

\subsection*{9.2 Reading Data from Other CPU Shared Memory}

Data can be read using the \(\operatorname{FROM}(\mathrm{P}) / \mathrm{DFRO}(\mathrm{P})\) instruction of multiple CPU system from the following memories.
- Buffer memory of intelligent function module
- CPU shared memory of other CPU module
- CPU shared memory of host CPU module (Executable in the Basic model QCPU and the Universal model QCPU)

The following figure shows the processing performed when the \(\operatorname{FROM}(\mathrm{P})\) instruction is executed in CPU No. 1.


\section*{Remark}

For reading the buffer memory of the intelligent function module with the FROM(P)/DFRO(P) instruction, refer to Section 7.8.1.

\subsection*{9.2.1 Reading data from other CPU shared memory}

FROM, DFRO

- Q00CPU and Q01CPU: Supported if first 5 digits of the serial number are " 04122 " or later
- High Performance model QCPU: Supported if the function version B or later

1 For Basic model QCPU and Universal model QCPU


\section*{I Function}

\section*{FROM}
(1) Reads n3 words of data from the CPU shared memory address specified for \(n 2\) of the CPU module specified for n 1 , and stores the data to (d) and the following devices.
(d)


(a) CPU shared memory address of the Basic model QCPU

CPU shared memory address

(b) CPU shared memory address of the Universal model QCPU*3

*2: Usable as a user setting area when auto refresh setting is not set.
When auto refresh setting is set, the auto refresh send range and later are usable as a user setting area.
*3: With Q02UCPU, data cannot be read from the multiple CPU high speed transmission area.
(2) When 0 is specified for n 3 as the number of data to be read, no processing is performed.
(3) The number of data to be read changes depending on the target CPU module.
\begin{tabular}{l|c}
\multicolumn{1}{|c|}{ CPU module } & Number of read points \\
\hline Basic model QCPU & 1 to 512 \\
\hline Universal model QCPU & 1 to 14336 \\
\hline
\end{tabular}

\section*{DFRO}
(1) Reads ( \(n 3 \times 2\) ) words of data from the CPU shared memory address specified for \(n 2\) of the CPU module specified for n 1 , and stores the data to (d) and the following devices.

(2) No processing is performed when read data n 3 is 0 .
(3) The number of data that can be read varies depending on the target CPU module.
\begin{tabular}{l|c}
\multicolumn{1}{|c|}{ CPU module } & Number of read points \\
\hline Basic model QCPU & 1 to 256 \\
\hline Universal model QCPU & 1 to 7168 \\
\hline
\end{tabular}

\section*{®POINT}

Data read from the CPU shared memory can also be performed using the intelligent function module devices.
For the intelligent function module device, refer to the User's Manual (Functions Explanation, Program Fundamentals) of the CPU module to be used.

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
(1) When the specified value is outside of the following range.
(Error code: 4101)
- The address of the CPU shared memory ( n 2 ) from which data are read plus the number of read points ( n 3 ) is outside of the CPU shared memory range.
- The address of the CPU shared memory (n2) from which data are read plus the number of read points ( n 3 ) is outside of the CPU shared memory range.
- The read data storage device number (©) plus the number of read points (n3) is outside of the specified device range.
(2) The CPU module does not exist in the position specified by the CPU module start I/O number.
(Error code: 2110)
(3) When the start of the CPU shared memory address ( n 2 ) which performs reading is an invalid value. (4097 to 9999)
(Error code: 4101)

\section*{\(\square\) Program Example}
(1) In the following program, 10 points of data from the address 800 H of the CPU shared memory of CPU No. 2 are stored to Var_D0 and the following devices when X0 turns ON. [Structured ladder/FBD]


\section*{[ST]}

FROM(X0,H3E1,H800,10,Var_D0);
(2) In the following program, 20 points of data from the address 10000 of the CPU shared memory of CPU No. 4 are stored to D0 and the following devices when X0 turns ON.
[Structured ladder/FBD]


\section*{[ST]}

DFROP(X0,H3E3,K10000,20,D0);

\section*{Remark}

The n 1 is specified by the first 3 digits of the hexadecimal 4 digits which represent the start I/O number of the slot mounted to the CPU module.
\begin{tabular}{c|c|c|c|c} 
& CPU Slot & Slot 0 & Slot 1 & Slot 2 \\
Start I/O number & 3 E 00 & 3 E 10 & 3 E 20 & 3 E 30 \\
\hline n 1 & 3 E 0 & 3 E 1 & 3 E 2 & 3 E 3 \\
\hline
\end{tabular}

The QCPU provides automatic interlocks for the FROM and TO instructions.

\section*{2 For High Performance model QCPU and Process CPU}

FROM(P)
P: Executing condition

\begin{tabular}{|l|l|}
\hline FROM & FROMP \\
instructions. \\
\\
\\
\\
\\
\hline
\end{tabular}

Input argument,
\begin{tabular}{|c|c|c|}
\hline EN: & Executing condition & :Bit \\
\hline n 1 : & Start I/O number of the read target CPU module *1 3EOH to 3E3H & :ANY16 \\
\hline n2: & CPU shared memory address of the read destination 0 to 4095 & :ANY16 \\
\hline n3: & Number of data to be read 1 to 4096 & :ANY16 \\
\hline ENO: & Execution result & :Bit \\
\hline d: & Start number of the device that stores read data & :ANY16 \\
\hline
\end{tabular}
*1 : Specified by the upper 3 digits of start I/O number expressed in 4-digit hexadecimal.
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J..} & \multirow[b]{2}{*}{Ulalal} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{Constant K,H} & \multirow[t]{2}{*}{Others U} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline n1 & - & & & & & \(\bigcirc\) & & \(\bigcirc\) & \(\bigcirc\) \\
\hline n2 & - & & & & & \(\bigcirc\) & & \(\bigcirc\) & - \\
\hline n3 & - & & & & & \(\bigcirc\) & & \(\bigcirc\) & - \\
\hline (d) & - & & & & & - & & - & - \\
\hline
\end{tabular}
(1) Reads n3 words of data from the CPU shared memory address specified for \(n 2\) of the CPU module specified for n 1 , and stores the data to (d) and the following devices.


CPU shared memory address of the High Performance model QCPU and Process CPU

*1: Usable as a user setting area when auto refresh setting is not set.
When auto refresh setting is set, the auto refresh send range and later are usable as a user setting area.
(2) When 0 is specified for n 3 as the number of data to be read, no processing is performed.
(3) The number of data to be read changes depending on the target CPU module.
\begin{tabular}{l|c|}
\multicolumn{1}{c|}{ CPU module } & Number of read points \\
\hline \begin{tabular}{l} 
High Performance model QCPU and \\
Process CPU
\end{tabular} & 1 to 4096 \\
\hline
\end{tabular}

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
(1) When the specified value is outside of the following range.
(Error code: 4101)
- The address of the CPU shared memory (n2) from which data are read plus the number of read points ( n 3 ) is outside of the CPU shared memory range.
- The address of the CPU shared memory ( n 2 ) from which data are read plus the number of read points (n3) is outside of the CPU shared memory range.
- The read data storage device number (@) plus the number of read points (n3) is outside of the specified device range.
(2) The CPU module does not exist in the position specified by the CPU module start I/O number.
(Error code: 2110)
(3) When the start of the CPU shared memory address ( n 2 ) which performs reading is an invalid value. (4097 to 9999)
(Error code: 4101)

\section*{\(\square\) Program Example}

In the following program, 10 points of data from the address 800 H of the CPU shared memory of CPU No. 2 are stored to Var_D0 and the following devices when X0 turns ON.
[Structured ladder/FBD]

[ST]
FROM(X0,H3E1,H800,10,Var_D0);

\section*{Remark}

The n 1 is specified by the first 3 digits of the hexadecimal 4 digits which represent the start I/O number of the slot mounted to the CPU module.
\begin{tabular}{c|c|c|c|c|}
\hline & CPU Slot & Slot 0 & Slot 1 & Slot 2 \\
\hline Start I/O number & 3E00 & 3E10 & \(3 E 20\) & \(3 E 30\) \\
\hline n 1 & 3E0 & 3E1 & \(3 E 2\) & \(3 E 3\) \\
\hline
\end{tabular}

The QCPU provides automatic interlocks for the FROM and TO instructions.

\section*{MULTPLE CPU HIGHSPEED TRASSMSSOON DEDCATED INSTRUCTONS}
10.1 Overview. ..... 10-2
10.2 Writing Device Data to Other CPUs ..... 10-12
10.3 Reading Device Data from Other CPUs ..... 10-16

\subsection*{10.1 Overview}

The multiple CPU high speed transmission dedicated instruction is an instruction used to read and write device data between the multiple Universal model QCPUs.

The following figure shows the operation when the data are read from the CPU No. 1 and written to the CPU No. 2 using the multiple CPU high speed transmission dedicated instruction.


\section*{XPOINT}

This instruction can be used for the following CPU modules only, including the ones used as host CPU and other CPU (target CPU for instruction execution).
- Q3UDCPU, Q4UDHCPU or Q06UDHCPU with a serial number whose first five digits are '10012' or higher
- Q10UDHCPU, Q13UDHCPU, Q20UDHCPU, Q26UDHCPU
- QnUDVCPU
- QnUDE(H)CPU
(1) System configuration and parameter settings for executing the multiple CPU high speed transmission dedicated instruction

This instruction can be executed under the following system configuration and parameter settings.
- QnUD(H)CPU, QnUDVCPU, or QnUDE(H)CPU is used as the CPU No.1.
- Multiple CPU high speed main base unit (Q3)DB) is used.
- "Multiple CPU High Speed Transmission Area Setting is valid." is selected in the multiple CPU setting of the PLC parameter.
(2) Devices that can be written/read

Names of devices that can be written/read
The following table shows the devices that can be written to/read from the other Universal model QCPUs using the multiple CPU high transmission dedicated instruction.
\begin{tabular}{|c|c|c|c|c|}
\hline Classification & Type & Device name & Target device applicable/not applicable & Remarks \\
\hline \multirow[t]{2}{*}{Internal user device} & Bit device & X, Y, M, L, B, F, SB & \(\triangle\) & \begin{tabular}{l}
Required conditions at the setting \\
- Digit specification of bit devices (K4/16 bits). \\
- The start number of bit device is a multiple of \(16(10 \mathrm{H})\).
\end{tabular} \\
\hline & Word device & T, ST, C, D, W, SW & \(\bigcirc\) & - \\
\hline \multirow[t]{2}{*}{Internal system device} & Bit device & SM & \(\triangle\) & \begin{tabular}{l}
Required conditions at the setting \\
- Digit specification of bit devices (K4/16 bits). \\
- The start number of bit device is a multiple of \(16(10 \mathrm{H})\).
\end{tabular} \\
\hline & Word device & SD & \(\bigcirc\) & - \\
\hline File register & Word device & R, ZR & \(\bigcirc\) & - \\
\hline
\end{tabular}
\(\bigcirc\) : Applicable \(\triangle\) : Applicable with required conditions

\section*{®POINT}

Devices SB, SW, SM, and SD contain the system information area.
When writing data to devices described above using the multiple CPU high speed transmission dedicated instruction \(D(P)\) _DDWR, be careful not to destroy the system information.
(3) Device specification method and applicable device range for writing/reading data The device specification and character string specification are the two types of device specification method for other CPUs. The applicable device range for writing/reading data to/from the other CPU is different between the device specification and character string specification.
(a) Device specification

The device specification is a method for specifying devices of the other CPU directly. Device specification of the DP_DDWR instruction


\(\boxed{\text { Directly specifies the device 'D200' }}\) of the other CPU to be written.

By using the device specification method, data can be written/read within the device range of the host CPU.
For example, when the data register of the host CPU is 12 K points, and the data register of other CPU is 16 K points, data of 12 K points can be written/read from the start of the data register of the other CPU.

The range for writing/reading data when the device is specified

(b) Character string specification

The character string specification is a method for specifying devices of the other CPU using the character string.

Character string specification of the DP_DDWR instruction


By using the character string specification method, data can be written/read within the whole device range of the other CPU.
For example, when the data register of the host CPU is 12 K points, and the data register of the other CPU is 16 K points, data of 16 K points can be written/read from the start of the data register of the other CPU.
The range for writing/reading data when the character string is specified


Remark
The following describes the precautions on the character string specification.
- Up to 32 characters can be specified.
- Any 0 s which are appended to upper digits of device number are ignored. The devices with the same lower digits are processed as the same device. For example, 'D1' and 'D0001' are processed as 'D1'.
- Devices specified with uppercase and lowercase of the same letter are processed as a same device. For example, 'D1' and 'd1' are processed as 'D1'.
- If devices that do not exist in other CPUs are specified by the character string, the instruction completes with an error.
(4) Managing the multiple CPU high speed transmission area
(a) The multiple CPU high speed transmission area is managed in blocks and each block contains minimum of 16 words.
The following table shows the number of blocks that can be used in each CPU and the number of blocks used for the instruction.
\begin{tabular}{c|c|c}
\multirow{2}{*}{ Number of CPUs } & \multicolumn{2}{|c}{ System area*1 \(^{*}\)} \\
\cline { 2 - 3 } & 1K point & 2K points \\
\hline 2 & 46 & 110 \\
\hline 3 & 22 & 54 \\
\hline 4 & 14 & 35 \\
\hline
\end{tabular}
*1 : For details of the system area settings, refer to QCPU User's Manual (Multiple CPU System).
(b) The following figure shows the configuration of the multiple CPU high speed transmission area when the system area size is 1 k word in the configuration of multiple CPU system with three CPU modules.

(5) Number of blocks used for the instruction

The number of blocks used for the instruction differs by the points to be written. The following table shows the number of blocks used for the instruction.
\begin{tabular}{|c|c|c|}
\hline Number of write/read points specified by the instruction & \(D(P)\) _DDWR instruction & \(D(P)\) _DDRD instruction \\
\hline 1 to 4 & 1 & \multirow{7}{*}{1} \\
\hline 5 to 20 & 2 & \\
\hline 21 to 36 & 3 & \\
\hline 37 to 52 & 4 & \\
\hline 53 to 68 & 5 & \\
\hline 69 to 84 & 6 & \\
\hline 85 to 100 & 7 & \\
\hline
\end{tabular}
(6) Multiple CPU high speed transmission dedicated instructions that can be executed simultaneously
For Universal model QCPU, the multiple CPU high speed transmission dedicated instructions can be executed simultaneously within the following range.
\[
\left[\begin{array}{l}
\text { Number of blocks that } \\
\text { can be used for each CPU }
\end{array}\right] \geqq\left[\begin{array}{l}
\text { Total of blocks used by the } \\
\text { simultaneously executed instructions }
\end{array}\right]
\]

By executing the multiple CPU high speed transmission dedicated instruction, if the number of blocks used by the multiple CPU high speed transmission dedicated instruction exceeds the total number of blocks of the multiple CPU high speed transmission area, this instruction is not executed (not processed) in the scan but it is executed again in the next scan. Note that, if the number of empty blocks of the multiple CPU high speed transmission area is fewer than the set value of the special registers SD796 to SD799 (for setting the maximum number of blocks for the multiple CPU high speed transmission dedicated instruction), this instruction completes with an error.
(a) The following table shows the instruction execution applicability by the amount of blocks used for the multiple CPU high speed transmission dedicated instruction, and when the number of empty blocks of multiple CPU high speed transmission area is fewer than the set value of the special registers SD796 to SD799.
\begin{tabular}{|c|c|c|}
\hline \begin{tabular}{l}
Relation between the number of blocks used by the instruction \({ }^{* 1}\) and the number of empty blocks \\
Relations between the SD set value and the number of empty blocks
\end{tabular} & Number of blocks used by the instruction*1 \(\leqq\) Number of empty blocks \({ }^{*}{ }^{2}\) & Number of blocks used by the instruction*1 \(>\) Number of empty blocks*2 \\
\hline \[
\begin{aligned}
& \hline \text { SD set value }{ }^{* 3} \leqq \\
& \text { Number of empty blocks }{ }^{\star} 2 \\
& \hline
\end{aligned}
\] & Executed & Not executed (not processed) \\
\hline SD set value \({ }^{*}\) > Number of empty blocks \({ }^{*} 2\) & \multicolumn{2}{|c|}{Completes with an error} \\
\hline
\end{tabular}
*1: Number of blocks used by the multiple CPU high speed transmission dedicated instruction
*2: Number of empty blocks of the multiple CPU high speed transmission area
*3: Set value of the special registers SD796 to SD799
(7) Interlocks when using the multiple CPU high speed transmission dedicated instruction
(a) The special relays SM796 to SM799 (information of blocks used for the multiple CPU high speed transmission dedicated instruction) can be used as the interlocks for the multiple CPU high speed transmission dedicated instruction.
When executing two or more multiple CPU high speed transmission dedicated instructions simultaneously, use the special relays SM796 to SM799 as the interlocks for the instruction.

\section*{®POINT}

When using the special relays SM796 to SM799, set the maximum number of blocks for the instruction used in each CPU to the special registers SD796 to SD799. (For example, when the maximum number of blocks for the multiple CPU high speed transmission dedicated instruction that is executed to CPU No. 3 is 5 , set 5 to SD798.)
If the multiple CPU high speed transmission area becomes less than the number of blocks set in the special registers SD796 to SD799, the corresponding special relay (SM796 to SM799) turns ON.

(b) Program example when the special relays SM796 to SM799 are used as the interlocks In the following program, the D_DDWR instruction is executed to the CPU No. 2 at the rising edge of X 0 , and to the CPU No. 3 at the rising edge of X 1 .

(8) Program example when the multiple CPU high speed transmission dedicated instructions are executed from one CPU to another CPU between the multiple CPU modules When executing the multiple CPU high speed transmission dedicated instructions from one CPU to another CPU between the Universal model QCPUs, set the interlocks to avoid the instructions to be executed simultaneously.
The cyclic transmission area devices (from U3En\G10000) are used as the interlocks. In the following program, the multiple CPU high speed transmission dedicated instructions are executed from one CPU to another CPU between the CPU No. 1 and CPU No. 2.

Program example when the multiple CPU high speed transmission dedicated instruction is executed in the CPU No. 1


Program example when the multiple CPU high speed transmission dedicated instruction is executed in the CPU No. 2

(9) Program example when data exceeding 100 words are written/read by the multiple CPU high speed transmission dedicated instruction
Up to 100 words of data can be used for the multiple CPU high speed transmission dedicated instruction. In order to write/read data exceeding 100 words, execute the instruction multiple times.
Note that the \(D(P)\) _DDWR instruction of the multiple CPU high speed transmission dedicated instruction is used in the following program examples, however, the same program configuration can be used for the \(\mathrm{D}(\mathrm{P})_{\mathbf{\prime}}\) DDRD instruction.
(a) Program example in which only one \(D(P)\) _DDWR instruction is executed In the following program, data in the devices ZR0 to ZR999 (1000 points) of the CPU No. 1 are written to the devices ZR0 to ZR999 of the CPU No. 2 using the D_DDWR instruction.
Only one D_DDWR instruction is executed in order to start the next D_DDWR instruction by turning ON the completion device (M2) of the D_DDWR instruction.

Program example in which only one \(D(P)\) _DDWR instruction is executed

(b) Program example in which two or more \(D(P)\) _DDWR instructions are executed simultaneously
In the following program, data in the devices ZR0 to ZR999 (1000 points) of the CPU No. 1 are written to the devices ZR0 to ZR999 of the CPU No. 2 using the D_DDWR instruction.
Two or more instructions to write/read devices between the multiple CPUs can be executed simultaneously as shown in the following program example.
When executing two or more multiple CPU high speed transmission dedicated instructions simultaneously to write/read devices, the higher total number of blocks of the multiple CPU high speed transmission area (send area) shortens the processing time of the instruction.

Program example in which two or more \(D(P)\) _DDWR instructions are executed simultaneously


\subsection*{10.2 Writing Device Data to Other CPUs}

- Universal model QCPU: Supported if first 5 digits of the serial number are "10012" or later
- Q00UJCPU, Q00UCPU, Q01UCPU, and Q02UCPU: Not supported

\begin{tabular}{|l|l|}
\hline D_DDWR & \\
\\
instructions. & \\
\\
\\
\\
\\
\hline
\end{tabular}

Input argument,
n : \(\quad\) Start I/O number of the other CPU divided by 16
CPU No. 1: 3EOн CPU No. 2: 3E1н CPU No. 3: 3E2н CPU
No. 4: 3E3H
s1: Start number or arrays of device in the host CPU that stores : Array of ANY16 (0..1)
control data
s2: Start number of device in the host CPU that stores data to be : ANY16
written
d1: Start number of device in the other CPU that stores data to be : ANY \({ }^{* 1}\)
written
: String \({ }^{* 2,{ }^{*} 3}\)
Output argument, d2: A bit device that turns on when the process is completed : Array of bit (0..1)
*1: When file registers \((R, Z R)\) are specified, data that are outside the range in the host CPU can be written to devices of the other CPUs.
*2: By specifying the start device as a character string " ", data that are outside the range in the host CPU can be written to devices of the other CPUs.
*3: Device that indexed cannot be specified. (Example: DOZO, etc.)
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow{2}{*}{R, RZ} & \multicolumn{2}{|c|}{J等,} & \multirow{2}{*}{U} & \multirow{2}{*}{Zn} & \multicolumn{2}{|l|}{Constant} & \multirow{2}{*}{Others} \\
\hline & Bit & Word *7 & & Bit & Word & & & K, H & \$ & \\
\hline \(\mathrm{n}^{*} 4\) & - & \(\bigcirc\) & \(\bigcirc\) & & & - & & \(\bigcirc\) & - & - \\
\hline (51) \({ }^{5}\) & - & \(\triangle^{*} 6\) & \(\triangle^{*} 6\) & & & - & & - & - & - \\
\hline (s2) \({ }^{5}\) & - & \(\bigcirc\) & \(\bigcirc\) & & & - & & - & - & - \\
\hline (d1) \({ }^{5}\) & - & \(\bigcirc\) & \(\bigcirc\) & & & - & & - & - & - \\
\hline (d2) \({ }^{5}\) & \(\triangle^{* 8}\) & - & \(\triangle{ }^{*} 6\) & & & - & & - & - & - \\
\hline
\end{tabular}
*4: Indexes cannot be set to the setting data n .
*5: Indexes can be set to the setting data (s1) to (d2).
*6: Local devices and file registers per program cannot be used as setting data.
*7: FD cannot be used.
*8: FX and FY cannot be used.

\section*{Control Data}
\begin{tabular}{|c|c|c|c|c|}
\hline Device & Item & Setting data & Setting range & Setting side \\
\hline (51) [0] & Completion status & Execution result of the instruction completion is stored.
\(\begin{array}{ll}0000_{(H)} & \text { : No error (normal completion) } \\ \text { Other than } 0000_{(H)} & \text { : Error code (error completion) }\end{array}\) & - & System \\
\hline (s1) [1] & Number of data to be written & Set the number of data to be written in unit of word. & 1 to 100 & User \\
\hline
\end{tabular}
(1) Writes data stored in the device \(\Im_{2}\) and the following devices specified in the host CPU at the time of multiple CPU system configuration, to the device ©11 and the following devices specified in the CPU No. \(n\) for the number of points specified for (11 [1].

(2) The status of the \(D(P)\) _DDWR instruction completion can be checked by the completion device ( (d2) [0]) and the completion status display device ( (d2) [1]).
(a) Completion device (©22 [0])

Turns ON at the END process of the scan in which the instruction is completed, and turns OFF at the next END process.
(b) Completion status display device (©2 [1])

Turns ON/OFF by the status of the instruction completion.
- Normal completion: OFF
- Error completion: Turns ON at the END process of the scan in which the instruction is completed, and turns OFF at the next END process. (The error code is stored to the control data ((①) [0]): completion status) at the error completion.)
(3) The number of blocks used for the instruction is specified by the number of data to be written. (Refer to Section 10.1)
\begin{tabular}{l}
\begin{tabular}{c} 
Number of blocks used for the instruction \\
\begin{tabular}{c} 
Write points to be specified in the \\
instruction
\end{tabular} \\
\hline 1 to 4 \\
\hline 5 to 20 \\
instruction
\end{tabular} \\
\hline 21 to 36 \\
\hline 37 to 52 \\
\hline 53 to 68 \\
\hline 69 to 84 \\
\hline 85 to 100
\end{tabular} \begin{tabular}{c}
\hline
\end{tabular}
(4) When empty blocks are not available in the multiple CPU high speed transmission area, the instruction completes with an error. This error completion is prevented by setting the number of blocks used for the instruction to the special registers (SD796 to SD799), and using the special relays (SM796 to SM799) as the interlocks. (Refer to Section 10.1)

\section*{O Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
(1) When the specified other CPU is wrong, or when the settings are not set correctly to use the multiple CPU high speed transmission instruction.
(Error code: 4350)
- A reserved CPU is specified.
- A CPU number that is not mounted is specified.
- The value, start I/O number of the other CPU divided by 16 n , is outside the range of 3 EOH to 3E3H.
- This instruction is executed when "Multiple CPU High Speed Transmission Area Setting is valid." is selected.
- This instruction is executed in Q02UCPU.
- The host CPU is specified.
- A CPU in which the instruction cannot be executed is specified.
(2) When this instruction cannot be executed in the CPU.
(Error code: 4351)
- The instruction is not supported by the other CPU.
(3) The number of devices specified is incorrect.
(Error code: 4352)
(4) A device that cannot be used is specified.
(Error code: 4353)
(5) A device is specified with the character string that is not applicable.
(Error code: 4354)
(6) Number of data to be written (①] [1]) is outside the range of 0 to 100.
(Error code: 4355)

In any of the following cases, an error completion occurs, and the error code is stored to the device specified for the completion status storing device (①) [0]).
(1) The amount of instruction requests to a target CPU is exceeding the allowable value. (Empty blocks are not available in the multiple CPU high speed transmission area.)
(Error code: 0010H)
(2) The device of the other CPU specified for (d1) is a device that cannot be used for the other CPU, or the device is out of the range.
(Error code: 1001H)
(3) The number of data to be written set for the \(D(P)\) DDWR instruction is 0 .
(Error code: 1080H)
(4) The response of the instruction cannot be returned from the other CPU module. (Empty blocks are not available in the multiple CPU high speed transmission area.)
(Error code: 1003H)

\section*{\(\square\) Program Example}

In the following program, 10 words of data from D0 in the host CPU are written to W10 and the following devices in the CPU No. 2 when XO turns ON.
[Structured ladder/FBD]


The number of write data '10' is stored to the write data points storing device of control data D101 (S1)[1]).

Data from D0 to D9 in the host CPU are stored to W10 to W19 in the CPU No. 2.
[ST]
MOVP(X0,10,D101);
D_DDWR(X0,H3E1,D100,D0,W10,M100);
IF M100=TRUE THEN
IF M101=FALSE THEN Y10:=TRUE;
ELSE
Y11:=TRUE;
END_IF;
END_IF;

\section*{Caution}
(1) Digit specification of bit devices can be set for \(n\), © \({ }^{2}\) ) and ©11 . However, the following conditions need to be satisfied when the digit specification of bit devices is set for (s2) and (41).
- Digit specification of bit devices (K4/16 bits)
- The start number of bit device is a multiple of \(16(10 \mathrm{H})\).
(2) Execute this instruction as the write target CPU is powered ON.

If the write target CPU is not powered ON and this instruction is executed, the instruction is not processed.
(3) After the execution of this instruction, data stored by the system (completion status, completion device) cannot be stored normally, if the range of the device specified for the setting data is changed before the completion device turns ON.
(4) Devices SB, SW, SM, and SD contain the system information area.

When writing data to devices described above using the multiple CPU high speed transmission dedicated instruction \(D(P)\) _DDWR, be careful not to destroy the system information.

\subsection*{10.3 Reading Device Data from Other CPUs}

- Universal model QCPU: Supported if first 5 digits of the serial number are "10012" or later - Q00UJCPU, Q00UCPU, Q01UCPU, and Q02UCPU: Not supported

\section*{D(P)_DDRD}



Input argument,
\begin{tabular}{ll} 
& \\
& \(\mathrm{s} 1:\) \\
& \(\mathrm{s} 2:\) \\
& \\
& \\
&
\end{tabular}
*1: When file registers \((R, Z R)\) are specified, data that are outside the range in the host CPU can be read to devices of the other CPUs.
*2. By specifying the start device as a character string " ", data that are outside the range in the host CPU can be read to devices of the other CPUs.
*3: Device that indexed cannot be specified. (Example: DOZO, etc.)
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting data} & \multicolumn{2}{|l|}{Internal device} & \multirow[b]{2}{*}{R, RZ} & \multicolumn{2}{|c|}{J等:} & \multirow[b]{2}{*}{} & \multirow[b]{2}{*}{Zn} & \multicolumn{2}{|l|}{Constant} & \multirow[b]{2}{*}{Others} \\
\hline & Bit & Word \({ }^{\text {7 }}\) & & Bit & Word & & & K, H & \$ & \\
\hline \(\mathrm{n}^{* 4}\) & - & \(\bigcirc\) & \(\bigcirc\) & & & - & & \(\bigcirc\) & - & - \\
\hline (s1) \({ }^{5}\) & - & \(\triangle^{* 6}\) & \(\triangle^{*} 6\) & & & - & & - & - & - \\
\hline (52) \({ }^{5}\) & - & \(\bigcirc\) & \(\bigcirc\) & & & - & & - & - & - \\
\hline (d1) \({ }^{5}\) & - & \(\bigcirc\) & \(\bigcirc\) & & & - & & - & - & - \\
\hline (d2) 5 & \(\triangle^{* 8}\) & - & \(\triangle{ }^{*}\) & & & - & & - & - & - \\
\hline
\end{tabular}
*4: Indexes cannot be set to the setting data \(n\).
*5: Indexes can be set to the setting data s1 to d2.
*6: Local devices and file registers per program cannot be used as setting data.
*7: FD cannot be used.
*8: FX and FY cannot be used.

\section*{Control Data}
\begin{tabular}{c|l|l|l|c}
\hline Device & \multicolumn{1}{|c|}{ Item } & \multicolumn{1}{|c|}{ Setting data } & Setting range & Setting side \\
\hline (51) \([0]\) & Completion status & \begin{tabular}{l} 
Execution result of the instruction completion is stored. \\
\(0000(H) \quad:\) No error (normal completion) \\
Other than 0000(H) : Error code (error completion)
\end{tabular} & - & System \\
\hline (11) \([1]\) & Number of data to be read & Set the number of data to be read in unit of word. & 1 to 100 & User \\
\hline
\end{tabular}

\section*{Function}
(1) Stores data read from the device (s2) and the following devices specified in the CPU No. n for the number of points specified for (51) [1], to the device (d1) and the following devices specified in the host CPU at the time of multiple CPU system configuration.

(2) The status of the \(D(P)\) _DDRD instruction completion can be checked by the completion device ( (d2 [0]) and completion status display device (d2) [1]).
(a) Completion device ( (d2) [0])

Turns ON at the END process of the scan in which the instruction is completed, and turns OFF at the next END process.
(b) Completion status display (©2) [1])

Turns ON/OFF by the status of the instruction completion.
- Normal completion: OFF
- Error completion: Turns ON at the END process of the scan in which the instruction is completed, and turns OFF at the next END process. (The error code is stored to the control data ((©22 [0]): completion status) at the error completion.)
(3) The number of blocks used for the instruction is specified by the number of data to be read. (Refer to Section 10.1)

Number of blocks used for the instruction
\begin{tabular}{c|c|}
\hline \begin{tabular}{c} 
Read points to be specified in the \\
instruction
\end{tabular} & \begin{tabular}{c}
\(D(P) \_D D R D\) \\
instruction
\end{tabular} \\
\hline 1 to 100 & 1 \\
\hline
\end{tabular}
(4) When empty blocks are not available in the multiple CPU high speed transmission area, the instruction completes with an error. This error completion is prevented by setting the number of blocks used for the instruction to the special registers (SD796 to SD799), and using the special relays (SM796 to SM799) as the interlocks. (Refer to Section 10.1)

\section*{Operation Error}

In any of the following cases, an operation error occurs, the error flag (SMO) is turned ON, and the corresponding error code is stored to SD0.
(1) When the specified other CPU is wrong, or when the settings are not set correctly to use the multiple CPU high speed transmission instruction.
(Error code: 4350)
- A reserved CPU is specified.
- A CPU number that is not mounted is specified.
- The value specified for n , start I/O number of other CPU divided by 16 , is outside the range of 3 EOH to 3 E 3 H .
- This instruction is executed when "Multiple CPU High Speed Transmission Area Setting is not valid" is selected.
- This instruction is executed in the Q02UCPU.
- The host CPU is specified.
- A CPU in which the instruction cannot be executed is specified.
(2) When this instruction cannot be executed in the CPU
(Error code: 4351)
- The instruction is not supported by the other CPU.
(3) The number of devices specified is incorrect.
(Error code: 4352)
(4) A device that cannot be used is specified.
(Error code: 4353)
(5) A device is specified with the character string that is not applicable.
(Error code: 4354)
(6) Number of data to be read (d2 [1]) is outside the range of 0 to 100.
(Error code: 4355)

In any of the following cases, an error completion occurs, and the error code is stored to the device specified for the completion status storing device ( (d2) [0]).
(1) The amount of instruction requests to a target CPU is exceeding the allowable value. (Empty blocks are not available in the multiple CPU high speed transmission area.)
(Error code: 0010h)
(2) The device of the other CPU specified for (d1) is a device that cannot be used for the other CPU, or the device is out of the range.
(Error code: 1001H)
(3) The number of data to be read set for the \(D(P)\) _DDRD instruction is 0 .
(Error code: 1081H)
(4) The response of the instruction cannot be returned from the other CPU module. (Empty blocks are not available in the multiple CPU high speed transmission area.)
(Error code: 1003H)

\section*{\(\square\) Program Example}

In the following program, 10 words of data from D0 in the CPU No. 2 are read and stored to W10 and the following devices in the host CPU when X0 turns ON.
[Structured ladder/FBD]


\section*{[ST]}

MOVP(X0,10,D101);
D_DDRD(X0,H3E1,D100,D0,W10,M100);
IF M100=TRUE THEN
IF M101=FALSE THEN
Y10:=TRUE;
ELSE
Y11:=TRUE;
END_IF;
END_IF;

The number of read data '10' is stored to the read data points storing device of control data D101 (S1)[1]).

Data from D0 to D9 in the CPU No. 2 are stored to W10 to W19 in the host CPU.
(1) Digit specification of bit devices can be set for \(n\), © 22 and (d11). However, the following conditions need to be satisfied when the digit specification of bit devices is set for © \({ }^{2} 2\) and (d1).
- Digit specification of bit devices (K4/16 bits)
- The start number of bit device is a multiple of \(16(10 \mathrm{H})\).
(2) Execute this instruction as the read target CPU is powered ON.

If the read target CPU is not powered ON and this instruction is executed, the instruction is not processed.
(3) After the execution of this instruction, data stored by the system (completion status, completion device) cannot be stored normally, if the range of the device specified for the setting data is changed before the completion device turns ON.

MEMO

\section*{REDUNDANT SYSTEM INSTRUCTIONS (FOR REDUNDANT CPU)}

\footnotetext{
11.1 System switching instruction
}

\subsection*{11.1 System switching instruction}

\section*{SP_CONTSW}


\section*{SP_CONTSW}

\begin{tabular}{lll} 
Input argument & EN: & Execution condition \\
& \(\mathrm{s}:\) & \begin{tabular}{l} 
Value other than 0 and used to identify the processing that :ANY16 \\
issued the system switching request
\end{tabular} \\
Output argument, & ENO: & \begin{tabular}{l} 
Execution result \\
\(\mathrm{d}:\)
\end{tabular} \\
& Error completion device number & :Bit
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Setting Data} & \multicolumn{2}{|l|}{Internal Device} & \multirow[b]{2}{*}{R, ZR} & \multicolumn{2}{|c|}{J...1)} & \multirow[b]{2}{*}{U:...|c:...} & \multirow[t]{2}{*}{Zn} & \multirow[t]{2}{*}{\[
\begin{gathered}
\text { Constant } \\
\text { K, H }
\end{gathered}
\]} & \multirow[t]{2}{*}{Other} \\
\hline & Bit & Word & & Bit & Word & & & & \\
\hline (5) & - & \multicolumn{2}{|c|}{\(\bigcirc\)} & \multicolumn{4}{|c|}{-} & \(\bigcirc\) & - \\
\hline (d) & \(\bigcirc\) & \multicolumn{2}{|c|}{**} & \multicolumn{4}{|c|}{-} & - & - \\
\hline
\end{tabular}

\section*{Function}
(1) Switches between the control system and standby system at the END processing of the scan executed with the SP_CONTSW instruction.
(2) When using the SP_CONTSW instruction for system switching, the "manual switching enable flag (SM1592)" must have been turned on (enabled) in advance.
(3) ⓢ is provided to identify the processing block of the program where system switching occurred when multiple SP_CONTSW instructions are used.
At ©s, specify a value within the ranges -32768 to -1 and 1 to 32767 ( \(1_{H}\) to FFFF \(_{H}\) ).
The (s) value specified by the SP_CONTSW instruction is stored into the "system switching instruction argument (SD6)" of the error common information when the system switching is normally completed. *2
When multiple SP_CONTSW instructions are executed during the same scan, the argument of the SP_CONTSW instruction executed first is stored into the system switching instruction argument (SD6).
*2 : The value ©s specified for the SP_CONTSW instruction can be confirmed in the error common information of the PLC diagnostics dialog box on GX Works2.
(4) The © value specified by the SP_CONTSW instruction is stored into the "system switching instruction argument (SD1602)" of the new control system CPU module when system switching is normally completed. \({ }^{* 3}\)
By reading the SD1602 value from the new control system CPU module, which the SP_CONTSW instruction was used for system switching can be confirmed.
*3: The new control system CPU module means the CPU module that was switched from the standby system to the control system by the SP_CONTSW instruction.
(5) The error completion device is turned on by the control system CPU module when system switching by the SP_CONTSW instruction was unsuccessful.
(a) When OPERATION ERROR is detected due to any of the following reasons at the execution of the SP_CONTSW instruction, the error completion device is turned on during the instruction execution.
- 0 is specified at ©s of the executed SP_CONTSW instruction.
- The "manual switching enable flag (SM1592)" is off.
- The SP_CONTSW instruction was executed by the standby system in the separate mode.
- The SP_CONTSW instruction was executed in the debug mode.
(b) If systems could not be switched due to any of the reasons given in the following table, the error completion device turns on when system switching is executed in the END processing.
\begin{tabular}{c|l}
\multicolumn{1}{c|}{ Reason No. } & \multicolumn{1}{c}{ Reason for System Switching Failure } \\
\hline 0 & Normally completed \\
\hline 1 & Tracking cable is disconnected or faulty. \\
\hline 2 & \begin{tabular}{l} 
Hardware fault, power-off, reset or watchdog timer error occurred in the \\
standby system.
\end{tabular} \\
\hline 3 & Watchdog timer error occurred in the control system. \\
\hline 4 & Preparations being made for tracking transfer. \\
\hline 5 & Communication time-out. \\
\hline 6 & Stop error occurred in the standby system. (Excluding watchdog timer error) \\
\hline 7 & Operating status different between the control system and standby system. \\
\hline 8 & Memory copy being executed from the control system to the standby system. \\
\hline 9 & Write during RUN being executed. \\
\hline 10 & Network fault detected by the standby system. \\
\hline
\end{tabular}

When the error completion device was turned on due to unsuccessful system switching, 16 is stored into the "reason(s) for system switching (SD1588)" and the reason No. of the above table into the "reason(s) for system switching failure (SD1589)".
(6) Use a user program or GX Works2 to turn off the error completion bit that has turned on. If normal system switching is performed by the execution of the SP_CONTSW instruction with the error completion device on, the error completion device of the new standby system CPU module is also turned off.
When system switching is performed due to a factor other than the SP_CONTSW instruction, however, the error completion device is not turned off.

\section*{Operation Error}
(1) In any of the following cases, an operation error occurs, the error flag (SMO) turns on, and an error code is stored into SD0.
- The value specified at © is 0 at execution of the SP_CONTSW instruction.
(Error code: 4110)
- The manual switching enable flag (SM1592) is off (disabled) at the execution of the SP_CONTSW instruction.
(Error code: 4120)
- The SP_CONTSW instruction was executed by the standby system CPU module in the separate mode.
(Error code: 4121)
- The SP_CONTSW instruction was executed in the debug mode. (Error code: 4121)
(2) If system switching was unsuccessful, the error flag (SM0) is turned on and an error code is stored into SD0.
- The tracking cable is disconnected or faulty.
(Error code: 6220)
- Hardware fault, power-off, reset or watchdog timer error occurred in the standby system.
(Error code: 6220)
- Watchdog timer error occurred in the control system.
(Error code: 6220)
- Preparations are being made for tracking transfer.
(Error code: 6220)
- Communication time-out occurred.
(Error code: 6220)
- A stop error, excluding watchdog timer error, occurred in the standby system.
(Error code: 6220)
- The operating status differs between the control system and standby system.
(Error code: 6220)
- Memory copy is being executed from the control system to the standby system.
(Error code: 6220)
- Write during RUN is being executed.
(Error code: 6220)
- Network fault was detected by the standby system.
(Error code: 6220)

\section*{\(\triangle\) Program Example}
(1) The following program executes system switching on the leading edge of the system switching command (M100). If the system switching command (M100) remains on, the SP_CONTSW instruction is also executed by the new control system CPU module after system switching. Therefore, M101 is added to the execution conditions as a consecutive switching prevention flag.
[Structured ladder/FBD]

[ST]
SET(LDP( TRUE , SM1518 ) , M101 );
IF M100 AND SM1515 AND NOT(1516) AND NOT(M101) THEN SP_CONTSW( TRUE , K1 , M105 );
END_IF;
OUT( M105, M100 );
RST( M105 , M105 );

MEMO
Appendix 1 Added/Changed Instructions with Version Upgrade of GX Works2 . . . App-2

\section*{Appendix 1 Added/Changed Instructions with Version Upgrade of GX Works2}

The following table shows instructions added/changed with version upgrade of GX Works2.

Table App. 1-1 Added/changed instructions
\begin{tabular}{|c|c|c|c|}
\hline Version & Instruction name & Addition/change & Reference \\
\hline \multirow{12}{*}{Version 1.15R} & LDP & \multirow{6}{*}{- Symbolic description is added for structured ladder.} & \multirow{6}{*}{Section 5.1.2} \\
\hline & LDF & & \\
\hline & ANDP & & \\
\hline & ANDF & & \\
\hline & ORP & & \\
\hline & ORF & & \\
\hline & LDPI & \multirow{6}{*}{- Added for structured ladder and ST.} & \multirow{6}{*}{Section 5.1.3} \\
\hline & LDFI & & \\
\hline & ANDPI & & \\
\hline & ANDFI & & \\
\hline & ORPI & & \\
\hline & ORFI & & \\
\hline Version 1.24A & UMSG & - Added for structured ladder and ST. & Section 7.18.20 \\
\hline \multirow{9}{*}{Version 1.91V} & S_REFDVWRB & \multirow{8}{*}{- Added for structured ladder and ST.} & \multirow[t]{2}{*}{Section 8.3.1} \\
\hline & SP_REFDVWRB & & \\
\hline & S_REFDVWRW & & \multirow[b]{2}{*}{Section 8.3.2} \\
\hline & SP_REFDVWRW & & \\
\hline & S_REFDVRDB & & \multirow[t]{2}{*}{Section 8.3.3} \\
\hline & SP_REFDVRDB & & \\
\hline & S_REFDVRDW & & \multirow[b]{2}{*}{Section 8.3.4} \\
\hline & SP_REFDVRDW & & \\
\hline & SP_CONTSW & - Added for ST. & Section 11.1 \\
\hline Version 1.98C & SP_CONTSW & - Added for structured ladder. & Section 11.1 \\
\hline
\end{tabular}
[Symbols]
-(P) ..................................................................... 6-29
*(P)
6-33
/(P) ................................................................... 6-33
+(P) .................................................................. 6-29
\$+(P) ................................................................. 6-67
\$MOV(P) ......................................................... 6-123

\section*{[A]}

ACOS(P) ......................................................... 7-269
ACOSD(P)........................................................7-271
ADRSET(P)..................................................... 7-418
ANB.................................................................. 5-12
AND..................................................................... 5-2
AND(P)(F) ........................................................... 5-5
AND<.................................................................. 6-2
AND<= ................................................................. 6-2
AND<> ................................................................ 6-2
AND=................................................................... 6-2
AND>................................................................... 6-2
AND>= ................................................................. 6-2
AND\$<................................................................ 6-17
AND \(<=\)............................................................. 6-17
ANDS<> ............................................................. 6-17
AND\$=............................................................... 6-17
AND\$>................................................................ 6-17
AND\$>= ............................................................ 6-17
ANDD<................................................................ 6-5
ANDD<=............................................................... 6-5
ANDD<> ............................................................... 6-5
ANDD \(=\)................................................................ 6-5
ANDD> ................................................................ 6-5
ANDD>=............................................................... 6-5
ANDDT<........................................................... 7-376
ANDDT<=........................................................ 7-376
ANDDT<>......................................................... 7-376
ANDDT=...........................................................7-376
ANDDT>...........................................................7-376
ANDDT>=......................................................... 7-376
ANDE<................................................................ 6-9
ANDE<= .............................................................. 6-9
ANDE<> ............................................................... 6-9
ANDE \(=\)................................................................. 6-9
ANDE> ................................................................. 6-9
ANDE>= ............................................................... 6-9
ANDED<............................................................. 6-13
ANDED<= ...........................................................6-13
ANDED<> ...........................................................6-13
ANDED=.............................................................6-13
ANDED>............................................................ 6-13
ANDED>= ......................................................... 6-13

ANDPCHK.......................................................... 7-403
ANDTM< ..........................................................7-381
ANDTM<= ........................................................7-381
ANDTM<> ........................................................7-381
ANDTM= ..........................................................7-381
ANDTM> ..........................................................7-381
ANDTM>= ..........................................................7-381
ANI .....................................................................5-2
APPLICATION INSTRUCTIONS.............................7-1
Application Instructions tables .............................. 2-21
Arithmetic operation instructions table ..................2-13
ASC(P) ............................................................7-226
ASIN(P) ...........................................................7-265
ASIND(P).........................................................7-267
ATAN(P)...........................................................7-273
ATAND(P) .......................................................7-275
[B]
B-(P)...................................................................6-39
\(\mathrm{B}^{*}(\mathrm{P}) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .6-43 ~\)
B/(P) ..................................................................6-43
B+(P).................................................................6-39
BACOS(P)....................................................... 7-327
BAND(P).......................................................... 7-336
BASIC INSTRUCTIONS......................................... 6-1
Basic instructions tables.........................................2-8
BASIN(P).........................................................7-324
BATAN(P)........................................................ 7-330
BCD(P)..............................................................6-74
BCDDA(P).........................................................7-181
BCOS(P) .........................................................7-318
BDSQR(P).......................................................7-311
BIN(P)................................................................6-77
BINDA(P).........................................................7-171
BINHA(P).........................................................7-176
Bit processing instructions table...........................2-24
BK-(P)................................................................6-60
BK+(P)...............................................................6-60
BKAND(P) ...........................................................7-6
BKBCD (P) ......................................................... 6-106
BKBIN(P)..........................................................6-109
BKCMP(P)..........................................................6-21
BKCMP< ............................................................6-21
BKCMP<= .......................................................... 6-21
BKCMP<> ..........................................................6-21
BKCMP= ............................................................6-21
BKCMP> ............................................................ 6-21
BKCMP>= .......................................................... 6-21
BKOR(P) ............................................................7-12
BKRST(P)........................................................... 7-62
BKXNR(P) .........................................................7-24
BKXOR(P) ..... 7-18
BMOV(P) ..... 6-131
Bond instructions table ..... 2-6
BREAK(P) ..... 7-115
BRST(P) ..... 7-56
BSET(P) ..... 7-56
BSFL(P) ..... 7-44
BSFR(P) ..... 7-44
BSIN(P) ..... 7-315
BSQR(P) ..... 7-311
BTAN(P) ..... 7-321
BTOW(P) ..... 7-90
Buffer memory access instructions table ..... 2-28
BXCH(P) ..... 6-144
[C]
CALL(P) ..... 7-118
CHK ..... 7-163
CHKCIR ..... 7-167
CHKEND ..... 7-167
CHKST ..... 7-163
CJ ..... 6-149
Clock instructions table ..... 2-38
CML(P) ..... 6-126
COM ..... 7-122
Comparison operation instructions table ..... 2-8
COMRD(P) ..... 7-196
Configuration of Instructions ..... 3-1,3-2
Contact instructions table ..... 2-5
Correspondence between Generic Data Types and Devices ..... App-2
COS(P) ..... 7-257
COSD(P) ..... 7-259
[D]
D-(P) ..... 6-31
D(P)_DDRD ..... 10-16
D(P)_DDWR ..... 10-12,11-2
\(D^{*}(P)\) ..... 6-36
D/(P) ..... 6-36
D+(P) ..... 6-31
DABCD (P) ..... 7-193
DABIN(P) ..... 7-186
DAND(P) ..... 7-3
Data control instructions table ..... 2-36
Data conversion instructions table ..... 2-16
Data link instructions ..... 2-44
Data processing instructions table ..... 2-25
Data table operation instructions table ..... 2-27
Data transfer instructions table ..... 2-18
DATE-(P) ..... 7-369
DATE+(P) ..... 7-366
DATERD(P) ..... 7-360
DATEWR(P) ..... 7-363
DB-(P) ..... 6-41
DB* \({ }^{*}\) (P) ..... 6-45
DB/(P) ..... 6-45
DB+(P) ..... 6-41
DBAND(P) ..... 7-336
DBCD(P) ..... 6-74
DBCDDA(P) ..... 7-181
DBIN(P) ..... 6-77
DBINDA(P) ..... 7-171
DBINHA(P) ..... 7-176
DBK-(P) ..... 6-63
DBK+(P) ..... 6-63
DBKCMP< ..... 6-25
DBKCMP<= ..... 6-25
DBKCMP<> ..... 6-25
DBKCMP= ..... 6-25
DBKCMP> ..... 6-25
DBKCMP>= ..... 6-25
DBL(P) ..... 6-92
DCML(P) ..... 6-126
DDABCD(P) ..... 7-193
DDABIN(P) ..... 7-186
DDEC(P) ..... 6-72
Debug and error diagnostics instructions table ..... 2-29
DEC(P) ..... 6-69
DECO(P) ..... 7-71
DEG(P) ..... 7-281
DEGD(P) ..... 7-283
DELTA(P) ..... 5-49
Device range check ..... 3-5
DFLT(P) ..... 6-80
DFLTD(P) ..... 6-83
DFMOV(P) ..... 6-138
DFRO(P) ..... 7-144
DGBIN(P) ..... 6-98
DGRY(P) ..... 6-96
DHABIN(P) ..... 7-190
DI ..... 6-156
DINC(P) ..... 6-72
DINT(P) ..... 6-86
DINTD(P) ..... 6-89
DIS(P) ..... 7-81
Display instructions table ..... 2-28
DLIMIT(P) ..... 7-332
DMAX(P) ..... 7-94
DMIN(P) ..... 7-97
DMOV(P) ..... 6-116
DNEG(P) ..... 6-100
DOR (P) ..... 7-9
DRCL(P) ..... 7-38
DRCR(P) ..... 7-35
DROL(P) ..... 7-38
DROR(P) ..... 7-35
DSCL(P) ..... 7-344
DSCL2(P) ..... 7-348
DSER(P) ..... 7-64
DSFL(P) ..... 7-50
DSFR(P) ..... 7-50
DSORT ..... 7-100
DSTR(P) ..... 7-202
DSUM(P) ..... 7-68
DTEST(P) ..... 7-59
DTO (Data write to CPU shared memory of host CPU)7-148
DTO(P) ..... 7-148
DUTY ..... 7-408
DVAL(P) ..... 7-209
DWSUM(P) ..... 7-107
DXCH(P) ..... 6-141
DXNR(P) ..... 7-21
DXOR(P) ..... 7-15
DZONE(P) ..... 7-340
[E]
E-(P) ..... 6-48
\(\mathrm{E}^{*}(\mathrm{P})\) ..... 6-54
E/(P) ..... 6-54
E+(P) ..... 6-48
ECON(P) ..... 6-112
ED-(P) ..... 6-51
ED*(P) ..... 6-57
ED/(P) ..... 6-57
ED+(P) ..... 6-51
\(\operatorname{EDCON}(\mathrm{P})\) ..... 6-114
EDMOV(P) ..... 6-121
EDNEG(P) ..... 6-104
EGF ..... 5-21
EGP ..... 5-21
El ..... 6-156
EMOD(P) ..... 7-248
EMOV(P) ..... 6-119
ENCO(P) ..... 7-75
End instructions ..... 2-7
ENEG(P) ..... 6-102
EREXP(P) ..... 7-251
ESTR(P) ..... 7-214
EVAL(P) ..... 7-221
Executing Conditions for Instructions ..... 3-13
\(\operatorname{EXP}(\mathrm{P})\) ..... 7-295
EXPD(P) ..... 7-298
Extended clock instructions ..... 2-41
[F]
FDEL(P) ..... 7-141
FEND ..... 5-60
FF ..... 5-47
FIFR(P) ..... 7-135
FIFW(P) ..... 7-132
FINS(P) ..... 7-141
FLT(P) ..... 6-80
FLTD(P) ..... 6-83
FMOV(P) ..... 6-135
FOR ..... 7-112
FPOP(P) ..... 7-138
FROM(P) ..... 7-144
[G]
GBIN(P) ..... 6-98
GOEND ..... 6-154
GRY(P) ..... 6-96
[H]
HABIN(P) ..... 7-190
HEX(P) ..... 7-229
HOUR(P) ..... 7-374
How to Read Instruction Tables ..... 2-4
HOW TO READ INSTRUCTIONS ..... 4-1
[I]
I/O refresh instructions ..... 2-19
IMASK ..... 6-156
INC(P) ..... 6-69
INSTR(P) ..... 7-240
INSTRUCTION TABLES ..... 2-1
INT(P) ..... 6-86
INTD(P) ..... 6-89
INV ..... 5-17
IRET ..... 6-165
[J]
JMP ..... 6-149
[K]
KEY ..... 7-419
[L]
LD ..... 5-2
LD(P)(F) ..... 5-5
LD ..... 6-2
LD<= ..... 6-2
LD<> ..... 6-2
LD= ..... 6-2
LD> ..... 6-2
LD>= ..... 6-2
LD\$< ..... 6-17
MC ..... 5-55
LD\$<= 6-17 MCR ..... 5-55
LD\$<> ..... 6-17
MEAN(P) ..... 7-109
LDS= ..... 6-17
MEF ..... 5-19
LD\$> ..... 6-17
MEP ..... 5-19
LD\$>= 6-17 MIDR(P) ..... 7-235
LDD< ..... 6-5
LDD<= ..... 6-5
LDD<> ..... 6-5
LDD= ..... 6-5
LDD> ..... 6-5
LDD>= ..... 6-5
LDDT< ..... 7-376
LDDT<= ..... 7-376
LDDT<> ..... 7-376
LDDT= ..... 7-376
LDDT> ..... 7-376
LDDT>= ..... 7-376
LDE< ..... 6-9
LDE<= ..... 6-9
LDE<> ..... 6-9
LDE= ..... 6-9
LDE> ..... 6-9
LDE>= ..... 6-9
LDED< ..... 6-13
LDED<= ..... 6-13
LDED<> ..... 6-13
LDED= ..... 6-13
LDED> ..... 6-13
LDED>= ..... 6-13
LDI ..... 5-2
LDPCHK ..... 7-403
LDTM< ..... 7-381
LDTM<= ..... 7-381
LDTM<> ..... 7-381
LDTM= ..... 7-381
LDTM> ..... 7-381
LDTM>= ..... 7-381
LEDR ..... 7-160
LEFT(P) ..... 7-232
LEN(P) ..... 7-200
LIMIT(P) ..... 7-332
LOG(P) ..... 7-301
LOG10(P) ..... 7-305
LOG10D(P) ..... 7-307
LOGD(P) ..... 7-303
Logical operation instructions table ..... 2-21
[M]
Master control instructions ..... 2-7
MAX(P) ..... 7-94
MIDW(P) ..... 7-235
\(\operatorname{MIN}(P)\) ..... 7-97
MOV(P) ..... 6-116
MPP ..... 5-14
MPS ..... 5-14
MRD ..... 5-14
MTR ..... 6-193
Multiple CPU dedicated instruction ..... 2-45,9-1
MULTIPLE CPU HIGH SPEED TRANSMISSIONDEDICATED INSTRUCTIONS2-46,10-1
[ N ]
NDIS(P) ..... 7-85
NEG(P) ..... 6-100
NEXT. ..... 7-112
NUNI(P) ..... 7-85
[ O ]
OR ..... 5-2
OR(P)(F) ..... 5-5
OR< ..... 6-2
OR<= ..... 6-2
OR<> ..... 6-2
OR= ..... 6-2
OR> ..... 6-2
OR>= ..... 6-2
OR\$< ..... 6-17
OR\$<= ..... 6-17
ORS<> ..... 6-17
OR\$= ..... 6-17
OR\$> ..... 6-17
OR\$>= ..... 6-17
ORB ..... 5-12
ORD< ..... 6-5
ORD<= ..... 6-5
ORD<> ..... 6-5
ORD= ..... 6-5
ORD> ..... 6-5
ORD>= ..... 6-5
ORDT< ..... 7-376
ORDT<= ..... 7-376
ORDT<> ..... 7-376
ORDT= ..... 7-376
ORDT> ..... 7-376
ORDT>= ..... 7-376
ORE< ..... 6-9
ORE<= ..... 6-9
ORE<> ..... 6-9
ORE= ..... 6-9
ORE> ..... 6-9
ORE>= ..... 6-9
ORED< ..... 6-13
ORED<= ..... 6-13
ORED<> ..... 6-13
ORED= ..... 6-13
ORED> ..... 6-13
ORED>= ..... 6-13
OR ..... 5-2
ORPCHK ..... 7-403
ORTM< ..... 7-381
ORTM<= ..... 7-381
ORTM<> ..... 7-381
ORTM= ..... 7-381
ORTM> ..... 7-381
ORTM>= ..... 7-381
Other convenient instructions table ..... 2-20
Other instructions ..... 2-7
Other instructions table ..... 2-42
OUT ..... 5-24
OUT_C ..... 5-30
OUTH_T ..... 5-26
Output instructions ..... 2-6
OUT T ..... 5-26
[P]
PLF ..... 5-44
PLOADP ..... 7-471
PLOW(P) ..... 7-401
PLS ..... 5-44
PLSY ..... 6-189
POFF(P) ..... 7-397
POW(P) ..... 7-285
POWD(P) ..... 7-288
PR ..... 7-152
PRC ..... 7-156
Precautions on Programming ..... 3-4
Program branch instructions table ..... 2-19
Program control instructions table ..... 2-41
Program execution control instructions table ..... 2-19
PSCAN(P) ..... 7-399
PSTOP(P) ..... 7-395
PSWAPP ..... 7-478
PUNLOADP ..... 7-475
PWM ..... 6-191
[Q]
QCDSET(P) ..... 7-357
QDRSET(P) ..... 7-354

\section*{[R]}
RAD (P) ..... 7-277
RADD(P) ..... 7-279
RAMP ..... 6-184
RBMOV(P) ..... 7-481
RCL(P) ..... 7-31
RCR(P) ..... 7-27
RET ..... 7-120
RFS(P) ..... 6-168
RIGHT(P) ..... 7-232
RND(P) ..... 7-309
ROL(P) ..... 7-31
ROR(P) ..... 7-27
Rotation instructions table ..... 2-22
ROTC ..... 6-181
RSET(P) ..... 7-351
RST ..... 5-38
[S]
S(P)_DATE- ..... 7-392
S(P)_DATE+ ..... 7-389
S(P)_DATERD ..... 7-386
S(P)_DEVLD ..... 7-469
S(P)_RTREAD ..... 8-7
S(P)_RTWRITE ..... 8-9,8-12
S(P)_TO ..... 9-4
S(P)_ZCOM_J(U) ..... 8-2
SCJ ..... 6-149
SCL(P) ..... 7-344
SCL2(P) ..... 7-348
SECOND(P) ..... 7-372
SEG(P) ..... 7-78
SEQUENCE INSTRUCTIONS ..... 5-1
Sequence Instructions tables ..... 2-5
SER(P) ..... 7-64
SET ..... 5-35
SFL(P) ..... 7-41
SFR(P) ..... 7-41
SFT(P) ..... 5-51
SFTBL(P) ..... 7-47
SFTBR(P) ..... 7-47
SFTWL(P) ..... 7-53
SFTWR(P) ..... 7-53
Shift instructions ..... 2-7
Shift instructions table ..... 2-23
SIN(P) ..... 7-253
SIND(P) ..... 7-255
SORT ..... 7-100
SPD ..... 6-187
SP_DEVST ..... 7-466
Special function instructions table ..... 2-33
SP_FREAD ..... 7-452
SP_FWRITE ..... 7-440
SQR(P) ..... 7-291
SQRD (P) ..... 7-293
SRND(P) ..... 7-309
STMR ..... 6-178
STOP ..... 5-62
STR(P) ..... 7-202
STRDEL(P) ..... 7-246
String processing instructions table ..... 2-30
STRINS(P) ..... 7-243
Structured instructions table. ..... 2-27
SUM(P) ..... 7-68
SWAP(P) ..... 6-147
Switch instructions table. ..... 2-38
[T]
TAN(P) ..... 7-261
TAND (P) ..... 7-263
TEST(P) ..... 7-59
TIMCHK ..... 7-410
TO(P) ..... 7-148
TRACE ..... 7-437
TRACER ..... 7-437
TTMR ..... 6-176
TYPERD(P) ..... 7-432
Types of Instructions ..... 2-2
[U]
UDCNT1 ..... 6-170
UDCNT2 ..... 6-173
UMSG ..... 7-486
UNI(P) ..... 7-83
UNIRD(P) ..... 7-426
[V]
VAL(P). ..... 7-209
[W]
WAND(P) ..... 7-3
WDT(P) ..... 7-406
WOR (P) ..... 7-9
WORD(P) ..... 6-94
WSUM(P) ..... 7-105
WTOB(P) ..... 7-90
WXNR(P) ..... 7-21
WXOR(P) ..... 7-15
[X]
XCH(P) ..... 6-141
[Z]ZONE(P)7-340
ZPOP(P) ..... 7-423
ZPUSH(P) ..... 7-423
ZRRDB(P) ..... 7-412
ZRWRB(P) ..... 7-415

\section*{MEMO}
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)
\(\qquad\)

\section*{WARRANTY}

Please confirm the following product warranty details before using this product.

\section*{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.

\section*{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.

\section*{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.

\section*{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 of damages caused by any cause found not to be the responsibility of Mitsubishi, loss in opportunity, lost profits incurred to the user by Failures of Mitsubishi products, special damages and secondary damages whether foreseeable or not, compensation for accidents, and compensation for damages to products other than Mitsubishi products, replacement by the user, maintenance of on-site equipment, start-up test run and other tasks.

\section*{5. Changes in product specifications}

The specifications given in the catalogs, manuals or technical documents are subject to change without prior notice.
```

Microsoft, Windows, Windows Vista, Windows NT, Windows XP, Windows Server, Visio, Excel, PowerPoint, Visual Basic, Visual C++, and Access are either registered trademarks or trademarks of Microsoft Corporation in the United States, Japan, and other countries.
Intel, Pentium, and Celeron are trademarks of Intel Corporation in the United States and other countries.
Ethernet is a registered trademark of Xerox Corp.
The SD and SDHC logos are either registered trademarks or trademarks of SD-3C, LLC.
All other company names and product names used in this manual are either trademarks or registered trademarks of their respective companies.

```
\begin{tabular}{|c|c|}
\hline MODEL & Q-KP-KM-E \\
\hline \begin{tabular}{c} 
MODEL \\
CODE
\end{tabular} & 13JW07 \\
\hline \multicolumn{2}{|c|}{ SH(NA)-080783ENG-N(1309)KWIX } \\
\hline
\end{tabular}```


[^0]:    *1: Refer to the User's Manual (Hardware Design, Maintenance and Inspection) of the CPU module used.

