KEYENCE

B89GB

Programable Controller
KV-8000/7000/5000/3000/1000 Series
KV Nano Series

Script Programming Manual

Please read this Manual before use.
Keep this Manual in a safe place for later reference.

Chapter 1 OVERVIEW
Chapter 2 PROGRAMMING SCRIPT
Chapter 3 DATA PROCESSING
Chapter 4 SCRIPT SYNTAX AND PRECAUTIONS
Chapter 5 BASIC FUNCTIONS
Chapter 6 APPLICATION FUNCTIONS
Chapter 7 ARITHMETIC OPERATION FUNCTIONS
Chapter 8 EXTENDED FUNCTIONS

· List of Control Relays/ Control Memories
· Index

Applicable CPU Unit
•KV-8000
•KV-7500
•KV-7300
•KV-5500
•KV-5000
•KV-3000
•KV-1000
•KV Nano Series

Frequently Asked Questions
FAQ
www.keyence.com/controlfaq

Preface

The manual describes functions required to compile KV-8000/7500/7300/5000/3000/1000 and KV Nano Series script program and setting method. Carefully read and fully understand the manual before compiling program.
During programming, about the command of CPU internal function and special command of expansion unit, please see User's Manuals of every unit such as "KV-8000 Series User's Manual", "KV-7000 Series User's Manual", "KV-5500/5000/3000 Series User's Manual", "KV Nano Series User's Manual" "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual" and "KV STUDIO User's Manual", etc.
Store this Manual in a safe place so that you can retrieve it whenever necessary.
Please handover this manual to the end-users who are going to use the PLC Series.

■ KV-8000/7000/5000/3000/1000 Series and KV Nano Series related manuals

All the following PDF manuals can be found and opened in the help file of KV STUDIO. In addition, the latest version of PDF manuals can be downloaded from the Keyence web site.

Type	Description
KV-8000 Series User's Manual	This manual describes how to connect, and maintain the KV-8000 Series. It also describes the CPU built-in functions and how to create ladder programs.
KV-7000 Series User's Manual	This manual describes how to connect, and maintain the KV-7000 Series. It also describes the CPU built-in functions and how to create ladder programs.
KV-5500/5000/3000 Series User's Manual	This Manual describes the configuration/specification, CPU internal function, ladder programming method of KV-5500/5000/3000 Series.
KV Nano Series Base Unit (terminal block type) User's Manual	This manual describes system configurations and specifications, built-in functions in the base unit, and how to create ladder programs for the terminal block type KV Nano Series.
KV Nano Series Base Unit (connector type) User's Manual	This manual describes system configurations and specifications, built-in functions in the CPU unit, and how to create ladder programs for the connector type KV Nano Series.
KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual	This Manual describes various instructions available in Ladder program.
KV-8000/7000/5000/3000/1000 Series and KV Nano Series Script Programming Manual	This Manual describes the script programming method and available operators, control statements and functions, etc.
KV STUDIO User's Manual	This Manual describes how to operate KV STUDIO.
KV-5500/5000/3000/1000 Series System Macro Manual	This Manual describes the system macros available for the KV-5500/5000/3000/1000 Series PLC.
KV-1000 Introduction Manual	This Manual describes the basic functions and how to use the KV-1000 briefly.
KV-1000 Series User's Manual	This Manual describes the specifications, integrated functions and operating method of the KV-1000 PLC.
KV-1000 Series Programming Manual	This manual describes how to make ladder programs using the KV-1000 PLC, and modules/macros and local devices.

Safety Precautions

■ Symbols

This document contains notices which you should observe to ensure your own personal safety, as well as to protect the product and connected equipment. These notices are highlighted in the manual by a warning triangle and are marked as follows according to the level of danger:

Danger	**Failure to observe this item causes a risk that consequently results in death or serious injury.**
Warning	**Failure to observe the items mentioned may result in death or serious injury.**
Caution	**Failure to observe the items mentioned may result in medium or minor injury.**
NOTICE	**Failure to observe this item may cause damage to the product itself as well to other property.**

Important **Notes on operations that are always performed**

Point **Notes on operations that are often performed erroneously**

Reference Here are listed items further enriching the understanding of the main text, and the information useful to know.

Page or manual providing related information

■ General Precautions

Warning	• **Before using KV STUDIO, confirm whether the function and performance of KV-8000/7500/7300/5500/5000/3000/1000 or KV Nano Series are normal before use.** • **Verify that this device functions correctly before the start of work and when operating the device.** • **Provide a safety circuit that bypasses PLC to enable fail safe operation of the entire system in the event that the PLC fails.** • **Output circuit or internal circuit malfunctions sometimes prevent control from being performed normally.Be sure to provide a safety circuit in control systems where circuit malfunction may lead to fire or other serious accidents.** • **Do not use this product for the purpose to protect a human body or a part of human body.** • **This product is not intended for use as explosion-proof product. Do not use this product in a hazardous location and/or potentially explosive atmosphere.**
Caution	**If you use the product in a manner that falls outside of the ranges indicated in its specifications, protection provided by the product may be damaged.**
NOTICE	• **Proceed with care when modifying the device, or when using it in a manner that falls outside of the ranges indicated in its specifications, since KEYENCE is unable to guarantee device functionality or performance in such situations.** • **Use this product in combination with other devices only after careful consideration, since the product may fail to satisfy its functionality and performance capabilities as a result of the conditions and environment in which it is used.**

KV-8000 CPU Function Version

The functions of the KV-8000 Series differ depending on the CPU function version.

The CPU function version can be checked using "Tools" → "Check CPU Function Version" → "PLC" in the KV STUDIO menu (* KV STUDIO Ver.10.0 or later is required).

The CPU function version can be updated using "Tools" → "Check CPU Function Version" → "PLC" → "Update System Program" in the KV STUDIO menu. (* KV STUDIO Ver.10.0 or later is required.)

■Differences in CPU function depending on CPU function version

Release		2021/09	2021/07	2021/04	2020/04	2019/04
CPU function version		2.3	2.2	2.1	2.0	1.0
KV STUDIO compatible version		Ver.11.3 or later	Ver.11.2 or later	Ver.11.1 or later	Ver.11.0 or later	Ver.10.0 or later
CPU function	Device comment	Up to 128 characters	Up to 128 characters	Up to 128 characters	Up to 32 characters	Up to 32 characters
	Variable	○	○	○	○*1	–
	Array	Compatible with function block type	Compatible with STRING type	–	–	–
	Structure	○	○	○	○	–
	Function block	Compatible with automatic data type conversion	○	○	○(Compatible with EN/ENO, instances)	○
	ST Language	○	○	○	–	–
Function/ Script	Function block function	○	○	○	2 function added (FBCALL, FBSTRT)	Not added
	Data size acquisition function	○	○	○	2 function added (WSIZE, ACNT)	–

*1 Variables that can be used for communication with the VT5 series can be used in CPU function version 2.0 or later instead of labels supported in CPU function version 1.0.

KV-7500/7300 CPU Function Version

The KV-7000 Series functions differ according to the CPU function version.

The CPU function version can be checked using "Tools"->"Functional version validation" ->"PLC" in the KV STUDIO menu (* KV STUDIO Ver. 8 or later is required).

The CPU function version can be updated using "Tools"->"Functional version validation" -> "Update system program" in the KV STUDIO menu (* KV STUDIO Ver. 8.1 or later is required).

■ Differences in CPU function according to CPU function version

<table>
<tr><th colspan="2">Release date</th><th>Dec 2018</th><th>July 2017</th><th>June 2016</th><th>December 2015</th><th>June 2015</th><th>March 2015</th></tr>
<tr><td colspan="2">CPU function version</td><td>2.3</td><td>2.2</td><td>2.1</td><td>2.0</td><td>1.1</td><td>1.0</td></tr>
<tr><td colspan="2">KV STUDIO compatible version</td><td>9.45 or later</td><td>9.2 or later</td><td>9.1 or later</td><td>9.0 or later</td><td>8.1 or later</td><td>8.0 or later</td></tr>
<tr><td rowspan="2">Compatible CPU unit</td><td>KV-7500</td><td colspan="2">Yes</td><td>Yes</td><td>Yes</td><td>Yes</td><td>Yes</td></tr>
<tr><td>KV-7300</td><td colspan="2">Yes</td><td>Yes</td><td>Yes</td><td>Yes</td><td>Yes</td></tr>
<tr><td rowspan="3">Compatible expansion unit</td><td>KV-XLE02
KV-XL402/XL202</td><td colspan="2">Yes</td><td colspan="4">No</td></tr>
<tr><td>KV-XH16ML/XH04ML</td><td colspan="4">Yes</td><td colspan="2">No</td></tr>
<tr><td>KV-XH16EC</td><td>Yes</td><td colspan="5">No</td></tr>
<tr><td rowspan="10">CPU function</td><td>Project name</td><td colspan="4">Up to 64 characters</td><td colspan="2">Up to 32 characters</td></tr>
<tr><td>Program name</td><td colspan="4">Up to 64 characters</td><td colspan="2">Up to 24 characters</td></tr>
<tr><td>Program comment</td><td colspan="4">Up to 256 characters</td><td colspan="2">Up to 32 characters</td></tr>
<tr><td>OFF processing setting when program execution is stopped</td><td colspan="4">Yes</td><td colspan="2">No</td></tr>
<tr><td>Assignment for interrupt program of work area for script text string</td><td colspan="2">Yes</td><td colspan="4">No</td></tr>
<tr><td>Inter-unit synchronization</td><td colspan="5">Yes</td><td>No</td></tr>
<tr><td>Inter-unit synchronization read/write CPU device</td><td colspan="4">Yes</td><td colspan="2">No</td></tr>
<tr><td>KV socket communication</td><td colspan="5">Yes*1</td><td>No</td></tr>
<tr><td>Function block function</td><td colspan="2">Yes (compatible with unspecified unit argument types)</td><td colspan="2">Yes</td><td colspan="2">No</td></tr>
<tr><td>Application package</td><td colspan="3">Yes</td><td>No</td><td colspan="2">No</td></tr>
<tr><td rowspan="6">Function script</td><td>Function block function</td><td colspan="4">3 function added (FB,FEND,FUN)</td><td colspan="2">No additions</td></tr>
<tr><td>Unit program function</td><td colspan="4">5 function added (UMALLOC,UPSTRT, UPSTOP,UFSUS,UFRSM)</td><td colspan="2">No additions</td></tr>
<tr><td>Floating point function</td><td colspan="4">1 function added (ATAN2)</td><td colspan="2">No additions</td></tr>
<tr><td>Data control function</td><td colspan="4">Specifications of 1 function expanded (APR)*2</td><td colspan="2">No additions</td></tr>
<tr><td>Data processing function</td><td colspan="4">Specifications of 4 function expanded (MAX,MIN,AVG,WSUM)*3</td><td colspan="2">No additions</td></tr>
<tr><td>Unit type judgment function</td><td colspan="3">Yes</td><td colspan="3">No</td></tr>
</table>

*1 KV socket communication can only be used with the KV-7500.

*2 Double precision floating point type real numbers can now be specified in linear approximation function (APR). (Scaling)

*3 Single/double precision floating point type real numbers can now be specified in search max. value (MAX), search min. value (MIN), average (AVG), and word sum (WSUM) function.

KV Nano Series CPU Function Version

The KV Nano Series functions differ according to the CPU function version.

The CPU function version can be checked using "Tools"->"Functional version validation"->"PLC" in the KV STUDIO menu (* KV STUDIO Ver. 7 or later is required).

The CPU version can be updated using "Tools"->"Functional version validation" -> "Update system program" in the KV STUDIO menu (* KV STUDIO Ver. 7.1 or later is required).

■ Differences in CPU function according to CPU function version

KV-N14*/N24*/N40*/N60*

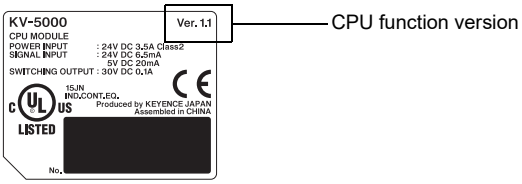
CPU function version	Release date	Compatible functions
Ver. 1.0	October 17, 2012	Initial version
Ver. 2.0	May 15, 2013	Compatible with EtherNet/IP unit KV-NC1EP

KV-NC32T

CPU function version	Release date	Compatible functions
Ver. 2.0	May 15, 2013	Initial version

KV-5500/5000/3000 CPU Function Version

The CPU function version has been added to the KV-5500/5000/3000 Series shipped after September 10, 2009. Check the serial label on the side for details on the CPU function version.



Differences in KV-5500/5000/3000 CPU function versions

Differences according to the CPU function version are listed below.

■ Differences in CPU functions according to CPU function version

Time of shipment		Following June 14, 2010	Following September 10, 2009	Following June 14, 2007
CPU function version		Ver.2.0	Ver.1.1	Ver.1.0
KV STUDIO compatible version		Ver. 6.0 or higher	Ver. 5.6 or higher	Ver. 4.0 or higher
Compatible CPU unit	KV-5500	○	-	-
	KV-5000	○	○	○
	KV-3000	○	○	○
CPU function	Bluetooth communication	○	○*1	×
	Battery-less operation	○	×	×
	Initial screen display (access window)	○	×	×
	Read folder specification (auto-loading function)	○	×	×
	SDHC card (memory card)	○	×	×
	Word device bit comment (KV STUDIO)	○	×	×
	Performance monitor (KV STUDIO)	○	×	×
	Differential monitor (KV STUDIO)	○	×	×
Function/script	Double precision floating point type real number	○	×	×
	Timer/counter function	1 function added (TMU)	No additions	-
	PID control function	1 function added (PIDAT)	No additions	-
	Data control function	3 functions added (RAMP,TPOUT,LLFLT)	No additions	-
	Move function	2 functions added (BYLMOV,BYBMOV) Specifications of 1 function expanded*2	No additions	-
	Arithmetic/comparison function	1 function added (POW)	No additions	-
	Data shift function	2 functions added (ASRA,ASLA)	No additions	-
	Data conversion function	4 functions added (ABS,BSWAP,CPMSET,CPMGET) Specifications of 2 functions expanded*3	No additions	-
	Floating point function	5 functions added (LOG10,DFLOAT,DINTG,FTODF,DFTOF)	No additions	-
	Simple operation function	Specifications of 2 functions expanded*4	No additions	-
	Text string processing function	5 functions added (SPLIT,STRIM,SFINDN,CPSASC,RCPSASC)*5 Specifications of 1 function expanded*6	No additions	-
	Data processing function	2 functions added (SORT,SORTN)	No additions	-
	Memory card function	7 functions added (MPRINT,MREADL,MCOPY,MMOV,MREN, MFREEK,MSTAT)	No additions	-
	Sensor setting function	3 functions added (SPRD,SPWR,SSVC)	No additions	-
	Refresh function	2 functions added (RFSCI,RFSCO)	No additions	-

*1 KV-3000 Ver.1.1 does not support Bluetooth communication.

*2 Constants can be specified for the PMOV operand S.

*3 Devices can be specified for the DECO/ENCO operand n.

*4 A signed integer can be specified for the shift function (CAL>>, CAL<<). (Arithmetic shift)

*5 Only scrip function is added for SPLIT.

*6 RCOM supports word device bit comment.

■ Differences in Ethernet functions according to CPU function version

Time of shipment		Following June 14, 2010	Following September 10, 2009	Following June 14, 2007
Compatible unit	KV-5500	Ver.2.0	-	-
	KV-5000	Ver.2.0	Ver.1.1	Ver.1.0
	Ethernet/IP unit	KV-EP21V	-	-
	Ethernet unit	KV-LE21V	KV-LE21V	KV-LE20V
Ethernet function	Simple PLC link	○	○	×
	FTP client	○	○	×
	BOOTP client	○	○	×
	Communication without unit setting	○	×	×
	APPE command support (FTP server function)	○	×	×

● Restrictions for using extended input/output unit (KV-B8XTD, KV-C16XTD, KV-C32XTD)

The following restrictions apply when connecting an extended input/output unit to the CPU function version 1.0 CPU unit. (These restrictions do not apply to the CPU function version 1.1 or higher CPU unit.)

- During the I/O test, 32DO is displayed on the KV-C16XTD and 64DO on the KV-C32XTD.
- During the I/O test, the details displayed for the KV-B8XTD differ according to the unit setting information stored in the CPU unit.
  - When the unit information stored in the CPU unit is the same as the actual unit configuration.
    - 32DO is displayed.
  - When the unit information stored in the CPU unit differs from the actual unit configuration.
    - 16DO is displayed.
- If the unit information stored in the CPU unit differs from the actual unit configuration, the I/O test operation will differ according to the type of unit to the right of the KV-B8XTD.
  - When the right unit is a special unit, or when there is no unit to the right.
    - The KV-B8XTD output relay is not controlled.
  - When the right unit is an output unit.
    - When the right unit's output relay turns ON, the KV-B8XTD output relay also turns ON.
  - When the right unit is an input unit.
    - When the right unit's input relay turns ON, the KV-B8XTD output relay also turns ON.
- The input refresh prohibit setting is invalid for the KV-B8XTD, KV-C16XTD and KV-C32XTD.
  If output prohibit is set, the input refresh will also be prohibited.
- If the input relay forced set assigned to the KV-B8XTD, KV-C16XTD, KV-C32XTD is canceled, the input status will be applied after the target input relay has turned OFF for one scan.

How This Manual Is Organized

Chapter 1	OVERVIEW	Describes features of KV scripts such as script application example (program example) and KV script type, operators and control statements, functions, etc.
Chapter 2	PROGRAMMING SCRIPT	Describes script making steps as well as error message and warning message during conversion.
Chapter 3	DATA PROCESSING	Describes script processing device, variable or constant range (type) and programming method, index modification and indirect specifying, etc.
Chapter 4	SCRIPT SYNTAX AND PRECAUTIONS	Describes how to use assignment statement and operators used in scripts as well as syntax of control statement, etc.
Chapter 5	BASIC FUNCTIONS	Describes contact functions, bit functions, bit output functions, output functions, and timer/counter functions, etc.
Chapter 6	APPLICATION FUNCTIONS	Describes program flow control functions, macro functions, Index register functions, and indirectly specifying functions, etc.
Chapter 7	ARITHMETIC OPERATION FUNCTIONS	Describes data more functions, arithmetic operation functions, logical operation functions, data shift functions, data conversion functions, floating real number functions, and text string processing functions, etc.
Chapter 8	EXTENDED FUNCTIONS	Describes data processing functions, table processing functions, clock processing functions, high speed processing functions, record functions, memory card functions, and access window functions, etc.
	APPENDICES	Presents character code table, a list of control relay/control memories, index, etc.

Contents

Safety Precautions **1**
How This Manual Is Organized **7**
Contents **8**

Chapter 1 OVERVIEW

KV Scripts **1-2**
Features of KV Script 1-2
How to Use KV Scripts 1-3
KV Script Type **1-8**
Box Script 1-8
Area Script 1-9
KV Script Function **1-10**
About Assignment Statement 1-10
About Operator 1-11
About Control Statement 1-12
About Suffix 1-14
About Script Function 1-17

Chapter 2 PROGRAMMING SCRIPT

PROGRAMMING SCRIPT **2-2**
Programming Script 2-2
Box Script **2-4**
Features of Box Script 2-4
Making Box Script 2-6
Precautions When Using Box Script 2-13
Area Script **2-16**
Features of Area Script 2-16
Making Area Script 2-18
Label Programming and Label Array **2-25**
Label Programming 2-25
Global and local variables 2-25
Variables and data formats at a glance 2-26
Array 2-27
Structure 2-29
Variable Setting 2-32
Precautions for using variables 2-35
- **Label Programming and Label Array** **2-37**
Label Programming 2-37
Label Array 2-46
Precautions when programming script **2-51**
About Parallel Connection of Box Script 2-51
Precautions When Using Script Programming Operation Formula 2-51

Script Conversion Output Message **2-52**
Warning Message 2-52
Error Message 2-54
Function block **2-58**
Calling Function Blocks 2-58
Programing Function Blocks Using Script 2-59
Macro **2-60**
Macro Call 2-60
Programming Macro by Using Script 2-61
Script Debug Function **2-62**
Monitor Function 2-62
Input Auxiliary Function and Other Info **2-64**
About Input Auxiliary Function 2-64
About Comments Function 2-69
Write Easy-reading Script 2-70
Inline Mnemonics 2-72
Precautions on Edit Screen and Customization 2-73
About Script Work Memory (Work Devices) 2-74
XYM Marking 2-77

Chapter 3 DATA PROCESSING

Devices **3-2**
Device List (When using KV-8000) 3-2
Device list (When using KV-7000) 3-6
Device table (for KV-5500 / 5000/3000/1000) 3-10
Device table (for KV Nano) 3-17
Device notation method 3-21
Bit Device 3-24
Word device 3-35
Processing Bit Device in Word Unit 3-45
Processing Word Device as Bit Device 3-46
Direct I/O Relay 3-47
Variables 3-49
Constant 3-51
Index Modification and Indirect Specifying **3-54**
Index modification 3-55
What is indirect specifying 3-57
Combination of index modification and indirect specifying 3-61
Data Type (suffix) **3-62**
Suffix 3-63
About Type Declaration (TYPE) 3-69
About Operation Type 3-70
PRAGMA Directive 3-72

Chapter 4 SCRIPT SYNTAX AND PRECAUTIONS

Assignment Statement 4-2
How to Use An Assignment Statement 4-2
Order of Priority low 4-6
Operator List 4-6
Control statement 4-10
Control Statement List 4-10
IF ~ Statement (Conditional Branch (1)) 4-12
IF ~ ELSE Statement (Conditional Branch (2)) 4-14
IF ~ ELSEIF Statement (Conditional Branch (3)) 4-16
SELECT CASE ~ Statement (Multi-branch control) 4-18
MC ~ MCR Statement (Master control) 4-22
FOR ~ NEXT Statement (specified loop control times) 4-24
WHILE Statement (pre-judge loop control) 4-27
DO ~ UNTIL Statement (Post Identification Loop Control) 4-29
BREAK Processing 4-31
Precautions on Using Control Statement 4-32
Precautions on Script Programming 4-38
Status of Hold Control Bit Device 4-38
Precautions for using the timer / counter 4-40

Chapter 5 BASIC FUNCTIONS

Contact function 5-2
LDP Rising edge differential of specified bit 5-2
LDF Falling edge differential of specified bit 5-4
LDPB Rising edge differential bar of specified bit 5-6
LDFB Falling edge differential bar of specified bit 5-7
Bit contact function 5-8
BLD Specified bit information of word data 5-8
BLDB Specified bit information bar of word data 5-10
Output function 5-12
SET Output ON of specified bit 5-12
RES Output OFF of specified bit 5-13
Bit output function 5-14
BOUT Specified bit output of word device 5-14
BOUB Specified bit inverted output of word device 5-16
BSET Specified bit output ON of word device 5-18
BRES Specified bit output OFF of word device 5-20
Timer / Counter function 5-22
TMR 100 ms 32-bit down timer 5-22
TMH 10 ms 32-bit down timer 5-26
TMS 1 ms 32-bit down timer 5-28
TMU 10µs 32-bit down timer 5-30
CNT 32 bit up counter 5-32
Direct I/O functions 5-36
RFSX Input refresh 5-36
RFSY Output refresh 5-37

Chapter 6 APPLICATION FUNCTION

Flow Functions **6-2**
- CALL — Start specified sub-routine 6-2
- ECALL — Module sub-routine instruction 6-4

Macro function **6-6**
- MCALL — Start sub-routine type macro 6-6
- MSTRT — Start self-hold type macros 6-7
- MEND — End self-hold type macros 6-8

Function block function **6-10**
- FB — Start function block 6-10
- FBCALL — Function block call 6-11
- FBSTRT — Function block start 6-13
- FEND — End function block 6-15
- FUN — Start function 6-16

Module function **6-18**
- MDSTRT — Module execution start 6-18
- MDSTOP — Module execution stop 6-19

Index register function **6-20**
- ZPUSH — Batch purge of index register 6-20
- ZPOP — Batch restore of index registers 6-24

Specifying functions indirectly **6-26**
- ADRSET — Address storage of indirectly specified target 6-26
- ADRINC — Add 1 to stored address 6-28
- ADRDEC — Decrement stored address by 1 6-30
- ADRADD — Add stored address 6-32
- ADRSUB — Subtract stored address 6-34

File register function **6-36**
- FRSET — Switch file register mode 6-36
- FRSTM — Batch-store file register 6-38
- FRLDM — Batch- read file register 6-40

Unit program function **6-42**
- UPSTRT — Unit program start 6-42
- UPSTOP — Unit program forced stop 6-44
- UFSUS — Suspend flow 6-46
- UFRSM — Resume flow 6-48
- UMALLOC — Allocate buffer memory 6-50

Buffer memory function **6-52**
- UREAD — Read buffer 6-52
- UWRIT — Write buffer 6-54
- UFILL — Insert same value in data string of buffer 6-56

Unit type judgment function **6-58**
- IS_ □□□ — Unit type judgment 6-58

Array element number acquisition function **6-60**
- ACNT — Get the number of array elements 6-60

Word size get function **6-62**
- WSIZE — Get word size 6-62

Chapter 7 ARITHMETIC OPERATION FUNCTIONS

Data move function.....7-2

BMOV Batch data move of word block.....7-2
FMOV Substitute same value in data string.....7-4
PMOV Move continuous bit information to a specified position 7-6
BYBMOV Move continuous byte string information to specified position (H -> L).....7-10
BYLMOV Move continuous byte string information to specified position (L -> H).....7-12

Arithmetic Operation Functions 7-14

INC Increament by 1 7-14
DEC Decrement by 1 7-16
ROOT Square root.....7-18
SQRT Square root.....7-20

Logic Operation Functions 7-22

ANDA Logical AND operation.....7-22
ORA Logical OR operation.....7-24
EORA Exclusive OR operation 7-26
ENRA Exclusive NOR.....7-28
COM Bit inversion 7-30
NEG Invert sign 7-32
ABS Absolute value 7-36

Data Shift Functions 7-38

SRA Shift right (bit unit).....7-38
SLA Shift left (bit unit).....7-40
ASRA Arithmetic right shift A (bit unit).....7-42
ASLA Arithmetic left shift (bit unit) 7-44
RRA Right rotate (bit unit, with carry).....7-46
RLA Left rotate (bit unit, with carry) 7-48
RRNCA Right rotate (bit unit, without carry).....7-50
RLNCA Left rotate (bit unit, without carry) 7-52
WSR Shift word device right 7-54
WSL Shift word device left.....7-56
BSR Shift bit device right 7-58
BSL Shift bit device left.....7-60

Data Control Functions 7-62

LIMIT Upper/lower limit value control.....7-62
BANDC Dead band control 7-64
ZONE Zone control.....7-66
APR Linear approximation (Scaling).....7-68
RAMP Ramp signal.....7-74
TPOUT Time proportional output.....7-76
LLFLT Lead-lag filter.....7-78

Data Conversion Functions 7-80
TBCD Convert BIN data to BCD data.....7-80
TBIN Convert BCD data to BIN data.....7-82
MPX 4-bit data decode.....7-84
DMX 4-bit data encode.....7-86
GRY Convert BIN data to gray code 7-88
RGRY Convert gray code to BIN data 7-90
DISN Disperse nibble.....7-92
UNIN Unite nibble.....7-94
DISB Disperse byte.....7-96
UNIB Unite byte.....7-98
SWAP High lower byte conversion.....7-100
BSWAP Block swap.....7-102
DECO 8-bit decode.....7-104
ENCO 8-bit encode.....7-106
TOU Convert data type to .U.....7-108
TOS Convert data type to .S.....7-108
TOD Convert data type to .D.....7-108
TOL Convert data type to .L 7-108
CPMSET CIP message creation 7-110
CPMGET CIP message retrieval 7-112
Floating Point Functions.....7-114
FLOAT Convert BIN data to single precision floating point type real number data 7-114
TOF Convert BIN data to single precision floating point type real number data 7-116
DFLOAT Convert BIN data to double precision floating point type real number data ...7-118
TODF Convert BIN data to double precision floating point type real number data ...7-120
INTG Convert single/double precision floating point type real number data to BIN data...7-122
INT Convert single precision floating point type real number data to BIN data 7-124
DISF Disperse mantissa/exponent of single precision floating point type real number... 7-126
UNIF Unite mantissa and exponent to single precision floating point type real number ..7-128
EXP Exponent function operation.....7-130
LOG Natural logarithm operation 7-132
LOG10 Constant logarithm operation.....7-134
RAD Convert degree (°) to radian unit.....7-136
DEG Convert radian to degree (°) unit.....7-138
SIN Angle (radian) to sine value.....7-140
COS Angle (radian) to cosine value 7-142
TAN Angle (radian) to tangent value.....7-144
ASIN Sine value to angle (radians).....7-146
ACOS Cosine value to angle (radians).....7-148
ATAN Tangent value to angle (radian).....7-150
ATAN2 Coordinates to angle (radian) 7-152
Text Processing Functions 7-154
ASC Convert BIN data to HEX ASCII code.....7-154
RASC Convert HEX ASCII code to BIN data.....7-156
DASC Convert BIN data to DEC ASCII text string.....7-158
STR Convert numerical value data to ASCII code.....7-162
RDASC Convert DEC ASCII text string to BIN data.....7-166
HASC Convert BIN data to HEX ASCII text string.....7-168

RHASC	Convert HEX ASCII text string to BIN data	7-172
FASC	Convert single precision floating point type real number to text string	7-174
RFASC	Text string to single precision floating point type real number conversion	7-178
VAL	Text string to single precision floating point type real number conversion	7-181
LEN	Detect text string length	7-182
SRGHT	Cut text string on right	7-184
RIGHT	Cut text string on right	7-186
SLEFT	Cut text string on left	7-188
LEFT	Cut text string on left	7-190
SMID	Cut text string	7-192
MID	Cut text string	7-194
SRPLC	Replace text string section	7-196
REPLACE	Replace text string section	7-199
SINS	Insert text string	7-200
INSERT	Insert text string	7-202
SDEL	Delete text string section	7-204
DELETE	Delete text string section	7-206
TRIM	Trim 1-byte space and tab at ends of text string	7-208
LTRIM	Trim 1-byte space and tab at left end of text string	7-210
RTRIM	Trim 1-byte space and tab at right end of text string	7-212
STRIM	Trim 1-byte space and tab from specified text string position	7-214
SPLIT	Split text string	7-216
SFIND	Search text string	7-218
SFINDN	Search text string	7-220
INSTR	Search text string	7-222
FIND	Search text string	7-226
DISS	Disperse text string	7-228
UNIS	Unite text string	7-230
CHR	Convert HEX ASCII code to characters	7-232
CPSASC	CIP text string conversion	7-234
RCPSASC	CIP text string type data reverse conversion	7-236

Chapter 8 EXTENDED FUNCTIONS

Data processing function 8-2

SEG	Decode 4-bit data to 7-segment	8-2
BCNT	ON bit count of specified device	8-6
DCNT	Count detected data in specified block	8-8
SER	Detect data in specified block	8-10
DSER	Detect MAX value in a specified block	8-12
MAX	Detect maximum value in a specified block	8-14
MIN	Detect minimum value in a specified block	8-18
AVG	Average value of specified block	8-22
WSUM	Total value of specified block	8-26
BSUM	Total value (in byte units) of specified block	8-30
CRC	Calculate CRC value in specified block	8-32
ZRES	Reset specified block	8-34
BCMP	Compare data between specified blocks	8-36
BCMPI	Compare data of value with block	8-38
RND	Generate random number	8-40
SORT	Binary data sort	8-42
SORTN	Binary data partition sort	8-44

Table processing function 8-48

FIFOW	Write to FIFO table	8-48
FIFOR	Read from FIFO table	8-52
LIFOW	Write to LIFO table	8-56
LIFOR	Read from LIFO table	8-60
FWRIT	Overwrite data table specified position	8-64
FINS	Insert to data table specified position	8-66
FDEL	Delete data table specified position	8-68

Clock processing function 8-70

WTIME	Write time data to calendar timer	8-70
SEC	Convert date/time format data to second format data	8-72
RSEC	Convert second format data to date/time format data	8-74
AJST	Calendar timer adjustment	8-76

High-speed processing function 8-78

DI	Disable interrupt	8-78
EI	Enable interrupt	8-78
RFSCTH	High-speed Counter refresh	8-79
DIC	Set interrupt disabled range	8-80
IEDGE	Set interrupt condition	8-82

PID Functions 8-84

PIDAT	PID control with automatic tuning	8-84

Log Functions 8-86

LOGE	Enable logging of specified ID	8-86
LOGD	Disable logging of specified ID	8-86
TRGD	Acquire logging data	8-87
TRGR	Get device / variable data	8-88

Storage Functions **8-90**
MWRIT Write to storage device (memory card/CPU memory) 8-90
MREAD Read from storage device (memory card/CPU memory) 8-92
MFREE Get storage device (memory card/CPU memory) free space in bytes 8-94
MFREEK Get storage device (memory card/CPU memory) free space in kilobytes ... 8-96
MMKDIR Make storage device (memory card/CPU memory) directory 8-98
MRMDIR Delete storage device (memory card/CPU memory) directory 8-102
MDEL Delete storage device (memory card/CPU memory) file 8-104
MPRINT Write text to a file in storage device (memory card/CPU memory) 8-106
MREADL Read one line from file in storage device (memory card/CPU memory) ... 8-108
MCOPY Copy file in storage device (memory card/CPU memory) 8-110
MMOV Move file in storage device (memory card/CPU memory) 8-112
MREN Rename file in storage device (memory card/CPU memory) 8-114
MSTAT Get status of file in storage device (memory card/CPU memory) 8-116
Access Window Functions **8-118**
AWNUM Display access window message 1 8-118
AWMSG Display access window message 2 8-120
AWSHOW User message display 8-122
AWHIDE Hide user message 8-123
FREE RUN Counter Functions **8-124**
RFSFRC FREE RUN counter refresh 8-124
Sensor Setting Function **8-126**
SPRD Read sensor parameter 8-126
SPWR Sensor parameter write 8-128
SSVC Sensor service execution 8-130
Cyclic Communication Refresh Function **8-132**
RFSCI Refresh cyclic communication input 8-132
RFSCO Refresh cyclic communication output 8-134

APPENDICES

CR/CM List **A-2**
Control Relays (CR) (For KV-8000) A-2
Control Memory Entries CM (For KV-8000) A-5
Control Relays CR (For KV-7500/7300) A-10
Control Memory Entries CM (For KV-7500/7300) A-13
Control Relays CR (For KV-5500/5000/3000) A-18
Control Memory CM (For KV-5500/5000/3000) A-24
Control Relays CR (For KV Nano series) A-31
Control Memory Entries CM (For KV Nano series) A-41
ASCII Code Table **A-53**
List of Characters That Cannot Be Used **A-54**
List of Unusable Functions **A-58**
Index **A-59**
Functions Index **A-67**

Terminology

This manual uses the following terminology excluding some instances.

Term	Explanation
Programmable controller	An electronic device which can control freely the machines by altering programs, which is also called PLC (Programmable Logic Controller).
KV-8000/7500/7300/5500/5000/3000/1000	Programmable Controller KV-5500/5000/3000/1000 Series unit made by Keyence Corporation
KV-8000/7000/5000/3000/1000 Series	The generic term for expansion units that can be connected to the KV-8000/7000/5000/3000/1000 CPU unit.
CPU unit	KV-8000/7500/7300/5500/5000/3000/1000
KV-N14/N24/N40/N60/NC32	Programmable Controller KV Nano Series unit made by Keyence Corporation
KV Nano Series	Generic term of expanded units connectable to Base units in "KV-N14/N24/N40/N60/NC32"
Base unit	KV-N14/N24/N40/N60/NC32
KV-8000 Series expansion unit	Expansion I/O units and expansion special units other than CPU units that can be used with the KV-8000 Series.
KV-8000/7000 Series expansion unit	Expansion I/O units and expansion special units other than CPU units that can be used with the KV-8000/7000 Series.
KV-5000/3000/1000 Series expansion unit	Expansion I/O units and expansion special units other than CPU units that can be used with the KV-5500/5000/3000/1000 Series.
KV Nano Series expansion unit	Expansion I/O units and expansion special units other than base units that can be used with the KV Nano Series.
KV STUDIO	Supportive software for the programming of KV Series.
Ladder program	Program made by using the KV STUDIO Ladder Support Software
<<KV-5000 Ver.1.1>> <<KV-3000 Ver.1.1>>	Indicates KV-5000/3000 released after September 10, 2009, of which the CPU function version 1.1 is indicated as Ver. 1.1 on the serial label on the side. Used for distinction from KV-5000/3000 older than CPU function version 1.1.

Symbols

Menus or buttons in descriptions in this manual are expressed using the following symbols.

Term	Explanation
#	Values prefixed by "#" are expressed in decimal. Basically, all numerical values are expressed in decimal even if they are not prefixed by "#".
$	Values prefixed by "$" are expressed in hexadecimal.
"DM0·DM1"	When 32-bit numerical values are handled, two devices are used. Programs are scripted using only device Nos. (in the example on the left, DM0) to which the lower 16 bits are stored. The example on the left indicates that 32-bit data is being handled.
↑	This indicates "(device turns ON) at the rising edge of a signal."
⊓	This indicates "for the duration that the signal is ON."
↓	This indicates "(device turns OFF) at the falling edge of a signal."

1

OVERVIEW

KV Scripts ····· 1-2
KV Script Type ····· 1-8
KV Script Function ····· 1-10

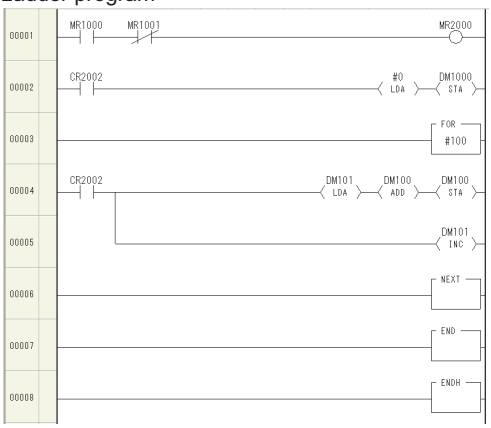
KV Scripts

KV script is a program language developed for operation processing and text string processing that failed using former Ladder language. Since KV script can be programed in Ladder language, Ladder program needs not to be interrupted during application.

Features of KV Script

Ladder program is applicable to feedback control programming of driver control and sensor, etc.
For complicated value operation processing and text string processing program, the programming becomes complicated, and the debug and maintenance also difficult.
KV script is a programming language developed to remedy main problems of these ladder programs. Using script to supersede complicated part that cannot be described by ladder program, allows to prepare program effectively.

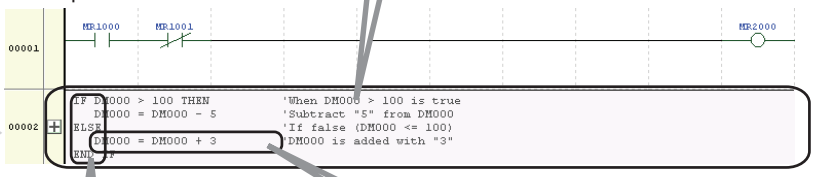
Ladder program



Multi-rung programming in ladder program can result in different understanding of steps.
Simple calculations (such as +, -, x, /) must use specified instructions. Comments by circuit block is difficult to understand.

Can coexist with Ladder. Each script can be added with comments.

KV script



Thanks to programming control statement, program steps are easy-to-understand.

Even if not aware of internal register, arithmetic operation can be described as mathematic formula.

Reference Script program is executed after ladder conversion.
Script program operation confirmation etc may be monitored by converted ladder programs.

How to Use KV Scripts

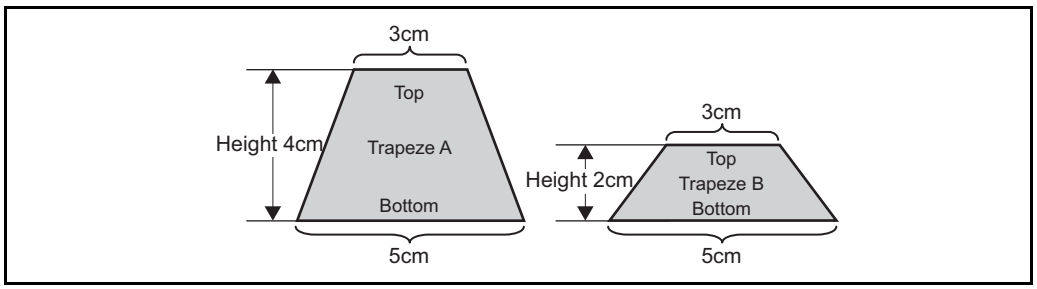
We will use the following example to describe how to realize operation processing and text string processing that failed using Ladder program.

■ Programming operation processing with KV Scripts

Exclusive Ladder program method is required to program operation processing with Ladder programs. When operation formula contains multiple operators and brackets, it be disassembled one by one, programed according to precedence order of the operation, otherwise correct result will be not obtained.

● When using operators

Trapezoid A and B in different heights are shown below.
Please write a script program, summate the area of trapezoid A and B, and store the result to DM50.

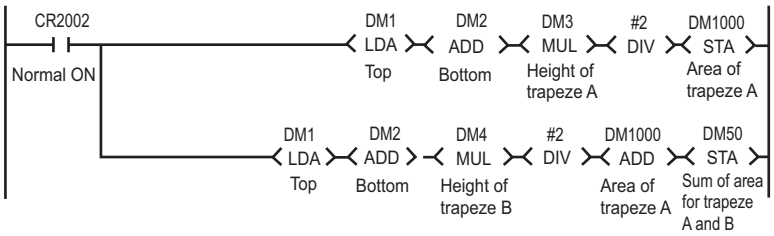


Programming Script

```
'------<The comments are as follows >-------------------------------------------
' Calculate trapezoid area according to (upper bottom + lower bottom) x height 12.
' DM01 = 3  : Stores value of upper bottom.
' DM02 = 5  : Stores value of lower bottom.
' DM03 = 4  : Stores height value of trapezoid A.
' DM04 = 2  : Stores height value of trapezoid B.
' Calculate summation of trapezoid A and B.
' ------< The above is comments  >-------------------------------------------
DM50 = ((DM01 + DM02) * DM03 / 2) +((DM01 + DM02) * DM04 / 2)
```

As mentioned above, the trapezoid area formula can be programmed using a rung of script program.
Same processing can be programmed using ladder as follows.

Ladder Programming



"Firstly save the area of trapezoid A to TM (DM1000), calculate area of trapezoid B, and plus the area of trapezoid A" shall be described in ladder program.

● When using operation functions

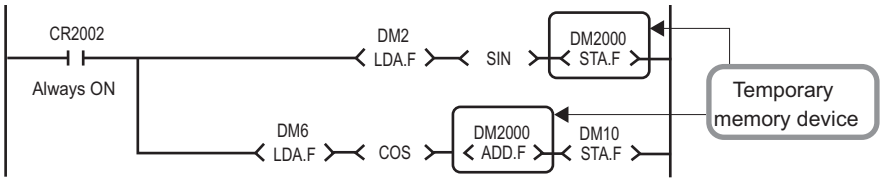
To add sin value and cos value of an angle.
The operation result (addition result) is stored in DM10.F.

Programming Script

```
'------<The comments are as follows >-----------------------------------------------
' DM02.F  : Stores angle value to be SINed.
' DM06.F  : Stores angle value to be COSed.
'------< The above is comments  >-----------------------------------------------
DM10.F = SIN(DM02.F) + COS(DM06.F)    ' Add sin value and cos value.
```

Programming via script functions such as SIN function and COS function is more simple than same processing via ladder.
Ladder programming example as follows.

Ladder Programming



During ladder programming, the operation results of SIN function and COS function need to be temporarily stored in TM before addition operation.

Reference

Operation function is a function for operation processing such as arithmetic/compare processing and logical operation, etc.
It includes ANDA function and ORA function for bit data operation, BMOV function, FMOV function for data movement, and SIN and COS floating point function, etc.

■ Programming text string processing with KV Script

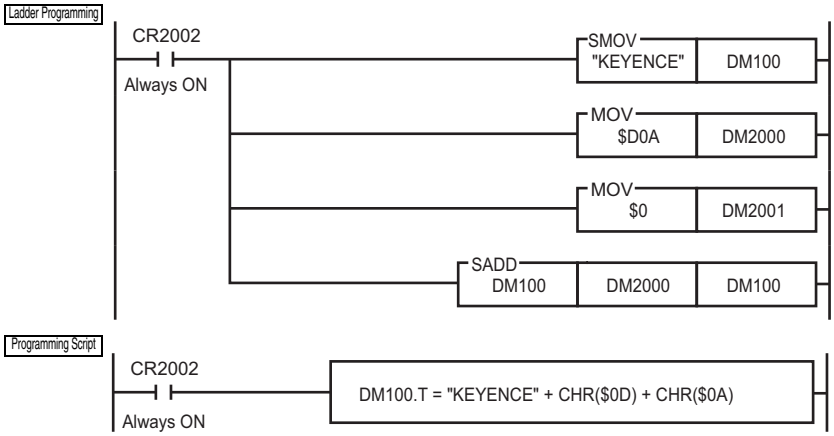
KV script simple and convenient programming even if character processing (text string processing).

● When using string operator

From DM 100, plus CR (0DH) and LF (0AH) into text string " KEYENCE" and then store.

Data save status

K(4BH)	E(45H)	: DM100
Y(59H)	E(45H)	: DM101
N(4EH)	C(43H)	: DM102
E(45H)	CR(0DH)	: DM103
LF(0AH)	NUL(00H)	: DM104



Because control codes cannot be directly processed by using SADD instruction in Ladder program, so, shall be stored in the device and then connected to the required text string. Thanks to special function, KV script can simply connect text strings and control codes.

Reference

- The text string is truly the "string" of "character". The text string is bracketed using double quote(" ").
  For detail, see "Fixed text string (text string type constant)", Page 3-53
- Suffix .T added to DM100 indicates that the device for storing text string with KV script.
  For detail, see "Data Type (suffix)", Page 3-62

● When using text string processing function

Search "@" from "Keyence@KV-5000", extract the programed " KV-5000" and save to DM50.T. The program is described in script form as follows.

Programming Script

```
' ------<The comments are as follows >-------------------------------------------
' DM40.T  : Stores string "Keyence@KV-5000".
' DM10    : Stores "@" position (SFIND search result).
' DM20    : Stores DM40.T string length (LEN return value).
' DM30    : Stores string length of "KV-5000".
' ------< The above is comments  >-------------------------------------------
DM40.T = "Keyence@KV-5000"

DM10 = SFIND(DM40.T, "@", 0)        ' Search "@" from DM40.T, store the position to DM10.

DM20 = LEN(DM40.T)                  ' Obtain string length of DM40.T.

DM30 = DM20 - DM10 - 1              ' Obtain string length of "KV-5000".

DM50.T = SRGHT(DM40.T, DM30)        ' Obtain "KV-5000".
```

When processing such a complicated processing as text string, script function allows simple programming as the sample program above mentioned.
Besides, above sample program can be also programmed according to following method.

Programming Script

```
DM40.T = "Keyence@KV-5000"
DM50.T = SRGHT(DM40.T, LEN(DM40.T) - SFIND(DM40.T, "@", 0) - 1)
```

Reference

Text string processing function is a function to perform text string operations such as edition and conversion of text string and string data processing such as search, etc., including LEN function for testing string length, SMID and SLEFT functions for extracting text string, etc.

For text string function, see "Chapter 7 ARITHMETIC OPERATION FUNCTIONS", Page 7-1.

■ Programming control statements with KV Scripts

KV script, can enable functions such as conditional branch and reprocessing, etc. simply via control statements.

● For multi-branch control

The control "To store corresponding version information according to the word data value", can be programmed with SELECT statement as follows.

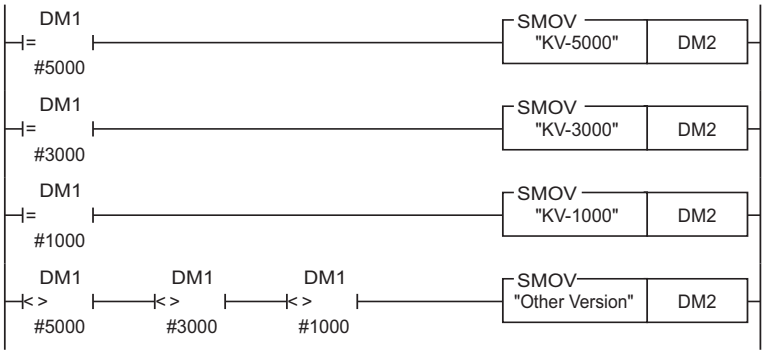
Programming Script

```
' ------<The comments are as follows >-------------------------------------------
' According to version No. stored in DM01,
' store corresponding version information to DM02.
' ------< The above is comments  >-------------------------------------------
SELECT CASE DM01                    ' Identify according to version No.
CASE 5000                           ' When version No. is 5000.
    DM02.T = "KV-5000"
CASE 3000                           ' When version No. is 3000.
    DM02.T = "KV-3000"
CASE 1000                           ' When version No. is 1000.
    DM02.T = "KV-1000"
CASE ELSE                           ' When  no corresponding version No.
    DM02.T = "Other Version"
END SELECT
```

About details of SELECT statement, please see "SELECT ␣CASE to statement (multiple branch control)", Page 4-18

The above script is programmed with ladder program as follows.

Ladder Programming Store version information corresponding version No. saved in DM01 to DM02.



KV Script Type

Two KV script types are available, each with different programming style.

Box Script

Box Script is a script programming execution conditions using ladder.
6 cells from the 5th to the 10th cell on a rung in edit screen is used for programming area. During programming, character code and the number of characters used will be limited and Box Script shall be written within this limited range.

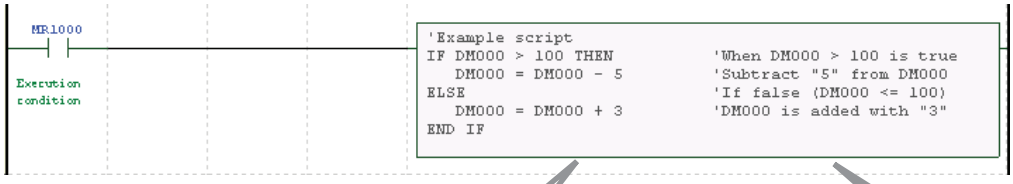
"PROGRAMMING SCRIPT", Page 2-2

Programming Script

```
'Example script
IF DM000 > 100 THEN              'When DM000>100 is true
   DM000 = DM000 – 5             'Subtract "5 " from DM000
When ELSE                        'If false (DM000 <= 100)
   DM000 = DM000 + 3             'DM000 is added with "3".
END IF
```

Programming

Box script programming



The contents behind a single quote are comments.
Chinese characters are allowed in comments.

Line feed must be ensured at the end of contents.

Point **When execution condition is false, the program inside Box Script will be not executed but jumped (please pay attention to partial functions such as differential command, etc.)**

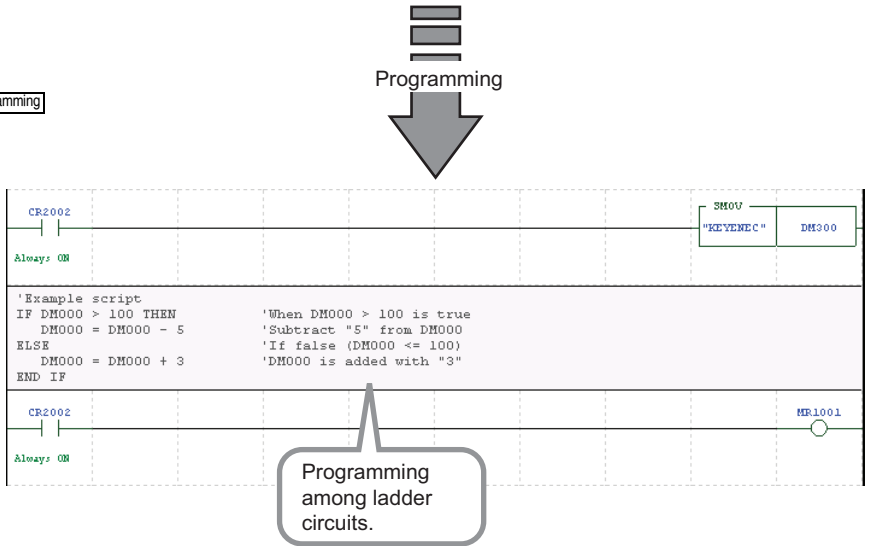
Area Script

Area Script is a script that can be executed during scanning even if the execution condition is not established.
The programming area on edit screen occupies whole 1 rung.

Programming Script

```
'Example script
IF DM000 > 100 THEN                 ' When DM000>100 is true
   DM000 = DM000 − 5                ' Subtract "5" from DM000
When ELSE                           ' If false ( DM000 <= 100)
   DM000 = DM000 + 3                ' DM0000 is added with "3".
END IF
```

Area script programming



KV Script Function

KV script can enable programming program a Series of ladder control in a format similar to BASIC language.

About Assignment Statement

Via KV script, you can program using assignment statement, and store the value and operation result to the devices, variables.

■ Assignment statement

DM1000 = DM1001 + 10

Right result (value) is stored into on the left of "=".

● Numerical value assignment

To store constant 1000 to DM1000.

Programming Script DM1000 = 1000

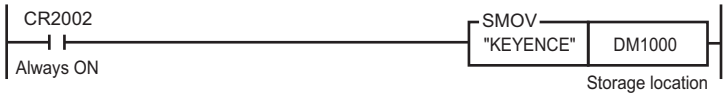
Ladder Programming

● Text string assignment

To store text string "KEYENCE" into DM1000 (NUL (00H) indicating the end of text string will be automatically added).

Programming Script DM1000.T = "KEYENCE"

Ladder Programming

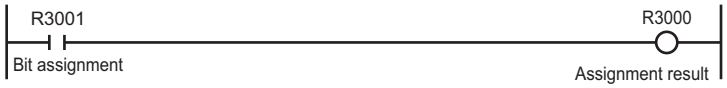


● Bit assignment

To map the status of bit device R3001 to R3000.

Programming Script R3000 = R3001

Ladder Programming



For assignment statement, see "Assignment Statement", Page 4-2.

About Operator

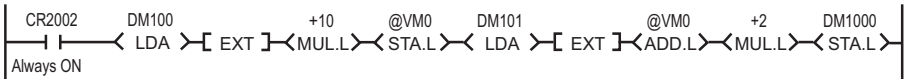
For KV script, arithmetic operation and logical operation, etc. need not to be converted to special instructions such as Ladder program, operators can be directly programed, which enabling more visible programming.

● Operation example

To multiply DM100 value by 10, add DM101 result and then multiply by 2. Assign (store) operation result to DM1000.

Programming Script DM1000.L = 2 * (DM100 * 10 + DM101)

Ladder Programming



● List of operators

Symbols		Processing content	Program example
Arithmetic operator	^	Calculate the power	DM500 = DM100 ^2
	*	Calculate product of 2 value (multiplication)	DM502 = DM100 * 3
	/	Calculate quotient of 2 values (division)	DM504=DM100/4
	MOD	Divide 2 values, return remainder	DM506=DM100 MOD 5
	+	Calculate summation of 2 values (addition)	DM508 = DM100 + 6
	–	Calculate difference of 2 values (subtraction)	DM510 = DM100 -7
Comparison operator	<	Less than	MR500=DM100<10
	<=	Less than or equal to (below)	MR501=DM100 <= 20
	>	Larger than	MR502=DM100 >30
	>=	Larger than or equal to (above)	MR503=DM100 >= 40
	=	Equal to (equivalent) * identical to "=" of assignment	MR504=DM100 = 50
	<>	Unequal to (unequivalent)	MR505 = DM100 <> 60
Text string operator	+, &	Connect 2 text strings	DM600.T="KEY"+"ENCE"
Logic operator	NOT	Calculate logical NOT of value	MR600=NOT R000
	AND	Calculate logical AND of 2 values (bit data relation, word data relation)	MR601=R000 AND R001
	OR	Calculate logical OR of 2 values (bit data relation, word data relation)	MR602=R000 OR R001
	XOR	Calculate XOR of 2 values (bit data relation, word data relation)	MR603=R000 XOR R001

For operators, see "Order of Priority low", Page 4-6.

About Control Statement

KV script allows to program various control statements. facilitating to simply change (control) program flow. Two kinds of control statements are available, including conditional branch statement and loop control statement.

■ Conditional transfer control statements

To select the processing to be implemented according to the condition.

● Type of conditional transfer statement

control statement	Basic format	Description
IF statement	IF␣<Conditional formula>␣THEN ↵ <processing statement> ↵ END␣IF ↵	According to the result of <conditional formula>, determine whether executed or not
	IF␣<Conditional formula>␣THEN ↵ <processing statement 1> ↵ ELSE␣ ↵ <processing statement 2> ↵ END␣IF ↵	According to the result of <conditional formula>, 2 branch control statements are executed separately

	IF␣<Conditional formula 1>␣THEN ↵
	  <processing statement 1> ↵
	ELSE␣IF␣<Conditional expression 2>␣THEN ↵
	  <processing statement 2> ↵
	  .
	  .
	  .
	ELSE ↵
	  <processing statement n> ↵
	END␣IF ↵
		According to the result of multiple conditional expression (<conditional formula 1>, <conditional expression 2 >... <conditional expression n>), multi-branch control statements are executed separately

SELECT statement	SELECT␣CASE␣< word data> ↵
	CASE␣<comparison value 1> ↵
	  <Processing statement 1 > ↵
	CASE␣<comparison value 2> ↵
	  <processing statement 2 > ↵
	  .
	  .
	  .
	CASE␣ELSE ↵
	  <processing statement n> ↵
	END␣SELECT ↵
		According to value of <word data>, select and execute multi-branch control statement correspondingly.

MC statement	MC␣<MC execution condition >␣THEN ↵ <processing statement> ↵ MCR ↵	Master control statement

* ␣ Indicates inserting 1-byte space.
* ↵ Indicates that line feed must be made at the mark.

■ Loop control statement

To process according to specified conditional loop.

● Type of loop control statement

Control statement	Basic format	Description
FOR statement	FOR␣<word data>=<default> ␣TO␣<final value>␣STEP␣<increment > <processing statement> NEXT	Loop processing is performed until <word data>= <final value>
WHILE statement	WHILE␣<Conditional formula> <processing statement> END␣IF	When conditional formula is TRUE, loop processing is performed for uncertain times
DO statement	DO <processing statement> UNTIL␣<Conditional formula>	When conditional formula is FALSE, loop processing is performed for uncertain times
BREAK statement	<Loop conditional statement start> BREAK <Loop conditional statement end>	Used in interrupt loop control statement, unrelated to condition (only used in FOR statement, WHILE statement and DO statement)

*␣ indicates inserting 1-byte space.
*↵ Indicates that line feed must be made at the mark.

For control statement, please see "Control statement", Page 4-10.

About Suffix

In Ladder language, the suffix attached with an instruction can indicate the operation type of the instruction; meanwhile, in KV script, the suffix attached with a device or constant can indicatethe operation type of a function or the meaning of value stored in the device.
For data type (suffix), see "Data Type (suffix)", Page 3-62.

Point

When using variables, they operate as follows.

- **When a suffix is added to variables･･･Operate according to the data type specified by the suffix**
- **When no suffix is added to variables･･･Operate according to the data type set in the variable edit window**

"Variables", Page 3-49

Suffix type

Suffix	Type	Programmable range
.U	unsigned 16-bit data	0 to 65535
.S	signed16-bit data	-32768 to +32767
.D	unsigned 32-bit data	0 to 4294967295
.L	signed 32-bit data	-2147483648 to +2147483647
.F	Single precision floating point type	-3.4E+38≤N≤-1.4E-45 N=0 1.4E-45≤N≤3.4E+38 (Number of effective digits: approx. 7 digits)
.DF*	Double precision floating point type	-1.79E+308≤N≤-2.23E-308 N=0 2.23E-308≤N≤1.79E+308 (Number of effective digits: approx. 16 digits)
.B	Bit type (Boolean value)	1(ON, TRUE), 0(OFF, FALSE)
.T	Text string type	Text string

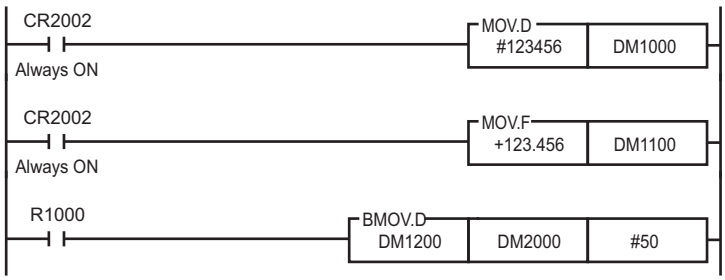
* • The ".DF" suffix can be used only with the KV STUDIO Ver. 6.00 and higher versions.
• The ".DF" suffix can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.
"KV-5500/5000/3000 CPU Function Version", Page 4

Operator

Programming Script

```
DM1000.D = 123456
DM1100.F = 123.456
BMOV(R1000, DM1200.D, DM2000, 50)
```

Ladder Programming



● Omitting suffix

For programming of the device whose suffix omitted, the processing type depends on the device type.

Device type	Type processed when suffix is omitted (default type)
R(DR),B,MR,LR,T,C,CTC*3,CR,UR	.B (bit type)
DM,W,EM*2,ZF*2,FM*2,TM,CM,V,P,UM,UV indirect specifying (*) device	.U (unsigned 16-bit data)
Z*1	KV-8000 :.L (signed 32-bit data) KV-7500/7300 :.L (signed 32-bit data) KV-5500/5000/3000 :.L (signed 32-bit data) KV-1000 :.S (signed 16-bit data) KV Nano Series :.L (signed 32-bit data)
CTH*3	.D (unsigned 32-bit data)

*1 The default type varies with the CPU units used.
*2 EM, FM, ZF cannot be used with the KV Nano Series.
*3 For KV-8000/7000 Series, CTC and CTH are not used.

● Variable suffix

When the suffixes of global variables or local variables are omitted, the type processed will be determined according to the "data type" selected when registering the variables.

Variable data type		Type when the suffix is omitted (default type)
BOOL	Bit	.B
INT	1-word signed integer	.S
DINT	2-word signed integer	.L
UINT	1-word unsigned integer	.U
UDINT	2-word unsigned integer	.D
REAL	Single-precision floating point type	.F
LREAL	Double-precision floating point type	.DF
STRING	String	(.T)*1
TIMER	Timer	.B
COUNTER	Counter	.B
ARRAY		Default data type specified when registering variables
Structure type		Default data type of each member set in the structure definition
Function block type		Default data type of each argument set in the argument setting

*1 No suffix can be written for variables with the data type set to "STRING" (Always use them with the suffix omitted).

● Label suffix

For programming when suffix of global label/local label is omitted, the processing type depends on the "data format" selected when registering a label.

Label data type		Type processed when suffix is omitted (default type)
bit	(BOOL)	.B
1-word unsigned integer	(UINT)	.U
2-word unsigned integer	(UDINT)	.D
1-word signed integer	(INT)	.S
2-word signed integer	(DINT)	.L
Single precision floating point type real number	(REAL)	.F
Double precision floating point type real number	(LREAL)	.DF*1
Text string	(STRING)	(.T)*2
Timer		.B
Counter		.B
Array		This varies depending on the data format in use when the array size is specified.

*1 • The ".DF" suffix can be used only with the KV STUDIO Ver. 6.00 and higher versions.
• The ".DF" suffix can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.
"KV-5500/5000/3000 CPU Function Version", Page 4

*2 For the label whose data type is "text string", suffix cannot be attached (Generally, suffix is omitted when using).

● Device type declaration

By declaring the type of device at the start of script program, default type of every device can be specified.

Declaration statement	Format	Description
TYPE	TYPE DM1000.D	DM1000 will be still processed as .D (32-bit unsigned number). even if suffix omitted
	TYPE EM1000 – EM2000.F (TYPE EM1000.F – EM2000)	EM1000 to EM2000 will be still processed as..F(32-bit single precision floating point type) even if suffix omitted.

For type declaration, see "Data Type (suffix)", Page 3-62.

Point

- **Type declaration is valid only for the script programmed using TYPE.**
- **During type declaration for local label and array, the type larger than data format selected during label registration and text string type shall be declared.**
  Example **When "1-word unsigned integer" is selected in Data Format, if 32 bit type declaration (TYPE [label name].D, etc.) is made, error will occur.**
- **When the type of global label is declared with TYPE, the suffix after type declaration will become valid.**

Reference

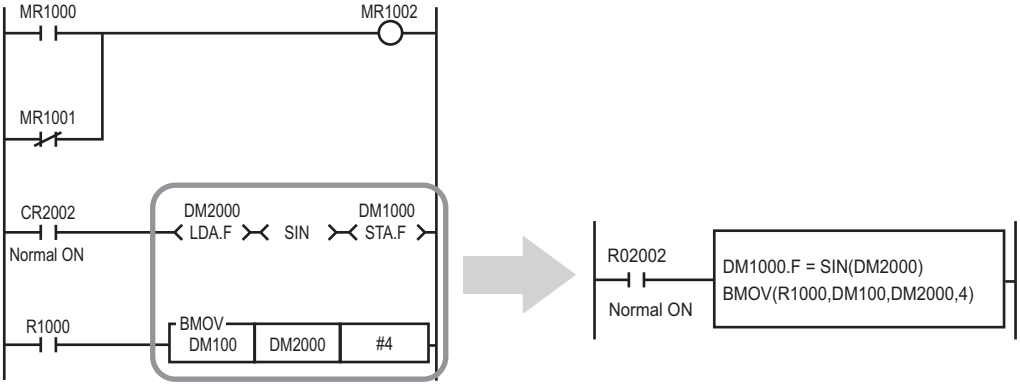
When the device after type declaration is used via indirect specifying, it will be processed as .U by default.

About Script Function

KV-8000/7500/7300/5500/5000/3000/1000 or KV Nano Series ladder instructions can be used via KV script for script function.
To discriminate from ladder instructions, the instruction used via KV script are called "script function".

■ Function

Ladder language compiles program by connecting instructions via symbol lines, but KV script uses assignment statement, operators, control statement and script function to compile program.
Script functions basically cover all ladder instructions. Because that operators and assignment statement can be used, the function corresponding to operation type instructions (ADD, MUL, etc.) and data move instruction (MOV , DW , etc.) can not be used.



Most of script functions shall be programmed according to function name identical to ladder instruction. The action is identical to ladder instruction.

Reference

Since script needs not to operate internal registers, LDA and STA instruction are not used. Some functions are unavailable in ladder instruction but can be added via KV script.

● Script function type

Script functions can be divided into the following types.

Type	Description
Operation processing function	Function group for operation of word data and bit data, Including ANDA and ORA logical operation, BMOV and FMOV for data movement, and SIN and COS floating-point operation etc.
String processing function	Function group for text string processing, including LEN for testing string length, SMID/SLEFT for extracting text string, etc.
Data processing function	Function group for data processing, including MAX, MIN, and BCNT for calculating the number of ON bits in specified word data.
Other Function	Function group for processing other than operation, text string, data processing, including TMR, ADRST for indirect specifying, MWRIT and MREAD functions for memory card control , etc.

■ About operation flag (CR2009 to CR2012)

Every script shall be converted to ladder. During function execution, the status of operation flag will be also changed correspondingly. The status of operation flag will vary before and after execution of script program. Besides, ladder generated via script conversion can be programed according to different function combinations, so, if operation flag varies with the change of script program, it cannot be used.

■ When looking up notes on indirect specifying and index modification

When using indirect specifying and index modification, be careful not to specify a device or work area outside the specified range.
When a device or work area outside the specified range is accessed using indirect reference or index modification in argument of every function, function operates as follows.

● Every function operation table when invalid address is specified

Function operation	Function name
"0 " is stored in return value	LDP,LDF,BLD,BLDB,LDPB,LDFB,UNIF,RDASC,RHASC,RFASC,VAL,LEN,SFIND,INSTR,FIND,SEG,MAX,MIN,AVG,WSUM,BSUM,BCMP,BCMPI,SEC
NUL (00H) is stored in return value	DASC,STR,HASC,FASC,SRGHT,RIGHT,SLEFT,LEFT,SMID,MID,SRPLC,REPLACE,SINS,INSERT,SDEL,DELETE
Value of 1st argument stored in return value *1	ROOT,ANDA,ORA,EORA,ENRA,COM,NEG,SRA,SLA,RRA,RLA,RRNCA,RLNCA,TBCD,TBIN,MPX,DMX,GRY,RGRY,SWAP,DECO,ENCO,TO,USDLF,FLOAT,INTG,INT,EXP,LOG,RAD,DEG,SIN,COS,TAN,ASIN,ACOS,ATAN,ASC,RASC,DASC,CHR,BCNT,DCNT,SER,CRC,WTIME,AWNUM,AWMSG,WTIME,SQRT,ABS,RND
The function is not executed	SET, RES, BOUT, BOUB, BSET, BRES, MCALL, MSTRT, ZPUSH, ZPOP, ADRSET, ADRINC, ADRDEC, ADRADD, ADRSUB, BMOV, FMOV, PMOV, INC, DEC, WSR, WSL, BSR, BSL, DISF, DISS, UNIS, ZRES, FIFOW, FIFOR, LIFOW, LIFOR, FWRIT, FINS, FDEL, RSEC, DIC, LOGE, LOGD, MWRIT, MREAD, MFREE, MMKDIR, RFSX, RFSY, ECALL, MDSTRT, MDSTOP, FRSET, FRSTM, FRLDM, UREAD, UWRIT, UFILL, LIMIT, BANDC, ZONE, APR, DISN, DISB, UNIN, UNIB, DSER, RFSCTH, TRGD, MRMDIR, MDEL, RFSFRC, ACNT

*1 But, if invalid indirect lookup is performed in the 1st argument, the correctness of return value cannot be guaranteed.

2

PROGRAMMING SCRIPT

PROGRAMMING SCRIPT ••• 2-2
Box Script ••• 2-4
Area Script •• 2-16
Label Programming and Label Array •••••••••••••••••••••••••••••••••••• 2-37
Precautions when programming script •••••••••••••••••••••••••••••••• 2-51
Script Conversion Output Message •••••••••••••••••••••••••••••••••••• 2-52
Macro •• 2-60
Script Debug Function •• 2-62
Input Auxiliary Function and Other Info •••••••••••••••••••••••••••••• 2-64

Programming Script

The chapter describes how to operate Box Script/Area Script as well as menus and edit screen.

Programming Script

KV script can be programed directly in text format in ladder edit image.
In addition, it also supports the Windows functions such as Cut/Paste, Copy, Undo/Redo, etc.

How to Program Script

Box script/area script can be programmed by combining ladder program in ladder edit window.
KV script can be programed after checking script programming area in ladder edit window.

(1) Call the script from the menu

Select "Script (S)" → "Insert area script (A)" or "Insert box script (B)".

(2) Call script from right-click menu

When the cursor locates on ladder edit window, select "Edit script (X)" from right-click menu, and then select "Insert area script (A)" or "Insert box script (B)".

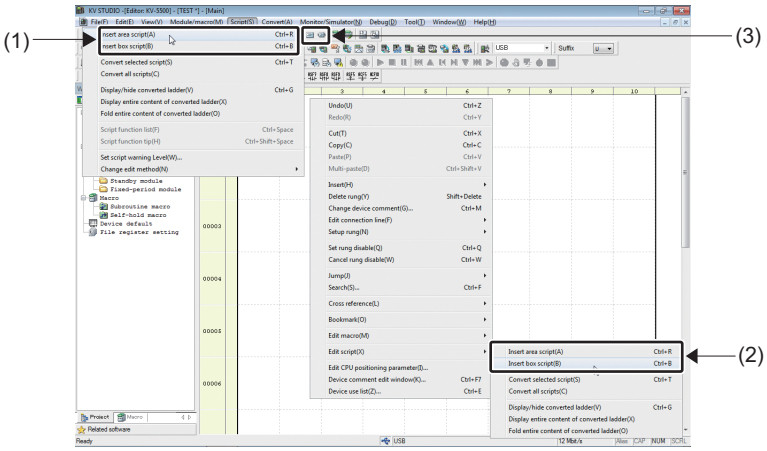
(3) Call script from icon

Click the icon on the toolbar.

- Area script:
- Box script:

(4) Call script using shortcut key

- Area script: Ctrl key+ R key
- Box script: Ctrl key+ B key

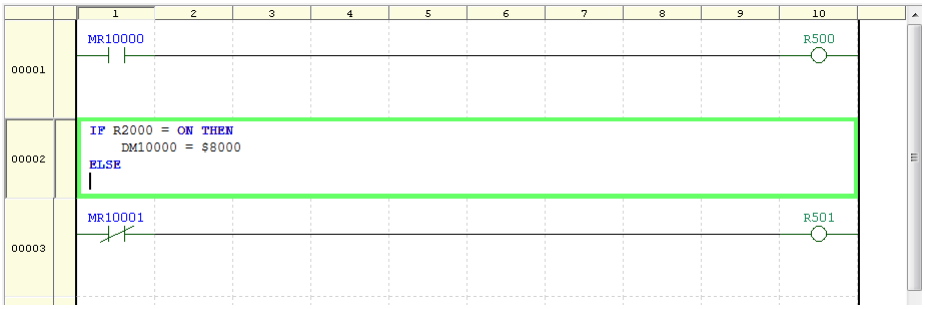


How to Input Scripts

2 Script input methods are available. But function will not vary with different input methods. So, please select as you like.

■ Direct input (default)

Like ladder instruction programming scripts can be directly programmed on Ladder Editor.

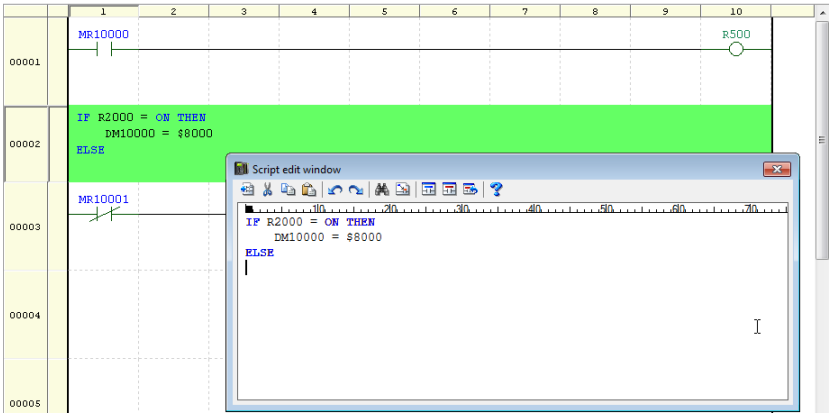


1 **From the menu, select "Script (S)" → "Change edit method (N)" → "Direct input (D)".**

Other procedure From "Tool (T)" → "Option (O)", select "Direct input" in "Input format" of "Script setting" tab.

■ Window input

To input scripts in text edit window displayed during script programming.



1 **From the menu, select "Script (S)" → "Change edit method(N)" → "Window(W)".**

Other procedure From "Tool (T)" → "Option (O)", select "Window (Window entry)" in "Entry method" of "Script" tab.

Box Script

This section describes precautions on how to make/edit and use box scripts.

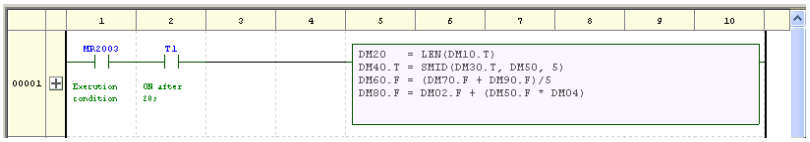
Features of Box Script

Box Script is a script that can specify the time and sequence to executing KV script programs using ladder with ladder execution conditions.
Box Script is executed only when execution condition specified by ladder is true, programming can be adaptable with operation formula.

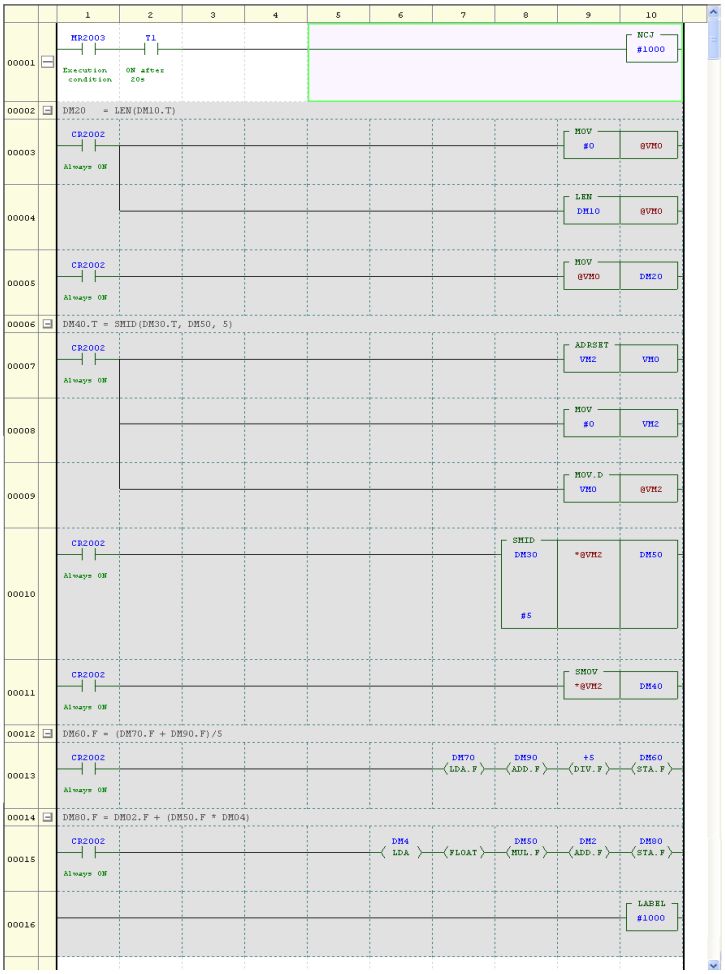
When using box scripts in the following cases

Box scripts are suitable for making programs excluding several lines of operation formula and complicated control statements.
Besides, for execution condition programmed via Ladder, the program in box script can only program execution processing statement, so, control statement is not required, it can be used simply.

Box script



Ladder development

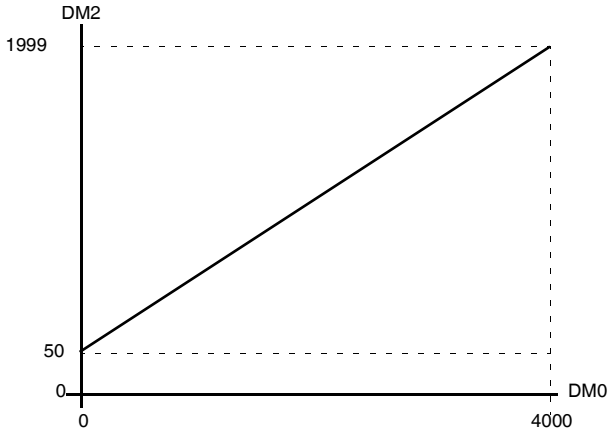


● Example for programming using box scripts

[Scaling]

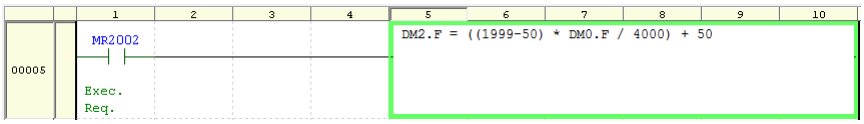
The value within the range of 0 to 4000 stored in DM0.F ([DM1/DM0]) is scaled to the range of 50 to 1999 and stored to DM2.F ([DM3/M2]). The operation is performed via single precision floating point type.

- DM0=0 → DM2=50
- DM0=50 → DM2=74
- DM0=100 → DM2=98
- DM0=500 → DM2=293
- DM0=1000 → DM2=537
- DM0=2000 → DM2=1024
- DM0=4000 → DM2=1999

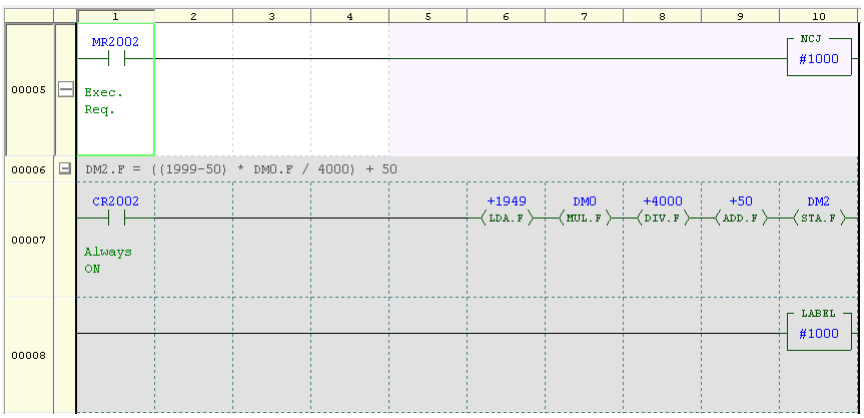


Use Box Script to program above operation.

Box script



Ladder development

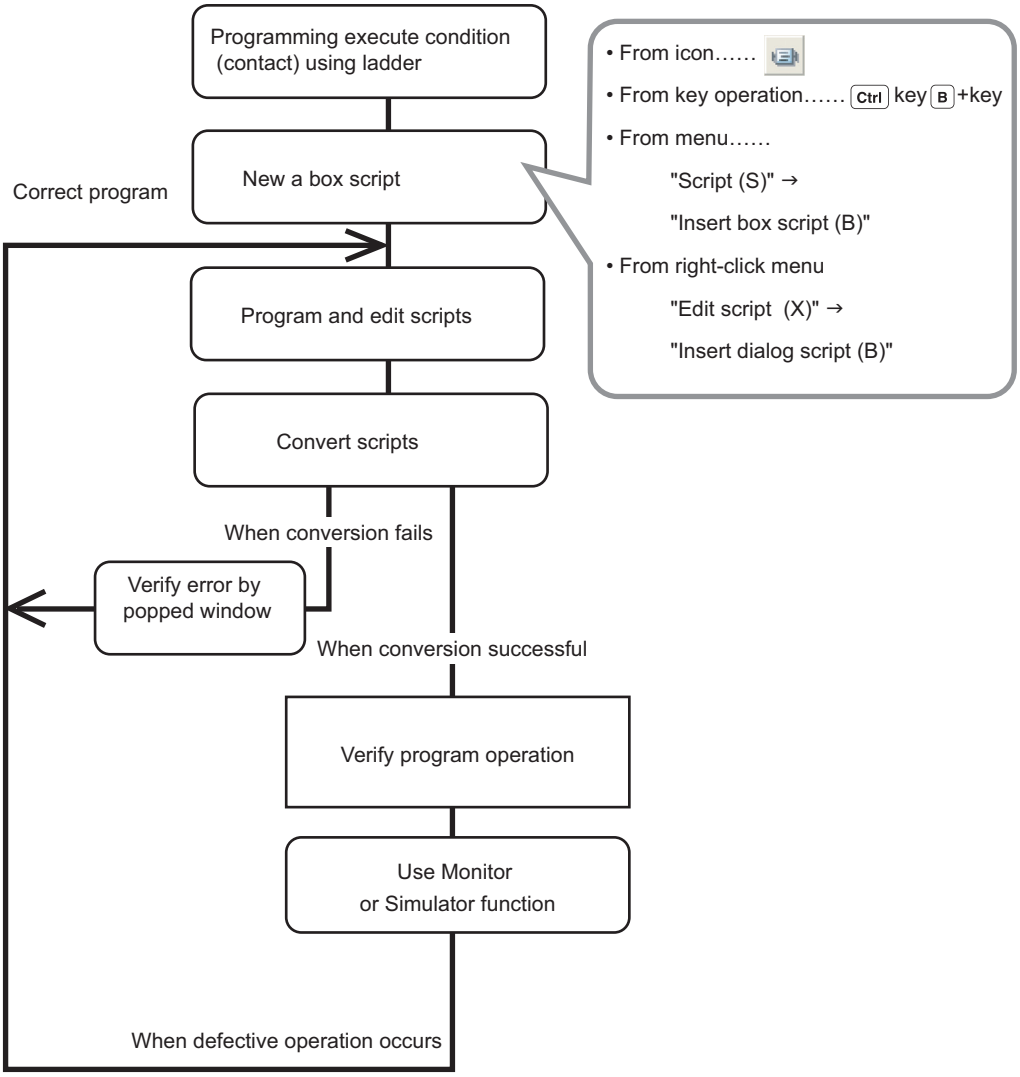


Making Box Script

The flow from making to debugging box script is described as follows.

Operation Flow

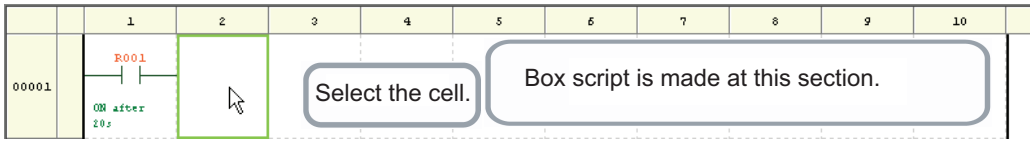
The operation flow for making/editing box script is described as follows.



New a box script

The making steps of box scripts are described as follows.

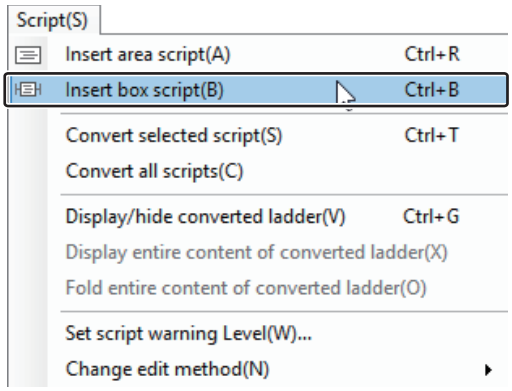
1 To program execution condition in Ladder Editor, click the cell in which box script to be edited is located.



Reference

- Box script occupies 6 cell (cell 5 to 10). Execution condition can be programmed within cell 1 to cell 4.
- Regardless of the position of the selected cell, the box script shall be make in the cell 5 to cell 10.

2 From the menu, select "Script (S)" → "Insert box script (B)".

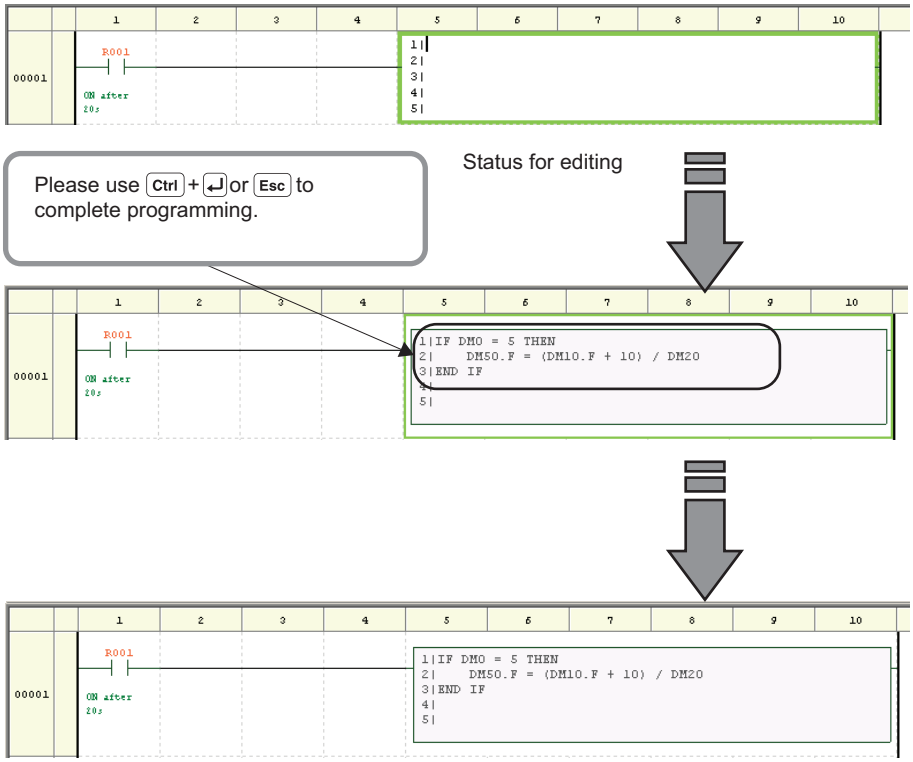


Other procedure

- Ctrl key + B key
- From right-click menu, select "Edit script (X)" → "Insert box script (B)".。

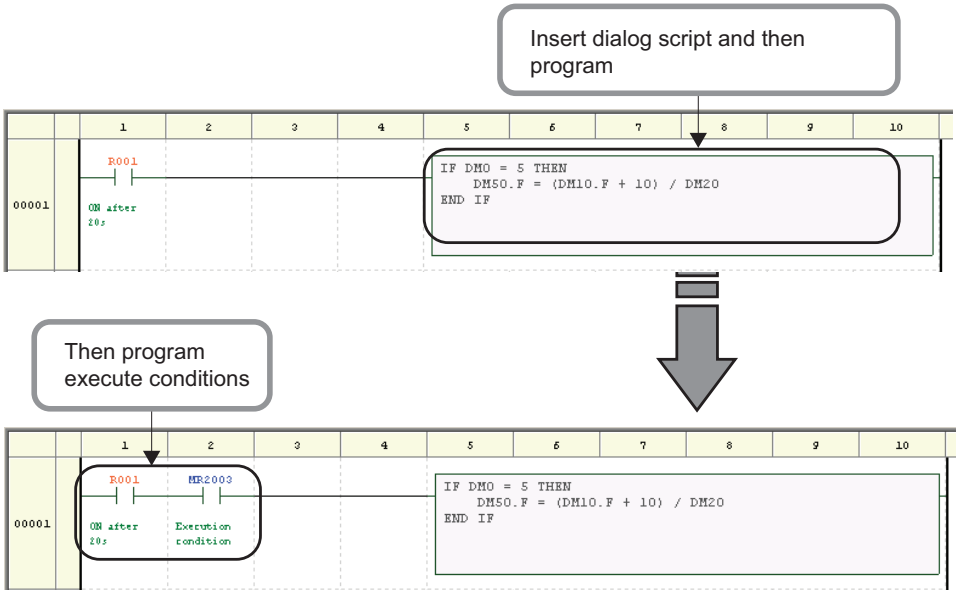
3 Program the inserted box script.

After programming, press Ctrl key + ↵ key, or press Esc key, click the ladder edit window other than box script being edited, complete programming.



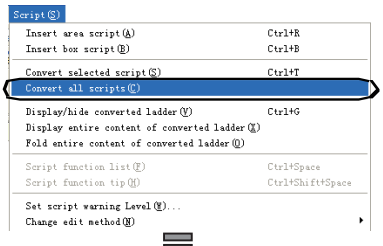
Status after edit completed

Reference Execution condition can be also programed after making box scripts.

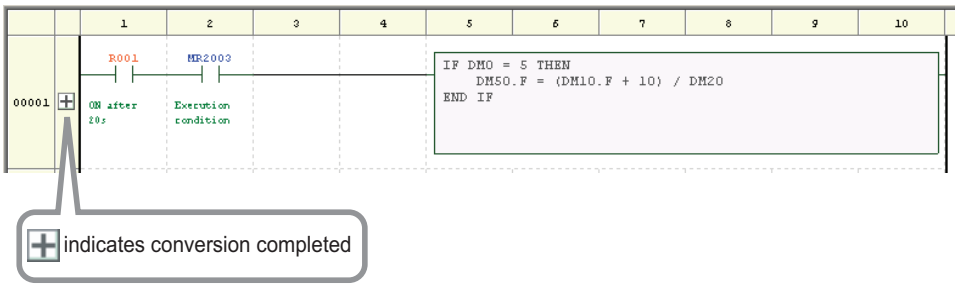


4 After programming, convert box script.

Other procedure From right-click menu, select "Edit script (X)" → "Convert all scripts (C)"



● When conversion successful



The popping up window displays "Warning" message sometimes.

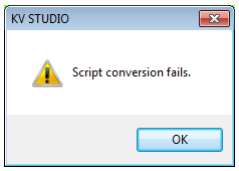
"Warning Message", page 2-52

● When conversion fails

"Script conversion error" will be displayed.

In addition, the popping up window displays error message.

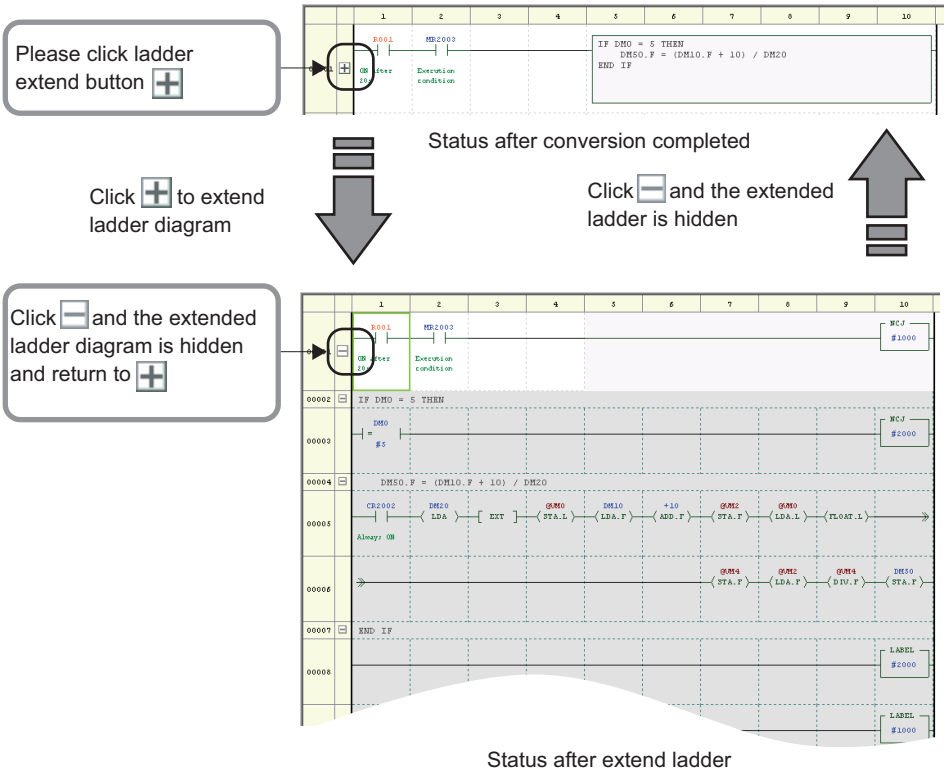
"When conversion error occurs", page 2-55



5 Verify program by using ladder after successful script conversion.

Click "⊞" displayed on the left side of the box script rung to display the converted ladder. (if script conversion fails, "⊞" will be not displayed)

Other procedure
- Select the extended box script, select "Edit script (X)" → "Display/hide converted ladder (V)"from right-click menu.
- Ctrl key+G key



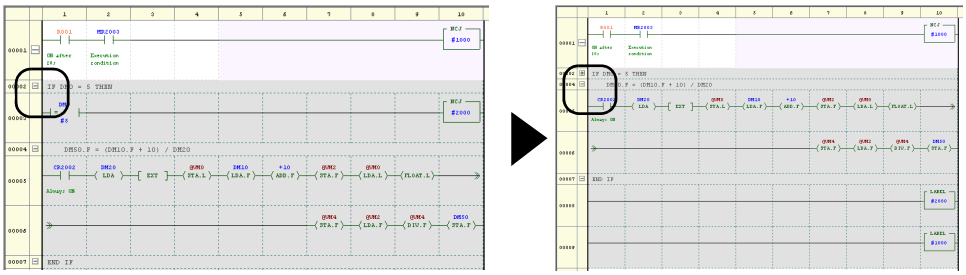
Reference

If editing script after script conversion, the converted ladder will be not displayed. (⊞disappears)
To display ladder, script shall be converted again.

6 The outspread Ladder can be partially hidden (folded).

Click ⊟ beside the extended ladder.

Other procedure Select the comment part (⊞) of the folded ladder rung. select "Display/hide converted ladder (V)" from right-click menu.



Reedit Box Script

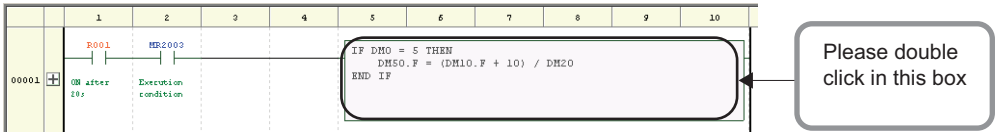
The re-editing steps of box script are described as follows.

■ Reedit Box Script

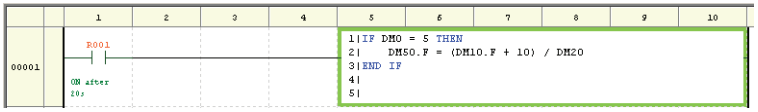
1 To reedit the box script programmed, double-click the Box Script to be edited.

Other procedure After selecting the box script to be edited, click the following keys:

- Esc key
- F2 key
- ↵ key



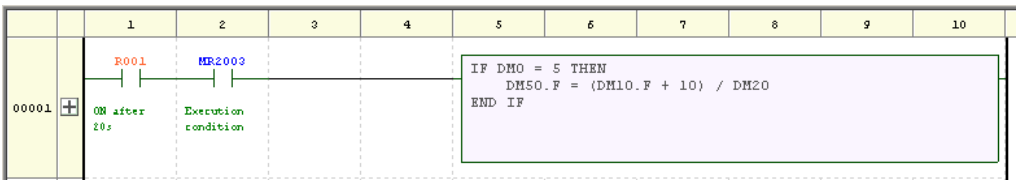
Uneditable status



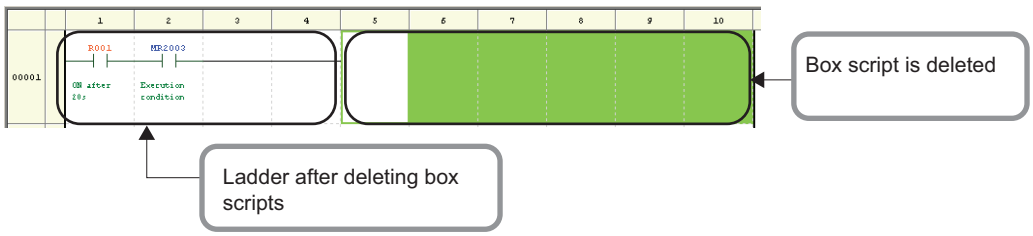
Editable status

■ Delete box script

1 Select the box script to be deleted.

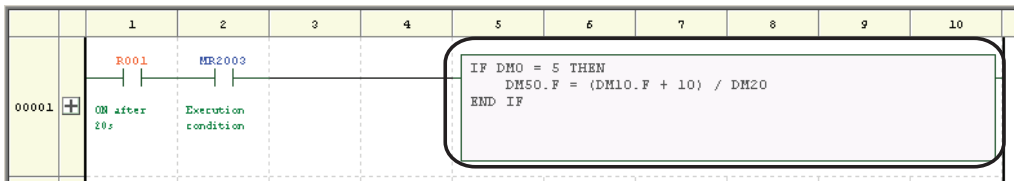


2 Press Delete key in keyboard.



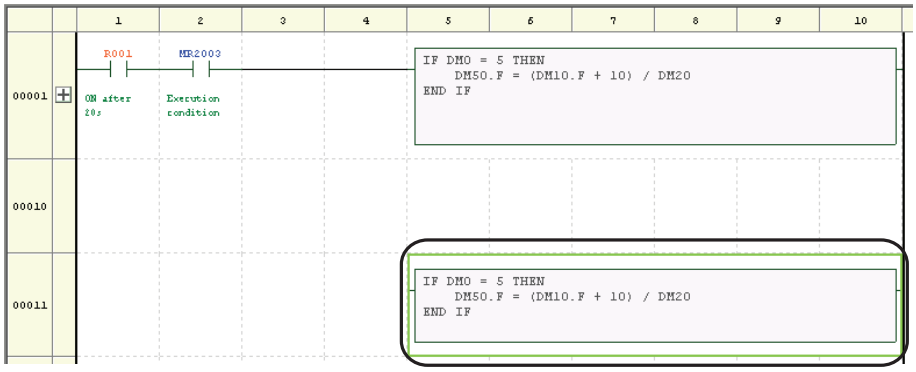
■ Copy box script

1 **Select the box script to be copied, select "Edit (E)" → "Copy (C)" from the menu.**



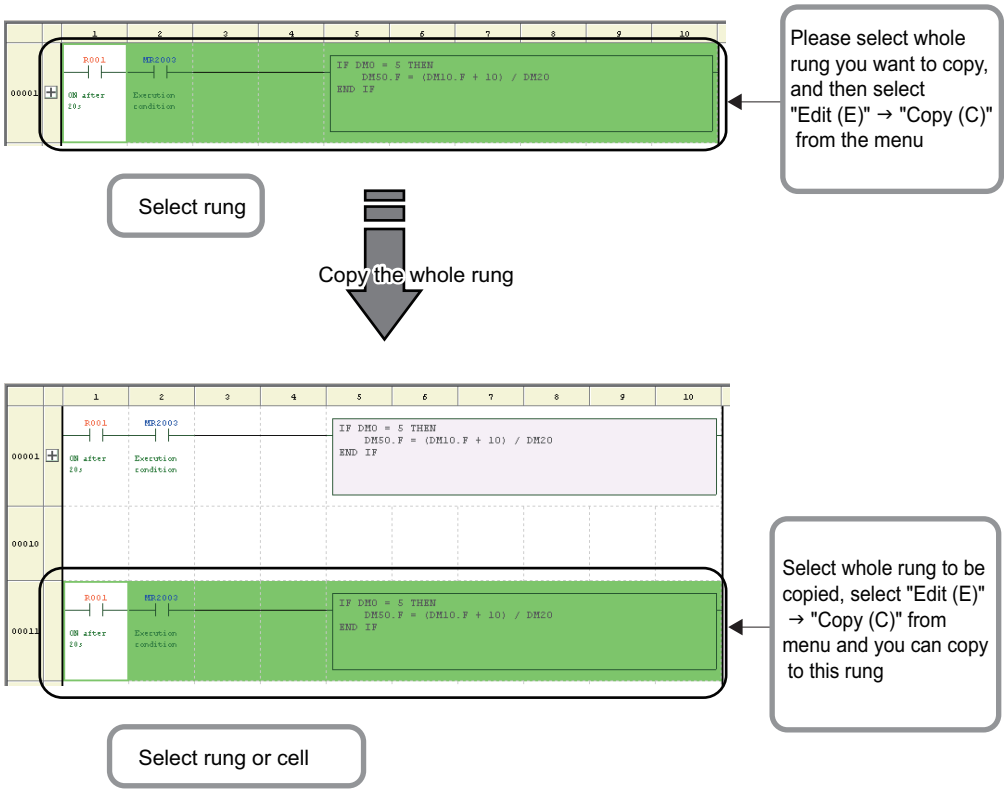
Other procedure
- Ctrl key + C key
- From the right-click menu, select "Copy (C)"

2 **Select the cell (rung) to paste box script copied according to step 1.**
From the menu, select "Edit (E)" → "Paste (P)".



Other procedure
- Ctrl key + V key
- From the right-click menu, select "Paste(P)"

Reference The whole rung including box script can be also copied.



Precautions When Using Box Script

When the execution condition of box script is OFF, the script program will be not executed. For this the following points shall be noted.

● When using differential execution type functions

The previous state of the execution condition of the differential execution type function is the state when the last box script (subroutine) was executed.

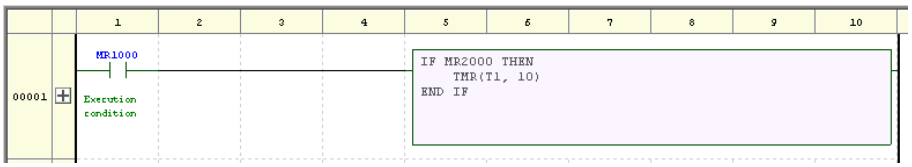
Care must be taken when differential execution type functions are executed for 2 times within 1 scanning, or after Box Script is executed for the nth times, the [n+1]th box script is not executed in continuous scanning.

For details, please see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Precautions when using differential execution Type Instructions."

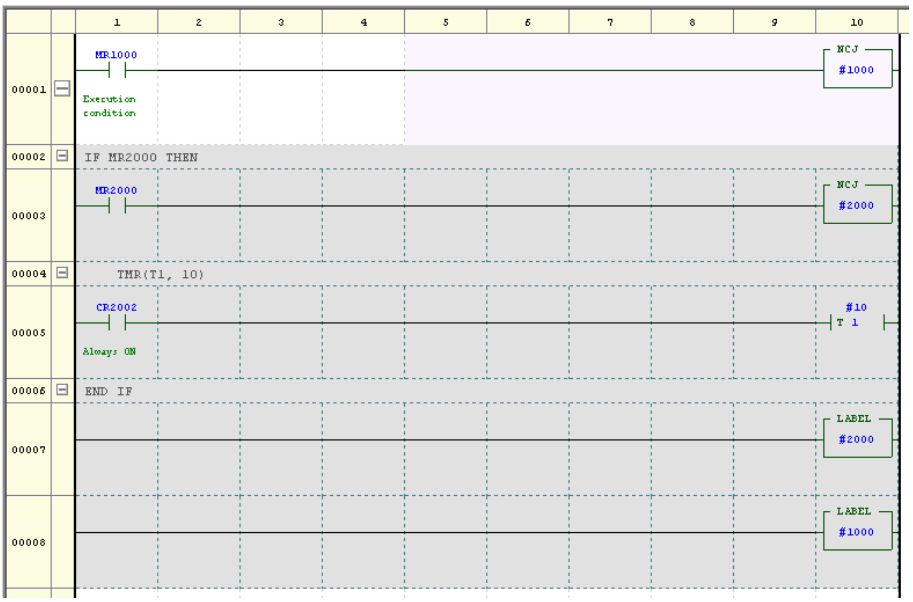
● When using timer function

When timer function is used by Box Script, TMR instruction will be programmed between NCJ to LABEL of the extended ladder program. From this, the timer contacts fail to be correct ON/ OFF sometimes. Besides, TMR function whose execution condition omitted shall be also noted.
For example, when programming the following programs:

Box script



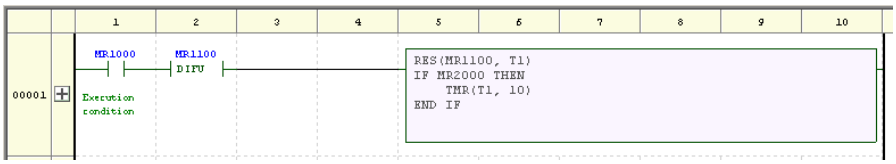
Ladder development



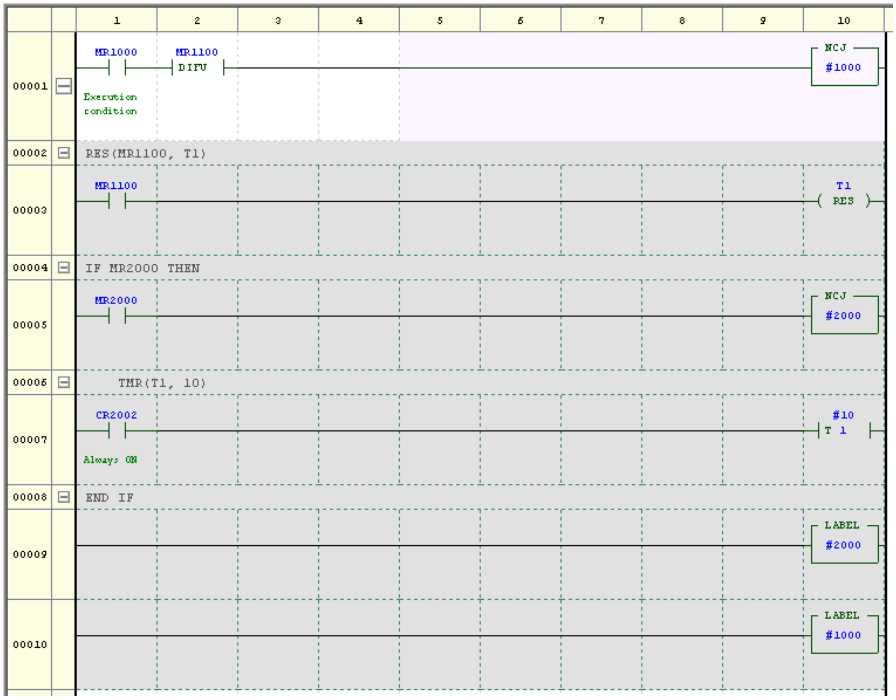
Observe the extend ladder program, if the reset condition of timer instruction is CR2002 (normal ON), the timer cannot be reset after executing.
To solve this problem, DIFU instruction can be connected with AND as execution condition.

As execution condition, connect AND to DIFU instruction

Box script



Ladder development



When execution condition (MR1000) of Box Script is TRUE, MR1100 will be ON only in 1 scanning via DIFU instruction. In this case, RES function programmed on the 1st rung of Box Script enables the timer to reset, unrelated to the previous status of the ladder. After execution of Box Script, the timer is cleared and timer function (contact) acts correctly.

Area Script

This section describes precautions on how to make/edit and use of Area Script.

Features of Area Script

Area script is programmed on a whole rung.

Area script is used in the following cases

Area script is every-scan execution type, suitable to the program including conditional branch and loop control.
Area script allows to simplify programming of complicated conditional branch and multiple branches.

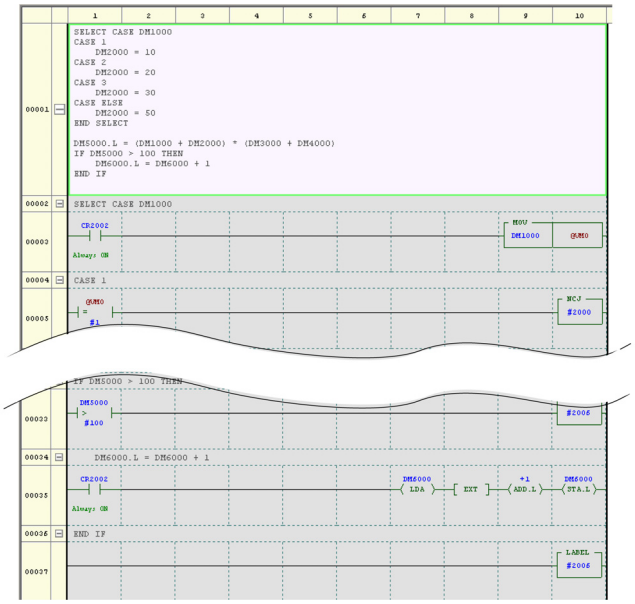
Area Script

```
00001
SELECT CASE DM1000
CASE 1
    DM2000 = 10
CASE 2
    DM2000 = 20
CASE 3
    DM2000 = 30
CASE ELSE
    DM2000 = 50
END SELECT

DM5000.L = (DM1000 + DM2000) * (DM3000 + DM4000)
IF DM5000 > 100 THEN
    DM6000.L = DM6000 + 1
END IF
```

* the description on the right side of single quote (') shall be processed as comment.

Ladder Programming



If conditional branch is complex, or program is lengthened via ladder programming, area script can enable improving programming efficiency.

● Example for programming with area scripts

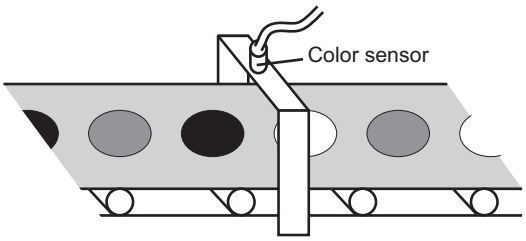
To identify the category of disk passing a certain point on belt conveyer, store corresponding color name.

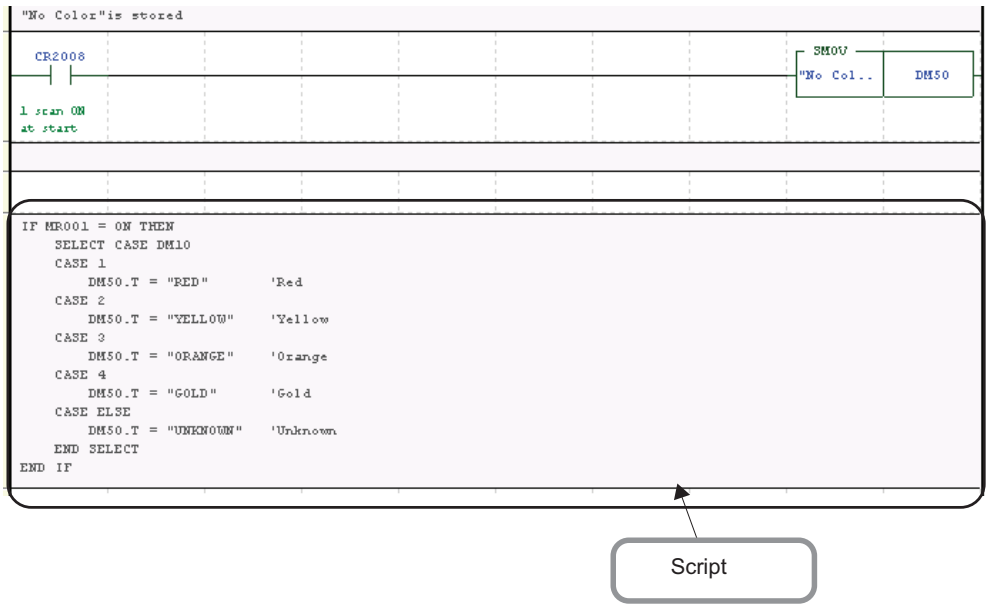
When the disk does not pass the point, the bit device MR001 is OFF. When the disk passes it, it is ON. When MR001 is ON, identify the category of disc stored in DM10, and store the color of disc corresponding to DM50.T.

Using area script to program the part from disk category identifying to storaging.

When the disk reaches the check point, disk color can be detected via color sensor.

- Red "1" is stored in DM10
- Yellow "2" is stored in DM10
- Orange "3" is stored in DM10
- Gold "4" is stored in DM10



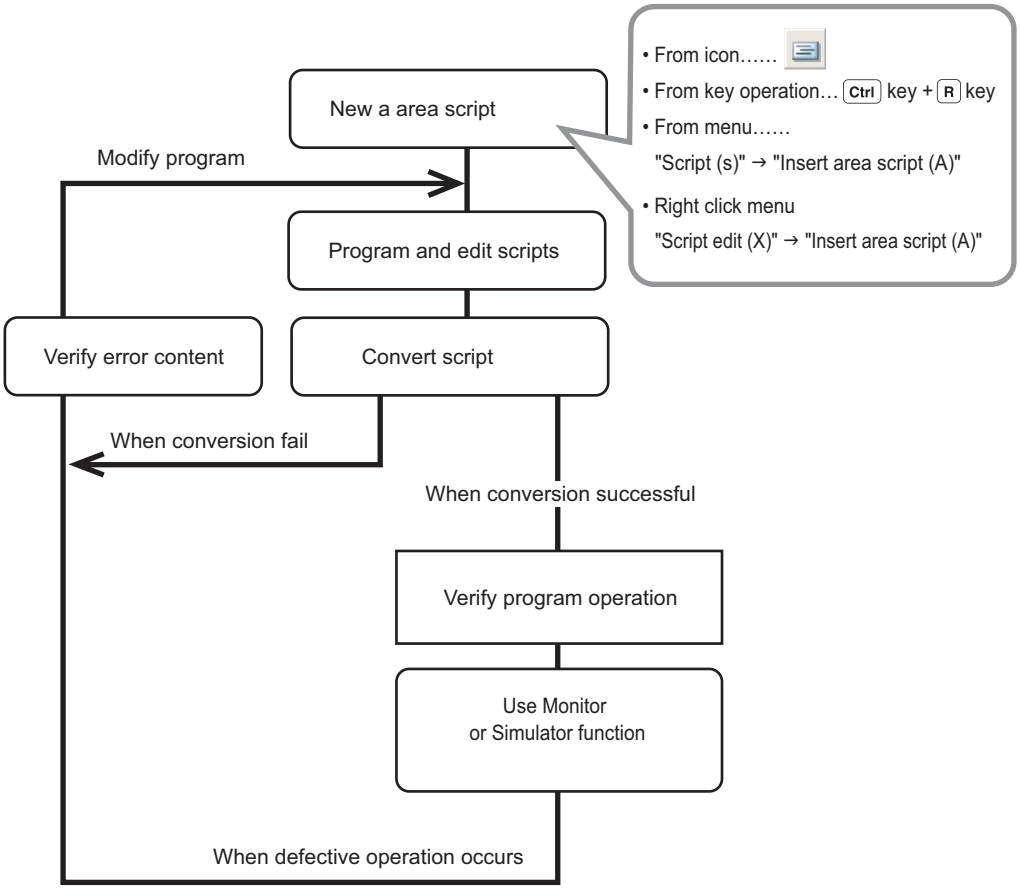


Making Area Script

The flow from making to debugging of Area Script is described as follows.

Area script is used in the following cases

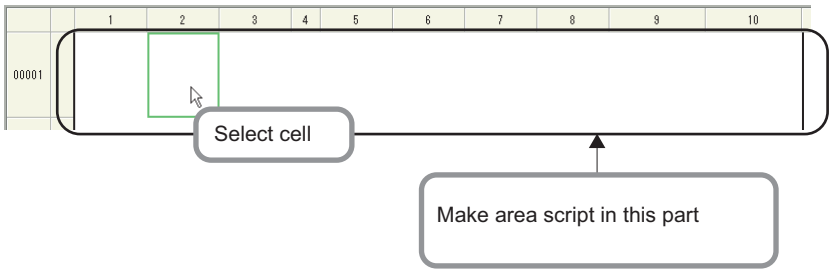
The operation flow for making/editing box script is described as follows.



New a area script

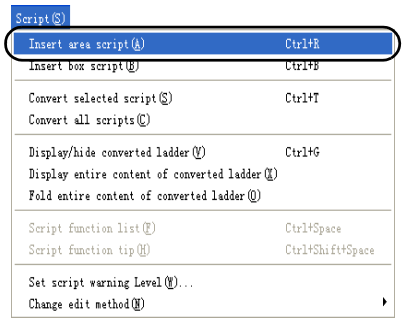
The creation steps of Area Script are described as follows.

1 Click the cell (rung) in which area script will be located.



Reference Area script occupies 1 whole rung (10 cells).

2 From the menu, select "Script (S)" → "Insert area script (A)".



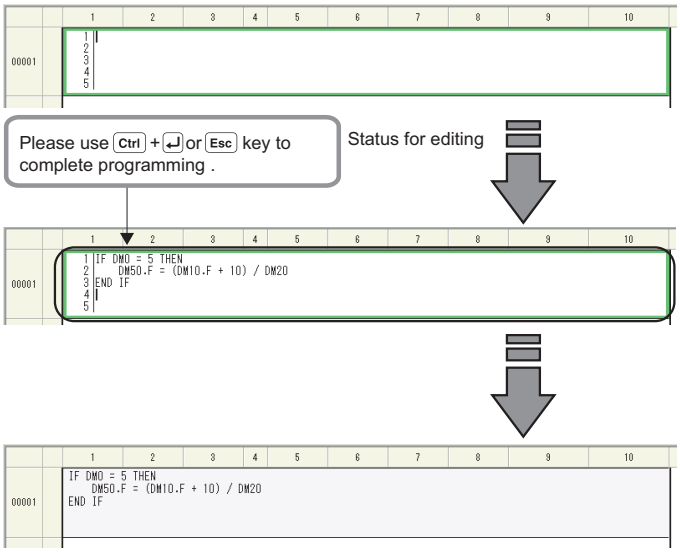
*Area script can be inserted from the menu, tool bar and keyboard.

Other procedure •Press Ctrl key+ R key.

• From right-click menu, select "Edit script (X)" → Insert area script (A)

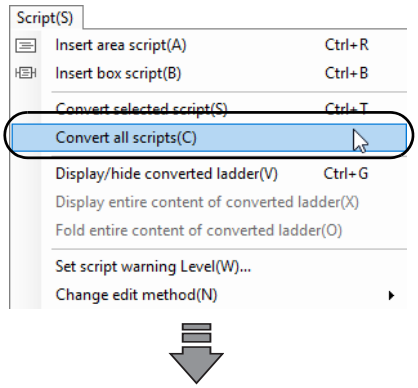
3 Program the inserted area script.

After programming, press Ctrl key + ↵ key, or press Esc key, click the Ladder Edit window other than Area Script being edited to complete programming.

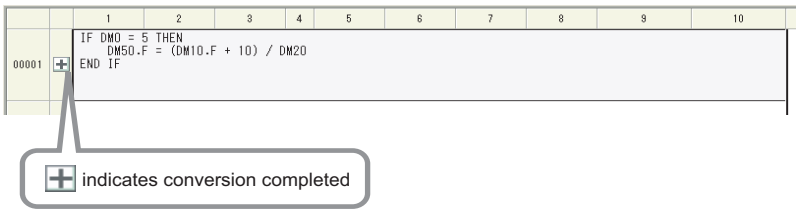


Status after edit completed

4 Area script shall be converted after programming completed.



● When conversion successful



The popping up window displays "Warning" message sometimes.

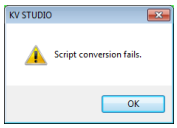
"Warning Message", page 2-52

● When conversion fails

"Script conversion error" will be displayed.

In addition, the popping up window displays error message.

"When conversion error occurs", page 2-55

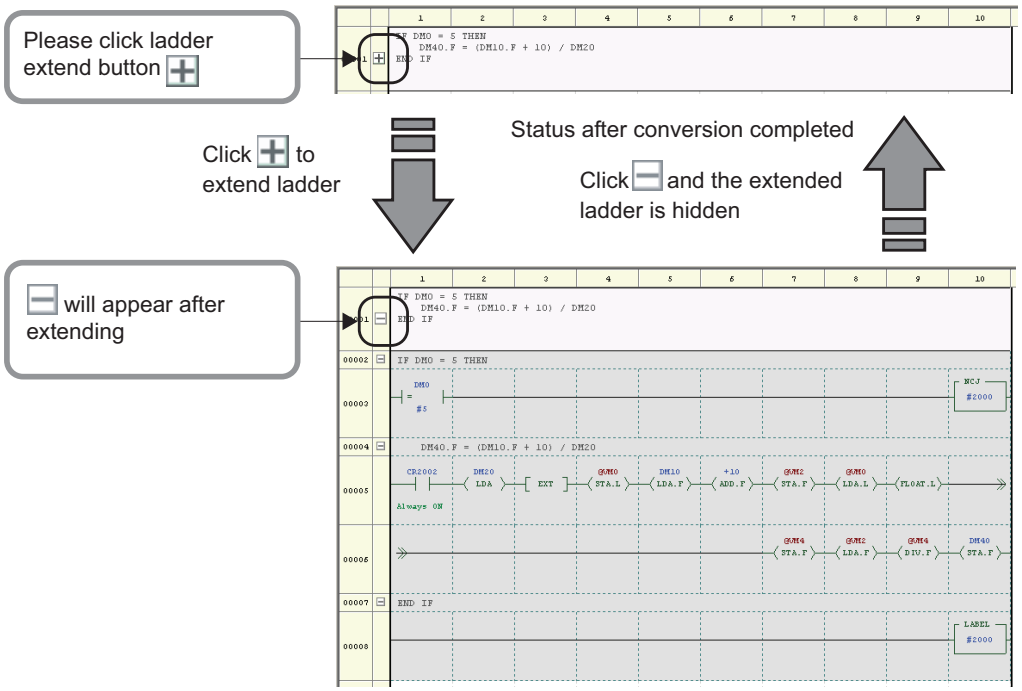


5 After successful script conversion, ⊞ is displayed on the left side of area script. Click ⊞ to extend ladder under area script.

(if script conversion fails, ⊞ will be not displayed)

Other procedure Select the expanded area script, select "Edit script (X)" → "Display/hide converted ladder (V)" from right-click menu.

- Ctrl key+ G key



Status after extending ladder

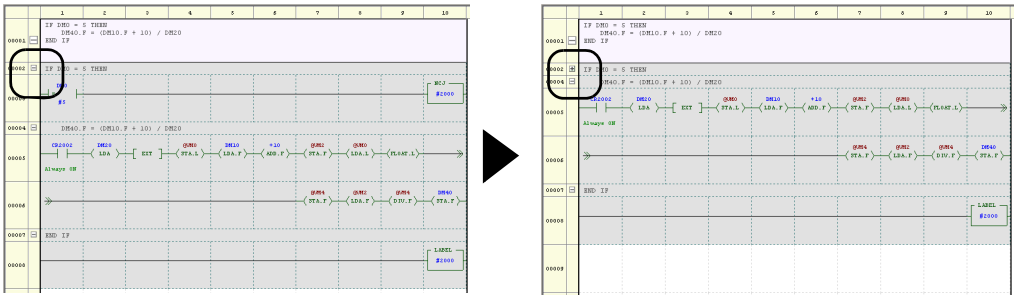
Reference

If editing script after script conversion, the converted ladder will be not displayed. (⊞ disappears)

To display ladder, script shall be converted again.

6 The outspread Ladder can be partially hidden (folded).

Click ⊟ .beside the extended ladder.



Reedit area script

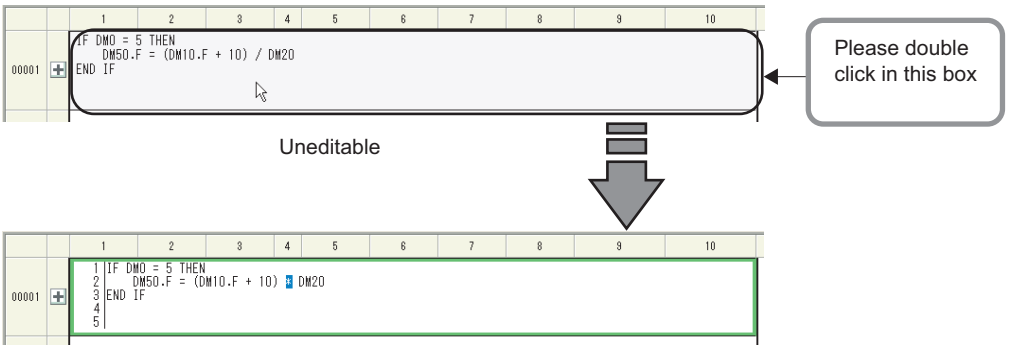
The re-editing steps of Area Script are described as follows.

■ Reedit area script

1 Double-click the area script to be edited to reedit the area script programmed.

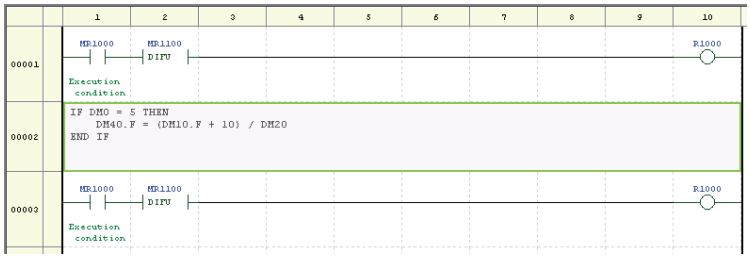
Other procedure Select the Area Script that you want to edit, press the following keys.

- Esc key
- F 2 key



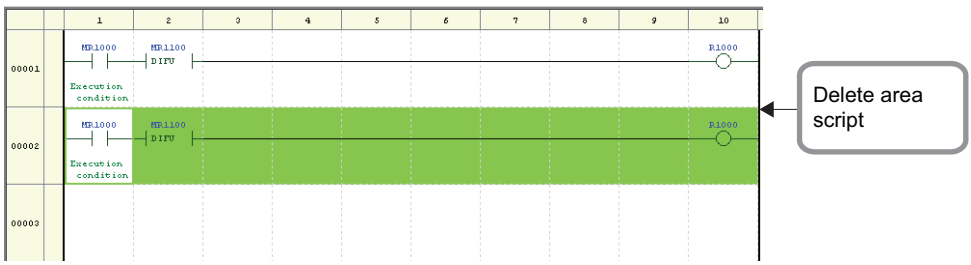
■ Delete area script

1 **Select the area script to be deleted.**



Selected

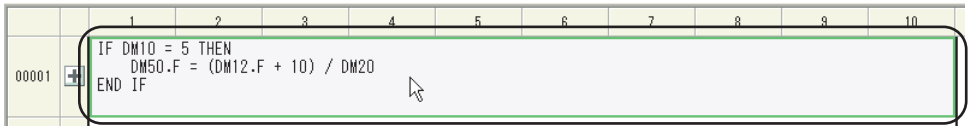
2 **Press Delete key in keyboard.**



Deleted

■ Copy area script

1 Select the area script to be copied, select "Edit (E)" → "Copy (C)" from the menu.

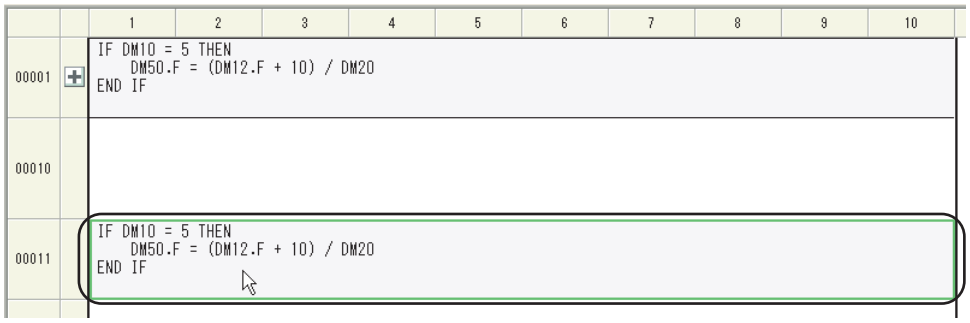


Selected

Other procedure
- Ctrl key + C key
- From the right-click menu, select "Copy (C)"

2 Select the cell (rung) of area script copied according to step 1.

From the menu, select "Edit (E)" → "Paste (P)".



Status of copy complete

Other procedure
- Ctrl key + V key
- From the right-click menu, select "Paste(P)"

Label Programming and Label Array

This section describes label program method and precautions, and label array application method, etc.

Point **Variables can be used only with KV STUDIO Ver11.0 or later, and the compatible model is only KV-8000 Ver.2.0 or later.**

Label Programming

Pre-setting the label for the device used in the program can allow label programming. KV STUDIO can use variables for programming in addition to devices. Programming using variables has the following advantages.

- Since the predefined names can be used for programming, the program readability is improved.
- The required size of the variable is automatically assigned to the work area of the CPU unit in combination with the defined variable. Therefore, there is no need to care about the assignment.
- The device map is no longer created in advance, so that the debugging efficiency, etc. can be improved, thus reducing the programming man-hours.
- Variables are automatically assigned even when a program is copied between different projects. Unlike the devices, the device map need not be changed before copying, which makes the program more reusable.
- Since the touch panel display of VT5 series also supports the variables of the KV-8000 series, there is no need to change the device number, etc. on the touch panel display side when changing a device number of PLC like the devices.

Global and local variables

There are 2 types of variables: global variables and local variables. Their features are as follows.

Details	Global variables	Local variables
Entity	The same entity is processed jointly throughout the project. When the same global variable name is written in different programs, the same entity is referenced.	These are variables that can be used only in the program. Different entities are referenced even if the same local variable name is written in different programs.
Assignment	There are two types: manual assignment to devices and automatic assignment from the CPU unit to the work memory.	The CPU unit automatically assigns the variables to the work memory. They cannot be assigned to a device.
Usage	The variables can be used for communication areas with external equipment such as touch panel display. Also, the expansion unit devices such as operation enable can be programmed using variables by assigning the expansion unit devices in advance.	Since no conflict occurs between different programs even if the same variable name is used, they can be used as areas for operation.

Variables and data formats at a glance

When using variables, it is necessary to define the data format in advance. The CPU unit automatically allocates the necessary data size to the work area *1 according to the defined data format.
The work area sizes and variable specifications that can be used with KV-8000 are as follows.

Item		Details
Work area size	bit	64000 point
	word	589824 words
	Timer	4000 point
	Counter	4000 point
Variable specifications	Definable number	100000 point*2
	Maximum length of variable name	128 character*3

*1 If it is defined as a global variable, it can also be manually assigned to a device of the CPU unit.
*2 Definable number means the variable name that can be registered in the variable edit window. For example, if one array type variable 10000 elements is defined, it is counted as 1 point.
*3 The character code of the variable name is UTF-16.

Point

During automatic assignment, BOOL type variables may not be assigned to the start of the channel. When using a BOOL type variable for an operand that must specify the start of the channel, specify this variable assigned to the start of the channel with one of the following methods.
- **Create a local/global structure type variable with a BOOL type member having a bit offset of 0.**
- **Create a global variable with the start of the channel of the global device specified as the device to assign.**

The data types and data sizes of variables that can be used with KV-8000 are as follows.

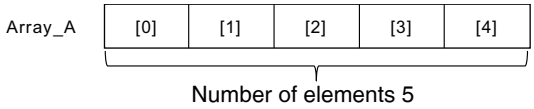
Data type		The amount of data (Byte unit)
Type name	Description	
BOOL	bit	1
UINT	1 word unsigned integer	16
INT	1-word signed integer	16
UDINT	2 word unsigned integer	32
DINT	2-word signed integer	32
REAL	Single precision floating point type	32
LREAL	Double precision floating point type	64
STRING*1	Text string	(Number of characters *2+1) × 8 rounded up to an integral multiple of 16
TIMER	Timer	-
COUNTER	Counter	-
ARRAY	Array	Data size in data format × array size
Structure	-	Based on definition*3
Function block	Function block	Based on definition*3

*1 The character code processed in STRING type follows the language set in the project language settings. For details on project language settings, refer to "KV STUDIO User's Manual".
*2 The number of characters is the number of bytes up to NULL. For double-byte characters such as Shift-JIS, the number of characters is calculated as 2 with 1 character.
*3 Structures and function blocks are data formats that use multiple data formats. Therefore, the data size is determined according to the setting contents.

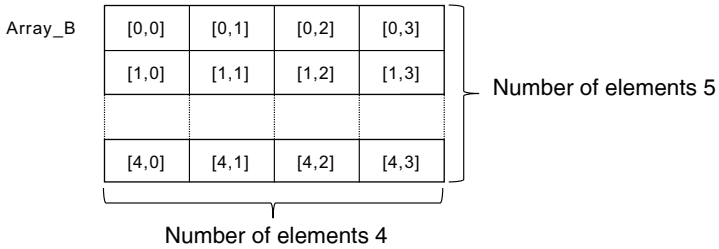
Array

Array is a data type that defines a collection of data of the same data type with 1 name. Each data that constitutes an array is called an element. When using an array, the data type, number of elements and number of dimensions should be set in addition to the variable name. An array with n dimensions is called an n-dimensional array. An example of 1-dimensional array to 3-dimensional array is shown below.

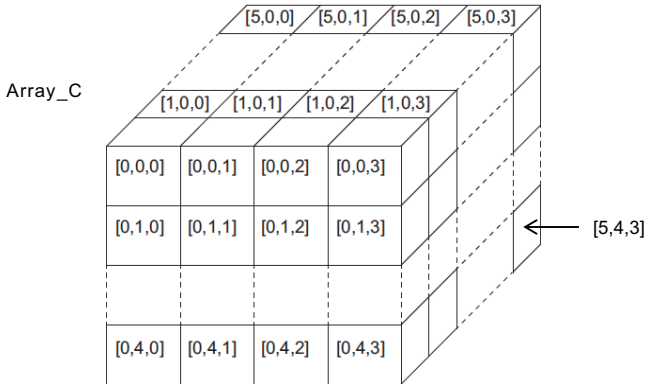
- 1-dimensional array　Variable name: Array_A　Number of elements: 5



- 2-dimensional array　Variable name: Array_C　Number of elements: 6×5×4



- 3-dimensional array　Variable name: Array_C　Number of elements: 6×5×4



■Writing of array type variables

● Specification of element of array type variable

When an array element is specified and used in a program, the combination of the array name and the subscript of [　] is used. For an array with 2 or more dimensions, specify the element numbers separated by "comma (,)". For example, when using the elements [5,0,3] of the 3-dimensional array Array_C, write "Array_C[#5,#0,#3]" in the program as follows. (# can be omitted)

```
Array_C[#5, #0, #3] = #0
```

● Specification of entire array type variable

All elements that constitute an array are called "the entire array". When an entire array is specified in a script, only the variable name is written. For example, when specifying the entire array of a variable called Array_A, write "Array_A".

```
ACNT(Array_A, #2)
```

Array specifications

Item	Details
Maximum number of elements per dimension	589823
Maximum number of dimensions	8
Maximum data size that can be occupied per array	589823 words
Data types that can be specified	BOOL / UINT / INT / UDINT / DINT / REAL / LREAL / STRING / Function block / Structure
Subscripts that can be specified*1	Device / constant / variable / operation formula*2
Data types that can be used for subscript	UINT / INT / UDINT / DINT

*1 Specify 0 or a positive integer for the subscript. If any other number is specified, a conversion error or operation error will occur.

*2 Only KV scripts can use operation formulas as subscripts. Operation formulas cannot be used as subscripts in ladder programs.

Point

- **When an element specified exceeds the specifiable range, such as specifying Array_A[10] (11th element) for an array called Array_A set with the number of elements "10", the value in an unintended area may be referenced. Especially when using variables or expressions as subscripts, the ACNT function or WSIZE function should be executed before referencing the array to obtain the number of elements in the array or the data size of the entire array, and avoid referencing elements beyond the specifiable range.**
  "ACNT function", page 6-60
  "WSIZE function", page 6-62
- **If a device is specified as the subscript of an array, it is processed as a 16-bit unsigned integer.**
- **Bit devices (R, MR, etc.), word device bits and BOOL type variables cannot be specified for the subscript.**
- **Depending on the KV STUDIO version, array data types are expressed as follows:**
  Example: UINT type array with three elements
  Ver. 11.20 or later: ARRAY[0..2] OF UINT
  Ver. 11.10 or earlier: UINT[3]

Structure

Structure is a data type that defines a collection of data of multiple data types with 1 name. Each data that constitutes a structure is called a member. When using a structure, the names of the members that constitute the structure and the data types of the members should be set in addition to defining its name. The structure has the following advantages.

- Registering the structure members and data types in advance makes it easier to create variables with the same structure.
- All members can be monitored in batch by registering a structure variable when monitoring in the registration monitor window, etc.

The following is an example of using the following information of a motor axis in a program.

Current coordinate (CurCoordinate)
Current speed (CurSpeed)
Axis control (Control)
Axis error (Error)
Axis error code (ErrorCode)

If multiple axes are used without a structure, variables with the same content except for the axis number should be registered as shown below.
When using a structure, the data type should be set to the same structure only when using multiple axes after defining the data structure of axis information as a structure in advance.

<When not using a structure>

Variable name	Data type
Ax1_CurCoordinate	DINT
Ax1_CurSpeed	DINT
Ax1_Control	BOOL
Ax1_Error	BOOL
Ax1_ErrorCode	UINT
Ax2_CurCoordinate	DINT
Ax2_CurSpeed	DINT
Ax2_Control	BOOL
Ax2_Error	BOOL
Ax2_ErrorCode	UINT

<When using a structure>

Variable name	Data type
Ax1	AxisInfo
Ax2	AxisInfo

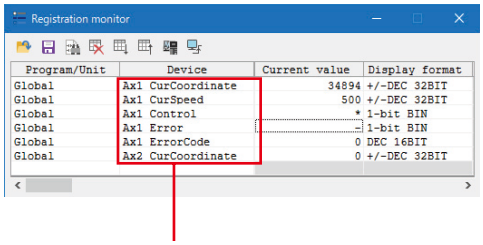
Definition of "AxisInfo" type data

Member name	Data type
CurCoordinate	DINT
CurSpeed	DINT
Control	BOOL
Error	BOOL
ErrorCode	UINT

Reuse the pre-set structure

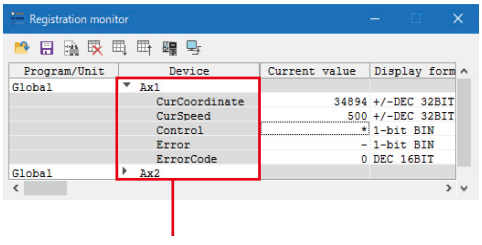
Also, if no structure is used when monitoring in KV STUDIO, it is necessary to manually enter all variables, but if a structure is used, the members in the structure can be monitored in batch simply by entering a variable name.

<When not using a structure>



All variables to be monitored must be entered.

<When using a structure>



When using a structure, the members can be monitored in batch simply by entering a variable name.

■Writing of structure type variables

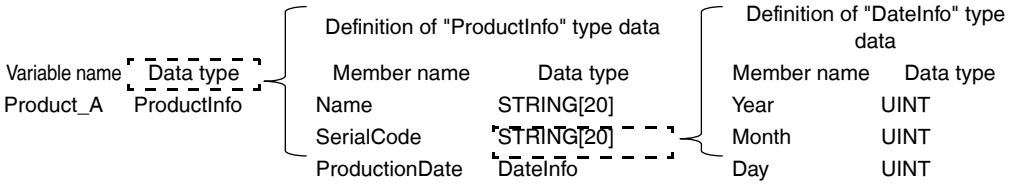
This section describes how to write structure type variables when using them in a program.

●Specification of member of structure type variable

When using a member of a structure type variable in a program, write "Structure type variable name + Period (.) + Member name". For example, "Ax1.CurSpeed" is written to reference a member called CurSpeed of an AxisInfo type variable called Ax1.

```
DM0.L = Ax1.CurSpeed
```

A structure can also be specified for the data type of a member of a structure type variable. In the following example, a structure type variable called "ProductionDate" is nested in the structure type variable Product_A of "ProductInfo".



When Day of Product_A is specified with a program, write "Product_A.ProductionDate.Day".

```
Product_A.ProductionDate.Day = #31
```

Point **Structures can be nested up to 8 stages.**

●Specification of entire structure type variable

All members that constitute a structure type variable are called "the entire structure". When an entire structure is specified in a program, only the variable name is written. For example, when specifying the entire structure of a structure type variable called Product_A, write "Product_A".

```
WSIZE(Product_A)
```

Point

- **When writing a value to the entire structure using the FMOV function, etc., the value in an unintended area may be referenced if a value larger than the number of data in the entire structure is specified as the number of move data. When referencing the entire structure with the FMOV function, etc., the WSIZE function should be executed before the function execution to obtain the data size of the entire structure.**
  "WSIZE function", page 6-62

Structure specifications

Item	Details
Maximum number of structures defined	800
Maximum number of members per structure	500
Maximum number of stages for nesting of structures	8*1
Data types that can be specified as members	BOOL / UINT / INT / UDINT / DINT / REAL / LREAL / STRING/ Array / Structure

*1 When stAAA.member is written, the number of nesting stages is 1. When stAAA.stBBB.member is written, the number of nesting stages is 2.

Variable Setting

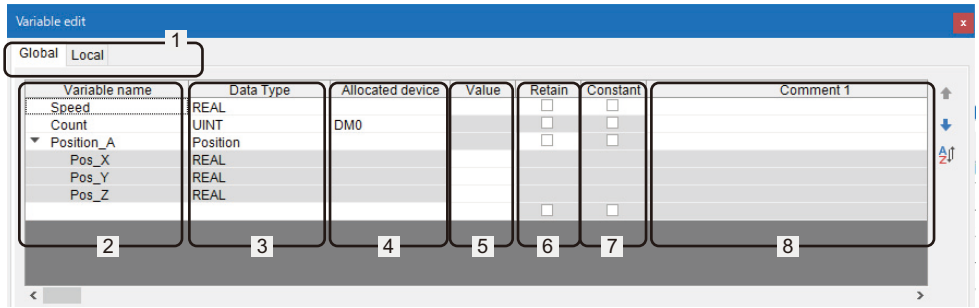
This section describes the procedure from defining variables to using them in a ladder program.

■How to set from the variable edit window

Global and local variables can be set with this method.

1 Select "Display (V)" → "Variable Edit Window (L)" to open the "Edit Settings" window.

Other procedure Double-click "Variable" in the workspace



No.	Item	Details
1	Global / Local tab	Switches between the global variable edit tab and local variable edit tab. "Global and local variables", page 2-25
2	Variable name	Enter the variable name (128 half-width characters / 128 full-width characters).
3	Data type	Select the data type of the variable from below. BOOL / UINT / INT / UDINT / DINT / REAL / LREAL / STRING / TIMER / COUNTER / ARRAY / Structure / Function block
4	Assignment Device	Set when manually assigning variables to devices of the CPU unit. Assignment is not mandatory. This item does not exist in the local variable edit tab.
5	Value	When the "Constant" item is checked: Specifies the initial value of a variable. When the constant is checked, the value is fixed and cannot be changed from the program. When the "Constant" item is not checked: Sets the initial value of a variable. The value can be changed in the program. When the assigned device is set, the initial value cannot be set in the variable edit window. The initial value of the assigned device should be set in the device initial value setting. For details, refer to the CPU Unit User's Manual.
6	Hold	Checked variables hold their values even when the power is turned off. The maximum number of variables that can be held is 30000 including bits, words and counters. When the assigned device is set, the power off holding should be performed for the set device since it is not included in the above 30000 devices. Variables without Retention checked will be initialized to the value set in "Value" when switching from PROGRAM mode to RUN mode. It is initialized to 0 when not set. For details, refer to the CPU Unit User's Manual.
7	Const	Checked variables are processed as constants. Variables set as constants are included in the maximum of 100000 definitions, but they do not consume the work area.
8	Comment	Sets the variable comment (32 half-width characters / 32 full-width characters). The set comment is displayed in the KV STUDIO ladder edit window. Comments 1 to 8 can be set for 1 variable.

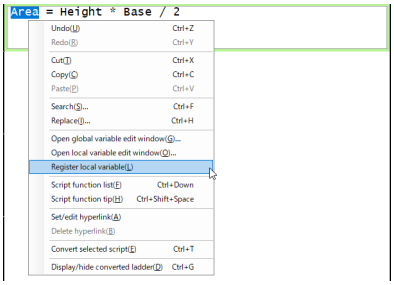
■How to set unregistered variables imported in a script

With this method, only local variables can be set.

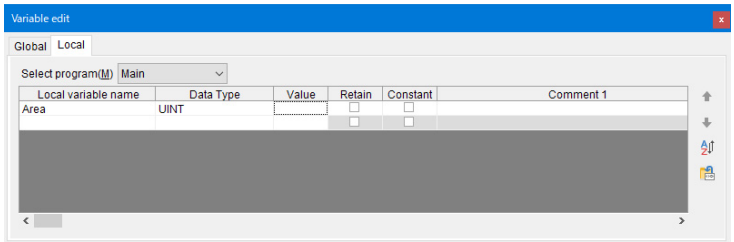
1 Write a script using variables.

```
Area = Height * Base / 2
```

2 With the string to be registered as a local variable in the script selected, select "Register Local Variable (L)" from the context menu.



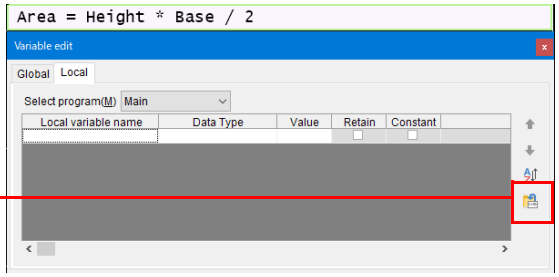
3 The variable edit window will be displayed with the variable name imported. Set the data type.



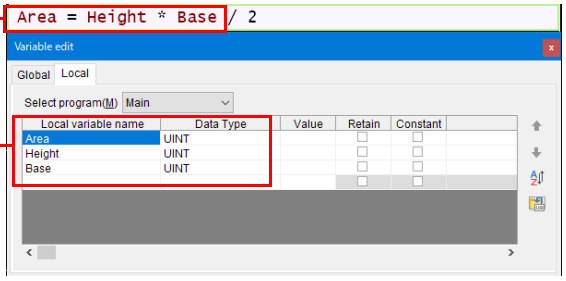
● Batch registration of local variables

This section describes how to import unregistered variables in a module in batch as local variables. Click in the variable edit window when there are unregistered variables. All unregistered variables existing in the target module are imported as local variables. Set the data type.

Click the variable import icon.



Unregistered variables are imported in batch, and the display color of registered variables changes.



Point

All variables in the module will be registered after performing batch registration. Therefore, errors are also registered as variables, so that errors that originally occur during conversion may be converted normally.

Example **DN1000=10 ....Before batch registration: during script conversion, error occurs.**

....After batch registration: error will not occur during script conversion.

It is recommended to confirm the registered contents in the variable edit window after batch registration.

Precautions for using variables

This section describes restrictions and precautions when using variables.

● About variable suffix

Variables can be written without adding a suffix when writing them in a script since the data type is set in the variable edit window. However, they can also be added with a suffix like the devices. Type declaration by TYPE is also possible. Their priority is as follows.

Data type < Type declaration < Suffix

low Order of priority high

The suffix that can be specified is limited by the data type set when registering the variable.

Item	Global variables (Manual assignment)									Global variables (Automatic assignment) / local variables								
	.B	.U	.S	.D	.L	.F	.DF	.T	Omitted	.B	.U	.S	.D	.L	.F	.DF	.T	Omitted
BOOL	○	×	×	×	×	×	×	×	.B	○	×	×	×	×	×	×	×	.B
UINT	○	○	○	○	○	○	○	○	.U	○	○	○	×	×	×	×	×	.U
UDINT	○	○	○	○	○	○	○	○	.D	○	○	○	○	○	○	×	×	.D
INT	○	○	○	○	○	○	○	○	.S	○	○	○	×	×	×	×	×	.S
DINT	○	○	○	○	○	○	○	○	.L	○	○	○	○	○	○	×	×	.L
REAL	×	○	○	○	○	○	○	×	.F	×	○	○	○	○	○	×	×	.F
LREAL	×	○	○	○	○	○	○	×	.DF	×	○	○	○	○	○	○	×	.DF
STRING	×	×	×	×	×	×	×	○	.T	×	×	×	×	×	×	×	○	.T
Timer	○	○	○	○	○	×	×	×	.B	○	○	○	○	○	×	×	×	.B
Counter	○	○	○	○	○	×	×	×	.B	○	○	○	○	○	×	×	×	.B

○: Suffix can be specified. However, if a suffix different from the data type is specified, a warning will be issued during conversion.

×: Suffix cannot be specified.

● Variable range check

The range check will be performed when using KV-8000 to avoid the result of function execution referencing an area that exceeds the variable range. The content of the range check differs depending on the variable settings.
The following is an example of using a variable called Array_A with the number of elements set to 10.

For global variables (automatic assignment) / local variables
When a constant exceeding the number of elements is specified, such as Array_A[20], an error will occur during script conversion.
When other than a constant is specified for the number of elements, such as Array_A[idx], no error occurs during script conversion. If the value of idx exceeds the range of 0～9 during program execution, a calculation error will occur during instruction execution.

For global variables (manual assignment)
The following is an example in which the start of Array_A is assigned to DM0.
When a constant exceeding the number of elements is specified, such as Array_A[20], an error will occur during script conversion.
When other than a constant is specified for the number of elements, such as Array_A[idx], a warning will be issued during script conversion, but no error will occur. If the value of idx exceeds the range of 0～9 during program execution, the target device will be processed, and no calculation error will occur. For example, when idx=19, Array_A[idx] will be processed as DM19.

● Other precautions

- The global variable can be set to the same variable name as the local variable, but it operates as a local variable when used.
- Variable names cannot use the strings such as reserved words.
  "List of Characters That Cannot Be Used", page A-54
- Index modification cannot be performed for global variables and local variables automatically assigned.
- Index modification and indirect specification cannot be performed for variables of array type and structure type.
- When "Constant" is checked in the [Variable Edit] window, index modification and indirect specification cannot be performed.

Label Programming and Label Array

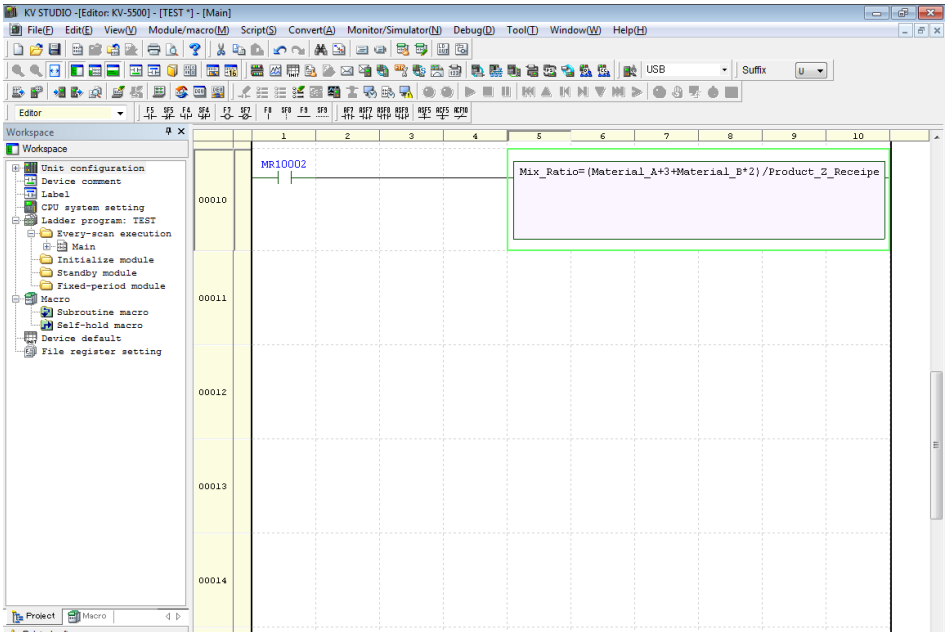
This section describes label program method and precautions, and label array application method, etc.

Important

- **Only labels can be used in projects where the model setting is other than KV-8000 (CPU function version 2.0 or later).**
  When the model setting is KV-8000 (CPU function version 2.0 or later), variables can be used instead of labels.
  "Label Programming", page 2-25
- **KV-1000 cannot use local labels and label arrays. Only global labels can be used.**

Label Programming

Pre-setting the label for the device used in the program can allow label programming.



Label can be divided into two types: global label and local label.

Point

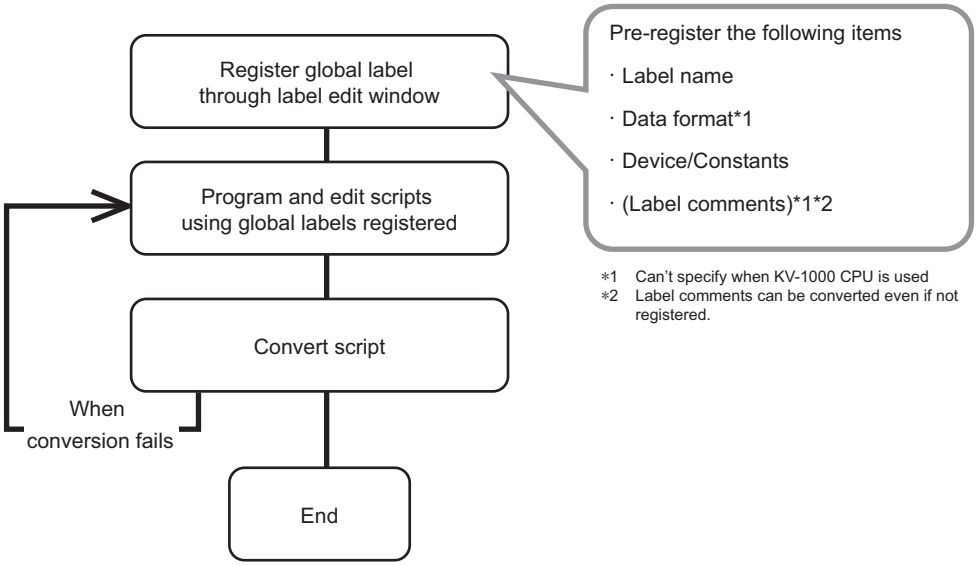
- **Only labels can be used in projects where the model setting is other than KV-8000 (CPU function version 2.0 or later).**
  When the model setting is KV-8000 (CPU function version 2.0 or later), variables can be used instead of labels.
  "When conversion error occurs", page 2-55
- **Local label cannot be used when CPU unit is KV-1000.**

■ Global label

Global label refers to the common label in whole project.

1 global device must be assigned with 1 global label.

● Programming flow using global label

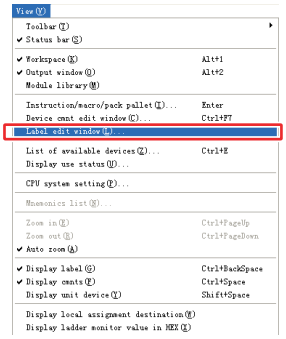


● Register global label

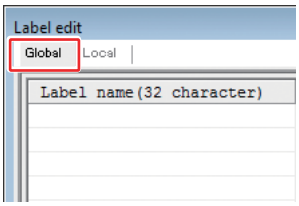
The global label registration steps are described as follows.

1 From the menu, select "View (V)" → "Label edit window (L)". Label edit window appears.

Other procedure
- Double-click "Label" in workspace.
- Click the tool bar in the Script Edit window.



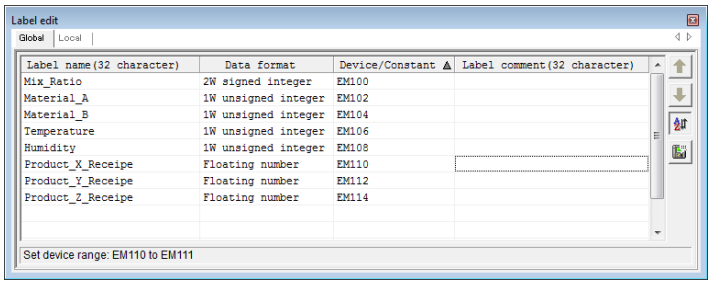
Click "Global" label when using KV-8000/7500/7300/5500/5000/3000 or KV Nano series CPU.



2 Register global label to be used.

For label editor window, see 📖 KV STUDIO User's Manual.

When selecting CPU KV-8000/7500/7300/5500/5000/3000 or KV Nano Series

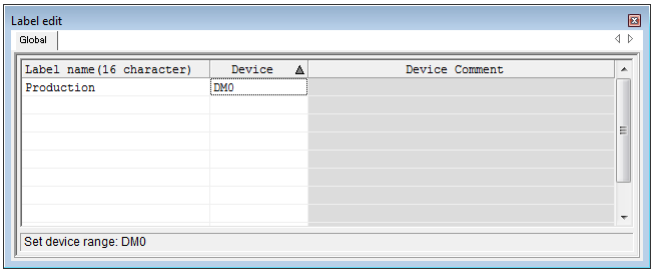


Item	Description
Label name	Set within 32 characters (or single-byte 32 characters, double-byte 16 characters for KV-5500/5000/3000/1000 or KV Nano Series).* 1
Data type	Specify label type. Click input box, select label type from the drop down menu displayed.
Device/constant	Specify the device or constant to which global label is assigned. The range of device and constant that can be specified varies with Data Format. 📖 "About label type", page 2-44
Label comment	Specify comment in global label.

The character and character string that cannot be specified also available, such as reserved word of KV STUDIO, etc.

📖 "Precautions when using label", page 2-44

When selecting CPU KV-1000



Item	Description
Label name	Set within single-byte 16 character, double-byte 8 character. 1
Device	Specify the device to which global label is assigned. 2
Device Comment	Display the device comment assigned. Cannot be edited.

The character and character string that cannot be specified also available, such as reserved word of KV STUDIO, etc.

📖 "Precautions when using label", page 2-44

*2 There are also devices that cannot be specified.

📖 "KV STUDIO User's Manual"

Point Labels can be set for global devices. But labels cannot be set for local devices.

3 Use global label registered to make script program.

```
00001  1|Material combinatin ratio=((materialA*10)+(materialB*3))*temperature/humidity )*productX
       2|
       3|
```

Character color of label registered in script edition will be changed.

Convert script after programming completed.

```
00001  Ratio = (((MaterialA * 10) + (MaterialB * 3)) *Temperature / Humidity) * ProductX
00002  Ratio = (((MaterialA * 10) + (MaterialB * 3)) *Temperature / Humidity) * ProductX
00003  CR2002 | Always ON | Humid.. LDA.S | EXT.S | @VM00.. STA.L | mater.. LDA.S | EXT.S | +3 MUL.L | @VM00.. STA.L | mater.. LDA.S
00004  EXT.S | +10 MUL.L | @VM00.. ADD.L | @VM00.. STA.L | Tempe.. LDA.S | EXT.S | @VM00.. MUL.L | @VM00.. DIV.L
00005  @VM00.. STA.L | Produ.. LDA.S | EXT.S | @VM00.. MUL.L | Ratio STA.S
```

● Example for programming using global label

```
00003  '---FOR-NEXT TXT---
       'store corresponding version information to DM02.
       '-----------------------------------------------------------------------------
       'X=EM100.U  Y=EM102.U    A=EM104.F  B=MR200 C=MR201

       FOR X = 0 TO 1000 STEP 2      '"2"
           FOR Y = 0 TO 200 STEP 1   '"1"
               A = Y X X/3
               IF B = ON THEN
                  C = ON
                  BREAK
               ENDIF
           END IF
         NEXT
       NEXT
00004
```

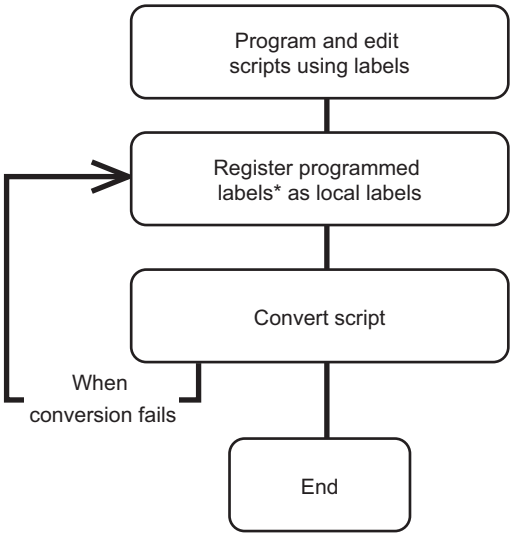
■ Local label

Local label refers to label that can be used when CPU is KV-8000/7500/7300/5500/5000/3000 or KV Nano Series, only valid in every module, and every module shall be set.
For the device (VB, VM) whose local label using system area (working memory) of KV-8000/7500/7300/5500/5000/3000 or KV Nano Series, local label can be also set even if service condition of the device is not known.

Point **When CPU is KV-1000, local label cannot be used.**

Reference
The work memory is configured as follows.
KV-8000 VM: 589824 words VB: 64000 points (4000ch)
KV-7500/7300/5500/5000/3000 VM: 50000 words VB: 16384 points (2024ch)
KV Nano Series VM: 9500 words VB: 8192 points (512ch)
The assignment status is managed by KV STUDIO.
"About Script Work Memory (Work Devices)", page 2-74

● Programming flow using local label

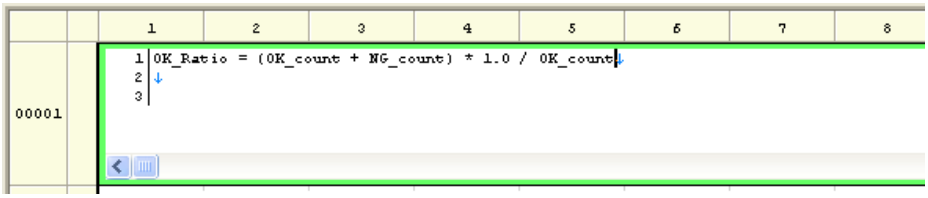


* Before script programing, local label can be also registered in advance.

● Register local label

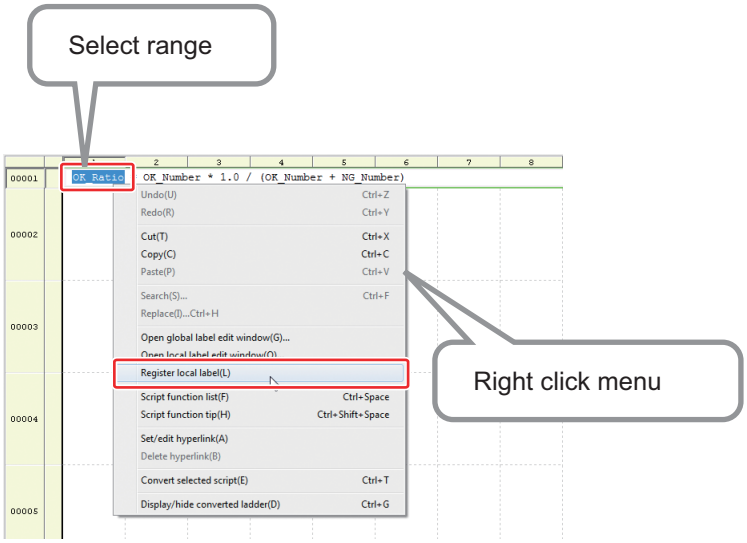
The registration steps of local label are described as follows.

1 Program script with labels.

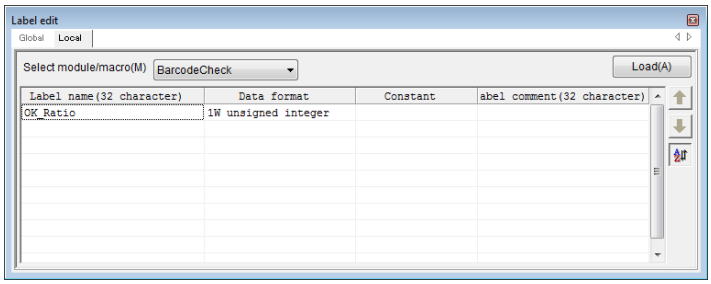


Character color is not changed due to label not registered.

2 **Under script editing mode, select the label range registered as local label, select "Register local label (L)" from right-click menu.**



3 **Set data format in label edit window.**



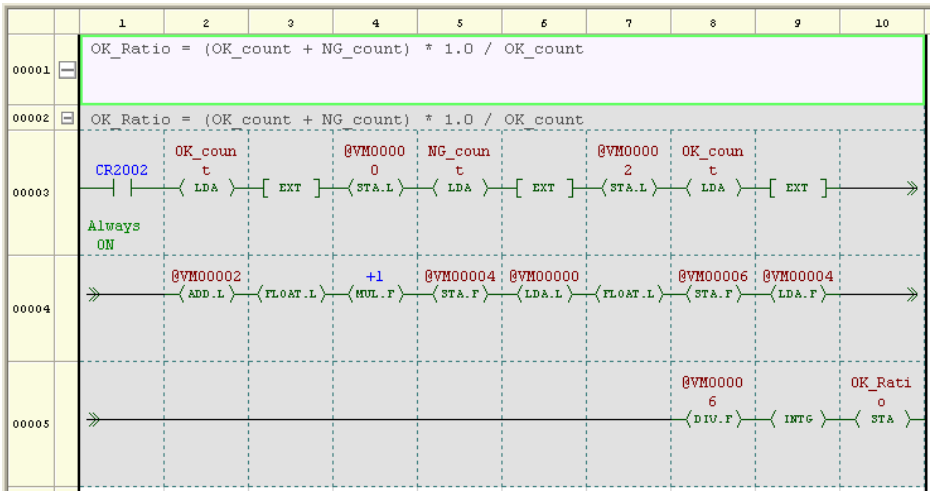
Item	Description
Label name	Automatically stored after label range selection.* 1
Data type	Default to "1-word unsigned integer". Data format can be changed by clicking drop-down menu. "About label type", page 2-44
Const	During constant assignment of local label, constant can be specified. Constant is not assigned if blank.
Label comment	Specify comment to be added to local label.

*1 The character and character string that cannot be specified also available, such as reserved word of KV STUDIO, etc.
"Precautions when using label", page 2-44

Reference

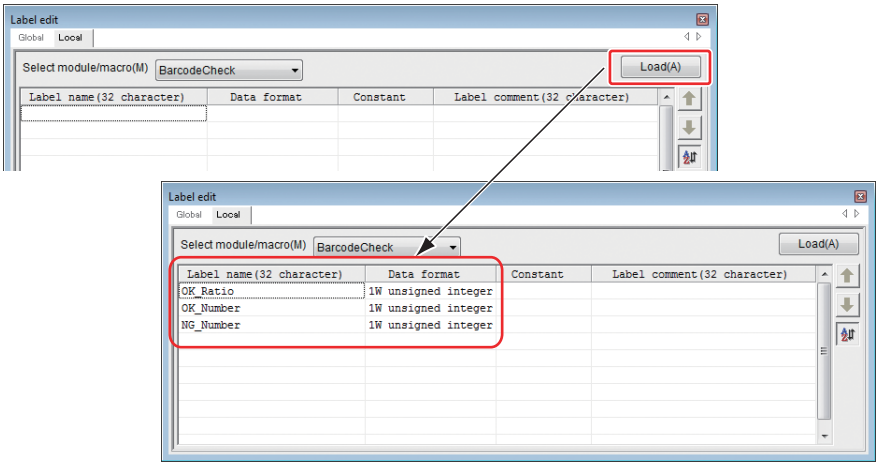
All labels of the module being edited can be registered in batch as local label.
"Batch register of local label", page 2-43

4 Convert after all local labels are registered.



● Batch register of local label

Via editing module, register all labels unregistered to local label in batch.



Data Format of every local label changes to "1-word unsigned integer", which, can be separately changed.

Point

During batch registration, all labels in module are registered. So, the error flag is also registered as label, the error occurring in script conversion may appear in normal conversion.

Exampl DN1000=10 ..... Before batch registration: during script conversion, error occurs.

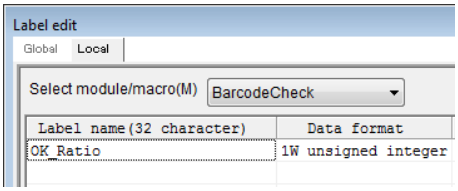
After batch registration: error will not occur during script conversion.

After batch registration, it is recommended to check content registered via label edit window.

■ Precautions when using label

● About label type

When CPU is KV-8000/7500/7300/5500/5000/3000 or KV Nano Series, data format (type) can be specified in label.



Data format can be selected as follows.

Item		Processing data range
bit	BOOL	O(OFF,FALSE),1(ON,TRUE)
1-word Unsigned integer	UINT	0~65535
2-word Unsigned integer	UDINT	0~4294967295
1 word signed integer	INT	-32768~+32767
2 word signed integer	DINT	-2147483648~+2147483647
Single precision floating point type Real number	REAL	-3.4E+38 ≦ N ≦ -1.4E-45 N ＝0 1.4E-45 ≦ N ≦ 3.4E+38 (For the KV Nano Series, an absolute value in a range of 1.4E-45 to 1.2E-38 cannot be used.)
Double precision floating point type Real number	LREAL	-1.79E+308 ≦ N ≦ -2.23E-308 N＝0 2.23E-308 ≦ N ≦ 1.79E+308
Text string	STRING	No more than 32 characters (single-byte character)
Timer*2	---	0（OFF,FALSE）,1（ON,TRUE）
counter*2	---	0（OFF,FALSE）,1（ON,TRUE）
Array	---	Specify via "Array size" dialog box "Label Array", page 2-46

*1
- The ".DF" suffix can be used only with the KV STUDIO Ver.6.00 and higher versions.
- The ".DF" suffix can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units that have CPU function version V2.0 or higher and with KV Nano Series base units.

"KV-5500/5000/3000 CPU Function Version", page 4

*2 If a .D suffix is added to the label, it will be handled as the current value.

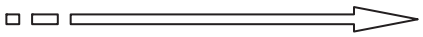
● About label suffix

During label registration, type of data (Data Format) is set via label edit window, so, programming can be also performed even if suffix not added. But, the programming with additional suffix can be also performed as same as other device.

Reference

Type declaration can be also conducted based on TYPE. The relationship between "Data format" set in the label editing window, "Type declaration" based on TYPE, and "Direct designation" of suffix is as follows.

Data format < Type declaration < Direct specifying



Low PRI High

When registering labels, the suffix which can be specified based on data type (data format) in label edit window is limited as follows.

Item	Global label									Local label								
	.B	.U	.S	.D	.L	.F	.DF	.T	Omitted	.B	.U	.S	.D	.L	.F	.DF	.T	Omitted
bit	○	×	×	×	×	×	×	×	.B	○	×	×	×	×	×	×	×	.B
1-word Unsigned integer	○	○	○	○	○	○	○	○	.U	○	○	○	×	×	×	×	×	.U
2-word Unsigned integer	○	○	○	○	○	○	○	○	.D	○	○	○	○	○	○	○	×	.D
1 word signed integer	○	○	○	○	○	○	○	○	.S	○	○	○	×	×	×	×	×	.S
2 word signed integer	○	○	○	○	○	○	○	○	.L	○	○	○	○	○	○	○	×	.L
Single precision floating point type real number	×	○	○	○	○	○	○	×	.F	×	○	○	○	○	○	○	×	.F
Double precision floating point type real number	×	○	○	○	○	○	○	×	.DF	×	○	○	○	○	○	○	×	.DF
Text string	×	×	×	×	×	×	×	○	.T	×	×	×	×	×	×	×	○	.T
Timer	○	○	○	○	○	×	×	×	.B	○	○	○	○	○	×	×	×	.B
Counter	○	○	○	○	○	×	×	×	.B	○	○	○	○	○	×	×	×	.B

○: Suffix can be specified (but, in case of suffix specified different from Data Format, prompt is given during conversion)

×: Suffix cannot be specified.

Reference When programming the suffix that cannot be specified, error will appear during script conversion.

● About the bit of global label

When bit is specified in Data Format of global label, different types of assignment device will use different suffixes.

Device to be assigned	Suffix that can be specified
R, B, MR, LR	.U/.S/.D/.L./.F/.DF/.T*/.B
Bit specifying of word device (DM1000.12, etc.)	.B

* When specifying ".T", please specify the leading device of channel to assign the bit device.

● Other precautions

- Same label name can be set for global label and local label. It operates in form of local label.
- Text string such as reserved words can not be used in label name.
  "List of Characters That Cannot Be Used", page A-54
- When CPU unit is KV-8000/7500/7300/5500/5000/3000 or KV Nano Series, label is double-byte character/single-byte character sensitive but is not case-sensitive.
- Index modification cannot be conducted for local label.
  "Index modification", page 3-55
- Index modification and indirect specifying cannot be conducted for label array.
  "Label Array", page 2-46
- When constant is assigned in label, index modification and indirect specifying cannot be performed.
- Label and word device bit cannot be specified.
  "Processing Word Device as Bit Device", page 3-46

Label Array

When registering label, label array can be used by specifying "Array" in Data Format via label edit window.

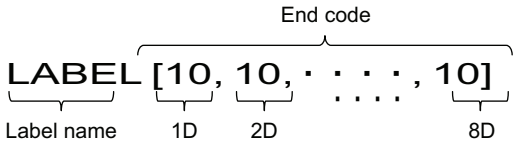
Point

When CPU is KV-1000, label array cannot be used (global label array, local label array).

● Array

Array refers to continuous aggregation of same data type.
KV script represents the array in form of "label + [label end code]", Multi-dimensional array can be realized by taking comma as end code for separating label.
Maximal 8-dimensional label array can be specified.

Denotation Label array is expressed as follows.



Reference For label end code of label array, except integer (DEC, HEX), integer type (.U /.S /.D /.L) device, macro argument (P0 to P9, V0 to V9) and the function with formula and return value can be also programmed.

Exampl: LABEL[1, 2, DM0]
LABEL[1, DM2 * 3, Z01]

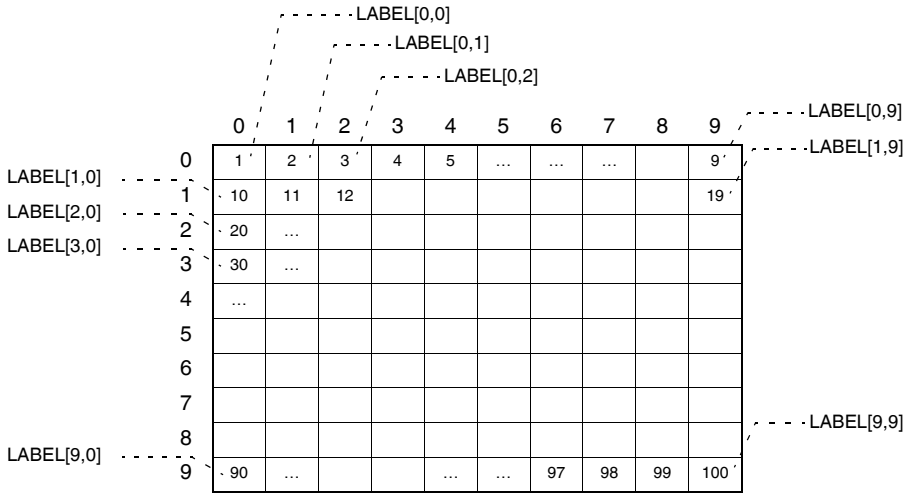
Point

- Index modification cannot be conducted for label array.
- The indirect specifying device cannot be specified in label array.
- When programming devices and formulas in the end-of-label character of the label array, the stored value should not exceed the range of the end-of-label character.

Range Multi-dimensional array: up to 8 dimensions
Label end code range : KV-5500/5000/3000 or KV Nano Series: 1 to 131071
KV-8000/7500/7300: 1 to 589823

Example

The following describes a data storage diagram when the array size of label is registered as "10,10" in label edit window.



When Data Format of label array is selected as "1-word unsigned integer" or "1-word signed integer", in above label array,

10x10 = 100 words

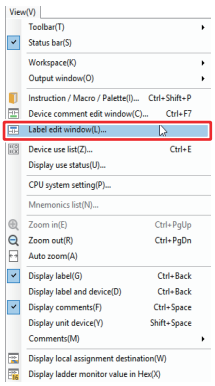
shall be guaranteed as label array area.

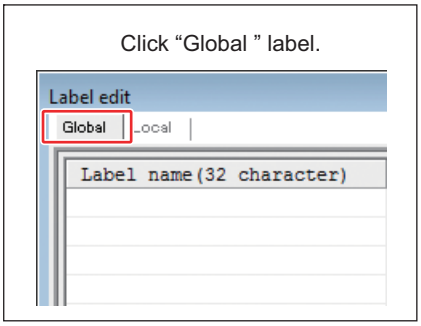
■ Global label array

When registering global label array, "Array" in data format shall be specified.

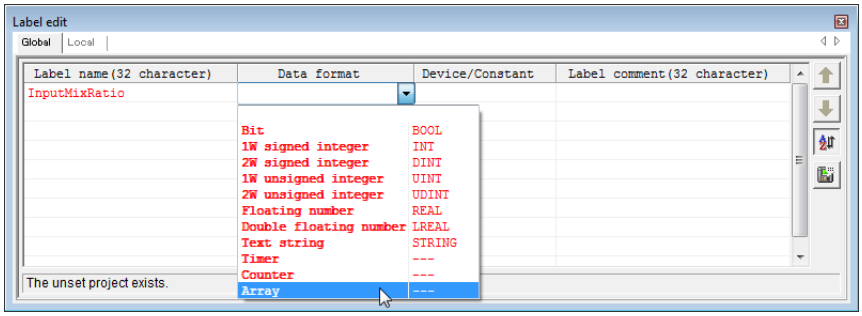
1 From the menu, select "View (V)" → "Label edit window (L)". Label edit window appears.

Other procedure
- Double-click "Label" in workspace.
- Click the tool bar in the Script Edit window.



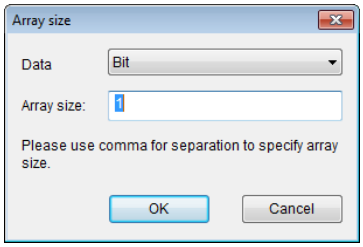


2 **Register global label array name and set data format to "Array".**



Select "Array", the "Array size" dialog box will pop up.

3 **Set the format and size of array data.**



Item	Description
Data type	Specify label type. Click input box, select label type from the drop down menu displayed.
Array size	• Specify array size. Via comma separation, maximal 8 dimensional array can be specified. • Please specify the array size within 1 to 131071.*1 • When major data of array is uncertain, can be replace with "?".*2

*1 Do not exceed the range of global device to be assigned.

*2 When it is converted under the status of "?" reserved, the error will occur.

Reference Please select data format from the following option.

Item	Processing data range
bit	0（OFF,FALSE）,1（ON,TRUE）
1-word Unsigned integer	0~65535
2-word Unsigned integer	0~4294967295
1 word signed integer	-32768~+32767
2 word signed integer	-2147483648~+2147483647
Single precision floating point type	-3.4E+38 ≦ N ≦ -1.4E-45 N ＝0 1.4E-45 ≦ N ≦ 3.4E+38 (For the KV Nano Series, an absolute value in a range of 1.4E-45 to 1.2E-38 cannot be used.)
Double precision floating point type	-1.79E+308 ≦ N ≦ -2.23E-308 N ＝0 2.23E-308 ≦ N ≦ 1.79E+308

4 **Specify global device to be assigned.**

Point **Global label array occupies global Point device whose size has been specified in array.**

■ Local label array

The label pre-registration method of local label array and global label array is identical to registration method after script programming.

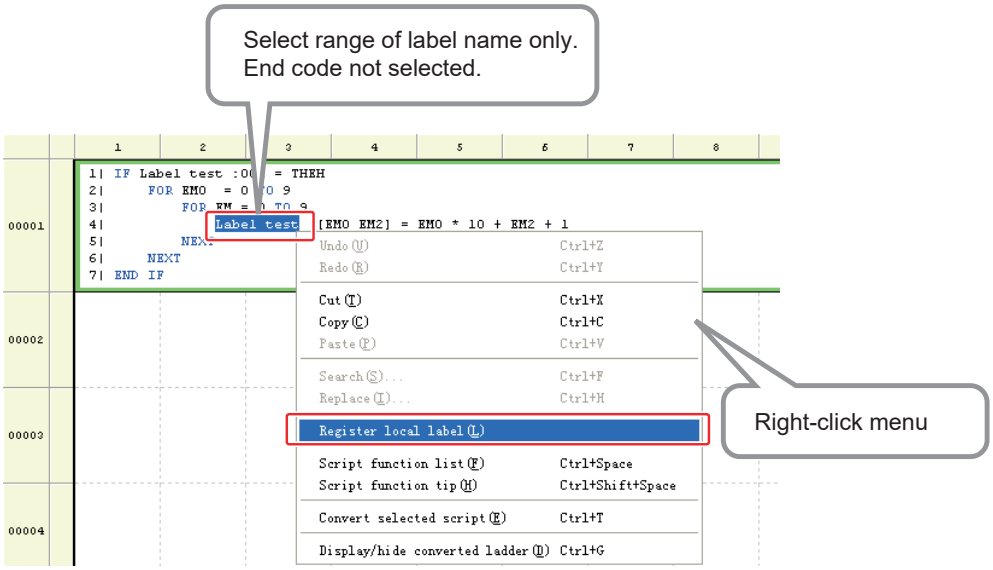
The method of registering local label array after script programming is introduced below.

1 Using local label array to program the script.

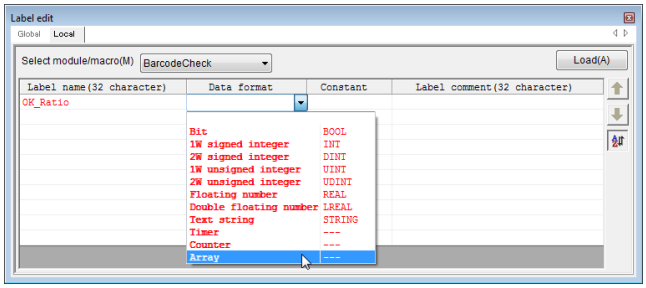
```
00001
1|IF Label test [0'0] = 0 THEH
2|    FOR EM0 = 0 TO 9
3|        FOR EM2 = 0 TO 9
4|            Label test [EM0,EM2] = EM0 * 10 + EM2 + 1
5|        NEXT
6|    NEXT
7|END IF
```

Character color is not changed due to label not registered.

2 Under script editing mode, select the label range as local label register, select "Register local label (L)" from right-click menu.

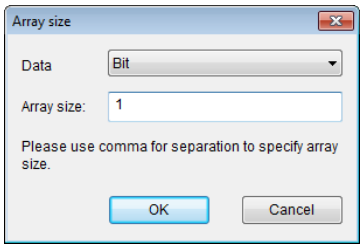


3 Set local label array in label edit window.



Select "Array", the "Array size" dialog box will pop up.

4 Set the format and size of array data.



Item	Description
Data type	Specify type of local label array. Click input box, select label type from the drop down menu displayed.
Array size*	• Specify array size. Via comma separation, maximal 8 dimensional array can be specified. • When major data of array is uncertain, can be replace with "?".

*Do not exceed the range of global device to be assigned.

"Application area of work memories", page 2-74

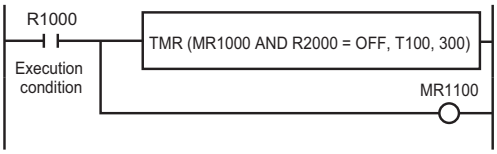
Reference Please select data format from the following option.

Item	Processing data range
bit	0（OFF,FALSE）,1（ON,TRUE）
1-word Unsigned integer	0~65535
2-word Unsigned integer	0~4294967295
1 word signed integer	-32768~+32767
2 word signed integer	-2147483648~+2147483647
Single precision floating point type	-3.4E+38 ≦ N ≦ 1.4E-45 N ＝0 1.4E-45 ≦ N ≦ 3.4E+38 (For the KV Nano Series, an absolute value in a range of 1.4E-45 to 1.2E-38 cannot be used.)
Double precision floating point type	-1.79E+308 ≦ N ≦ -2.23E-308 N ＝0 2.23E-308 ≦ N ≦ 1.79E+308

Precautions when programming script

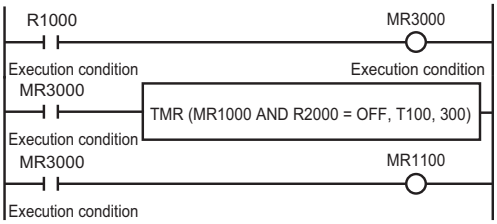
About Parallel Connection of Box Script

"OR" (logical OR) connection cannot be made between Box Script and ladder instruction.
Error will occur during conversion of right program.



● Ladder program equivalent to OR connection

Logic OR connection shall be made via other devices (MR3000 as shown on the right side).



Precautions When Using Script Programming Operation Formula

In the script, all operation formulas are handled as .L (signed 32-bit), .F (single precision floating point type) or .DF (double precision floating point type).

- When all types (U,.D,.S) other than .F and .L and .DF are specified by arithmetic element (device or constant) in operation formula, it will be still processed as .L. Operation result shall be also calculated according to .L .
- If .F is used by arithmetic element (device and constant) in operation formula, all operations before .F will be conducted as .L, all operations and operation result after this will be processed as .F (single precision floating point type).
- If .DF is used by arithmetic element (device and constant) in operation formula, all operations before .DF will be conducted as .L or .F, all operations and operation result after this will be processed as .DF (double precision floating point type).

Point

- **Operation formula shall be processed within the range of .L (signed 32-bit) (-2147483648 to +2147483647). Care shall be taken during calculation exceeds the .L range.**
- **The operation formula expands all arithmetic elements to .L operation, and operational precision will be impaired sometimes when .D (unsigned 32-bit) contained in the operation formula.**
  "Type conversion leads to data loss", page 2-52
- **When impairing precision, "Warning" display can be set.**
  "Setting method of warning level", page 2-53

Script Conversion Output Message

Script program is converted to ladder program, then transferred to CPU unit.
This section describes warning message and error message displayed during script conversion to ladder program.

Warning Message

The message displayed when expected operation result cannot be obtained after execution of Ladder program converted from script program. The warning reason shall be removed as possible.
It will occur when the result type of assignment statement on the right side is different from the left type, the operation formula includes .D, or when the function cannot be executed with the current model.

■ Type conversion leads to data loss

The required data may be not obtained if data loss in the following script.

Programmin DM1000.U = EM2000.S * 10000

- For example, when EM2000 value is "-1", as DM1000 type is suffix 16-bit , correct calculation result cannot be stored. In addition, when EM2000 value is "10", operation result will exceed suffix 16-bit range (0 to 65535), so, the expected correct operation result cannot be applied.

After conversion of above script, output window displays following messages (when setting script warning level to "high").

[Warning 1103]: "=": (signed 32-bit integer type → suffix 16-bit integer type) data may be lost dueto assignment.

In this case, by changing suffix of the left device from .U to .L (i.e. left type and right type are conformable), warning message will be no longer displayed.
" Precautions when programming script", page 2-51

■ Functions that cannot be executed with the current model

For the models below, a warning occurs when a function is executed because the function cannot be executed.

Functions that generate warnings	KV-N14□□	KV-N24□□ KV-N40□□ KV-N60□□	KV-NC32T
WTIME/AJST	Warning	-	-
MWRIT/MREAD/MFREE/MFREEK /MMKDIR/MRMDIR/MDEL/MPRINT/ MREADL/MCOPY/MMOV/MREN/MSTAT	Warning	Warning	-
AWNUM/AWMSG	-	-	Warning

When converted, the following message is displayed in the output window (when the script warning level is set to "High").
[Warning 1008]: "MWRIT": Using a function that cannot be executed on the current model.
For the KV Nano Series, the number of internal positioning axes and the internal high speed counter numbers differ according to the unit. If there is a function that specifies an axis number or counter number that cannot be used, a warning occurs during conversion.
For the axis numbers and high speed counter numbers that can be used with the KV Nano Series, refer to the "KV Nano Series User's Manual".

Script warning level setting

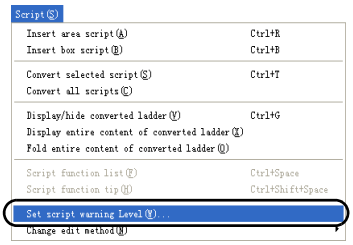
The "Warning" level is displayed in output window when changing script conversion.

■ About warning level

Two levels available: "Error" and "Warning".

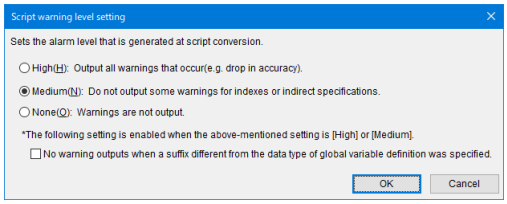
● Setting method of warning level

1 From the menu, select "Script (S)" → "Set script warning level (W)".



Other procedure Select "Tool(T)" → "Option(O)" from the menu, and then select "High", "Medium" or "None" in "Warning level" of "Script" tab.

2 "Script warning level setting" dialog box appears. You can select the warning level.



Item	Description
High	All warning information can be displayed including errors and warnings. Example) when the precision of operation formula is impaired • DM1000.U = DM2000.L (Assign .L type data to .U type device)
Medium	Only error is displayed (for example, when processing out-of-range constants). Example) When the operation both two constants exceeds the range of constants. • DM1000.D= 90000 * 50000 (Assign a value out of constants range (4500000000))
None	No alarm will be displayed.
No warning outputs when a suffix different from the data type of global variable definition was specified.	After the checkbox is selected, the warning that occurs when specifying a suffix different from the data type defined when registering global label is not output. Example) When specifying global label to “1-word signed integer: UINT” and suffix .L is specified via script

Error Message

Error will occur when part of scrip programs can not be converted ladder program and error causes will be displayed in output window.
Errors must be removed. Otherwise, programs cannot be transferred into CPU unit even if conversion successes. You can correct scripts according to error messages displayed in the window that pops up.

■ Error message when script conversion

Maybe due to the following reasons.

● Typing error

- Function name error.
  Example) correct →BLD (DM1000, 10) error →BLF BLF (DM1000, 10)
- Device name error.
  Example) correct →DM1000.D error →DN1000.D
- 2-byte character (Only comments and character constants can use 2-byte characters)
  Example) correct →DM1000.D error →DM1000.D
- 0(zero) and O (letter) are confused.
- 1-byte space and 2-byte space error.
  Example) True →END IF (between half-width spaces) Wrong → END IF (between full-width spaces)

etc.

● Syntax error

- The control statement format is wrong.
  Example) correct →IF R1000 THEN error→IF R1000 THEN
  DM1000 += 1 DM1000 += 1
  END IF END␣␣␣
- BREAK is programmed outside the loop control statement.
- DO is not paired with UNTIL .
- Other general syntax error.

● Function argument error

- Argument error.
  Example) correct →FIND (DM1000.T, DM1200.T)error→FIND (DM1000.T,
- The macro that is executing calling instruction is not programmed.

etc.

● Specify invalid type

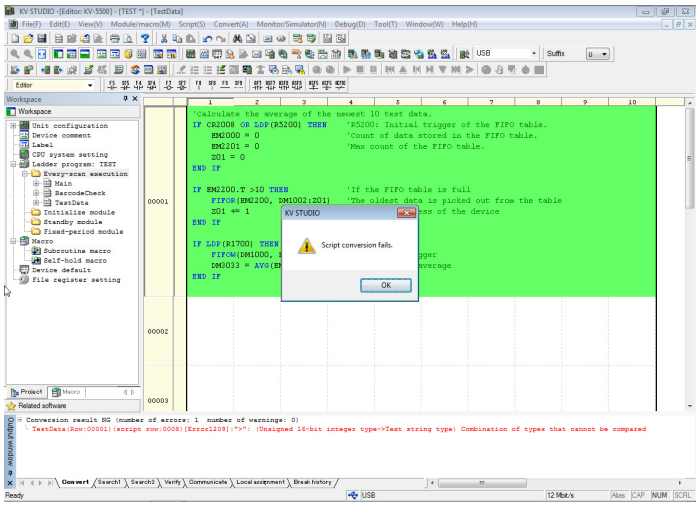
- You want to substitute a character string to a variable other than STRING type or a device without .T.
  Example) correct→DM1000.T = "ABCD"error→DM1000 = "ABCD"
- Word devices and bit device must be specified with arguments specified by text string.
- Numerical constants is assigned to bit device.
- Other invalid type is specified.
- The size of the type specified in local label is larger than the data format when registering.
- A .DF type was specified when using KV-1000.

● Others

- Nonexistent device is specified.
- There are more than 1024 (single-byte) characters in a script rung.
- Array label end code is specified in the label that has not been registered to array.

When conversion error occurs

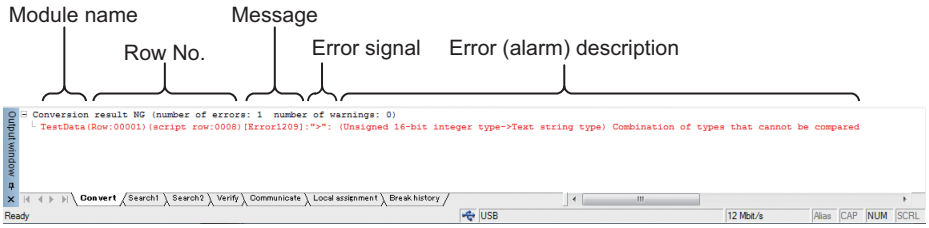
If you convert a script when the code in the script program is invalid due to a mistake such as mistyping a function name, specifying an invalid type (suffix), or mistaking the usage of a control statement, a script conversion error occurs and a dialog box is displayed.



When a conversion error occurs, the error position and contents will be displayed in the window that pops up.

■ Error display in output window

The following error contents will be displayed when script conversion error occurs:



In case of "Error" : Script program cannot be converted to ladder program.
Programs cannot be transferred to PLC under this status.

In case of "Warning" : Script program can be converted to ladder program. Program can be transferred to PLC and executed by Simulator.
But the expected operation result cannot be obtained after executing programs.

■ How to use output window

Error (warning) contents will be displayed in output window when conversion error occurs.
Double-click the highlighted rung and the cursor moves to the error rung in Ladder Edit window.
To correct error according to error contents.

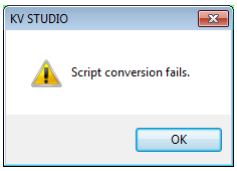
1 To convert the edited scripts.

Select "Script(S)" → "Convert selected script(S)" from the menu.

Other procedure
- Ctrl key + T key
- Select "Edit script(X)" → "Convert selected script(S)" from the right-click menu.

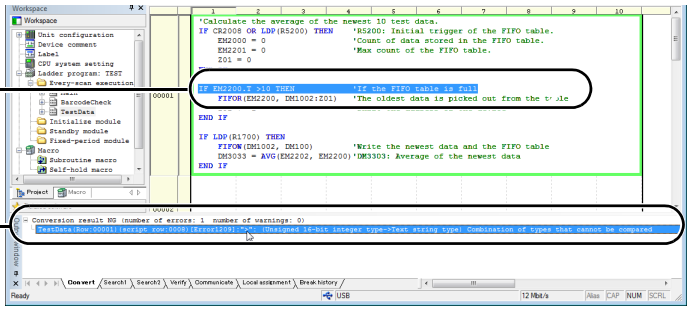
When a conversion error occurs, the following dialog box will appear and error information will be displayed in the window that pops up.

Click "OK" key

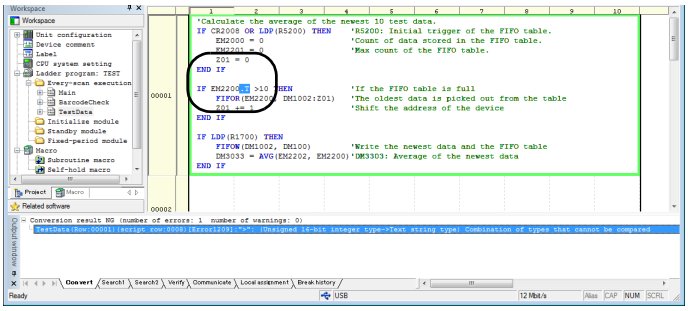


2 Double-click the error contents (1) displayed in the window.

The cursor moves to the rung (2) with programming error and script is turned to editable status.



3 To correct error programming.



4 After correcting, only the corrected scripts are converted.

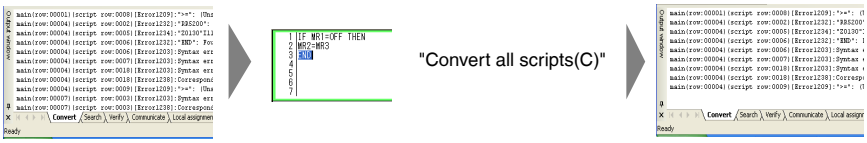
Select "Script(S)" → "Convert selected script(S)" from the menu.

Other procedure
- Ctrl key + T key
- From the right-click menu, select "Edit script(X)" →"Convert selected script(S)"

No error displayed in output window indicates that script correction is completed.

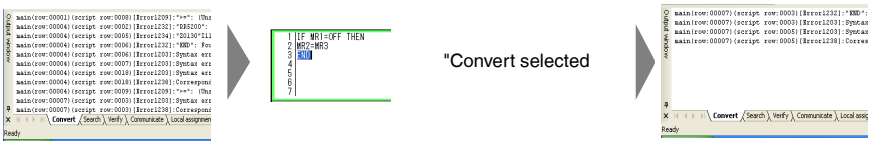
Reference
- In the case of many errors and warnings, other errors will be still displayed when selecting "Convert all scripts(C)" even if one error is corrected.

Many warnings　　Fix only 1　　Many warnings



In this case, select “Convert selected script(S)” from the menu.

Many warnings　　Fix only 1



Since only the corrected scripts are converted, the correction verification will be simpler.

- By using converted ladder show/hide (Ctrl+G), a converted ladder can be displayed and confirmed or a developed ladder can be closed even while editing the script.

Function block

Function blocks are ladder circuits used many times repeatedly in a project that are registered and standardized under a name of your choosing.
The registered function blocks can be called from a script program, and scripts can be used within the function blocks as well.

Calling Function Blocks

■ Calling Function Blocks

Function blocks can be called using FB / FBSTRT / FBCALL functions. To stop a function block, perform the following steps:

- Stop the calling source program (FB/FBSTRT/FBCALL)
- Execute the FEND function (instruction) in the function block (FB/FBSTRT)
- Turn the execution condition of the function block OFF (FBCALL)

"FB function", page 6-10
"FBCALL function", page 6-11
"FBSTRT function", page 6-13
"FEND function", page 6-15

● How to use

Programming Script FB(MR1000,"FunctionBlock",DM0,#0,MR2000)
Programming Script FBSTRT(MR1000, "FunctionBlock", Instance, DM0, #0, MR2000)
Programming Script FBCALL(MR1000, "FunctionBlock", Instance, DM0, #0, MR2000)

Point
After using the FBCALL/FBSTRT function, please upgrade the CPU function version of KV-8000 to the version after 2.0.

■ Calling functions

Functions can be called using the FUN instruction. The function stops when its execution condition is turned OFF.

■ How to use

Programming Script FUN(MR1000,"Function",DM0,#0,MR2000)

Reference
When using KV-8000 with CPU function version 2.0 or later, the return value can be processed after writing the FUN function in an expression.
The following is an example of storing the return value of Function1 in DM0.
Programm DM100 = FUN(MR1000, "Function1", DM0, DM1)

Programing Function Blocks Using Script

Function blocks can be programmed with a mixture of ladder programs and scripts, or can be created using scripts only.
For information on creating function blocks, see the "KV-8000 Series User's Manual", "KV-7000 Series User's Manual", "4-8 Function Block".

Point **END and ENDH instructions Point cannot be written in scripts.**

Macro

Ladder program section in project that is used for many times repeatedly, registered with any designated name, is called "macro".
Registered macro can be called from script program, and the macro can be also made using script.

Macro Call

■ Calling subroutine macro

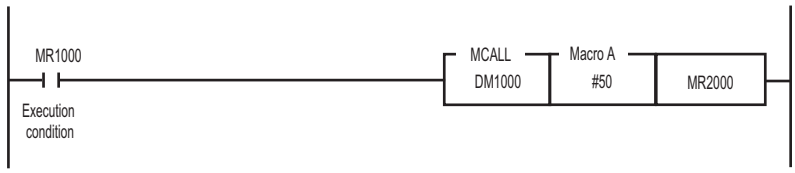
The macro that is registered as subroutine type can call MCALL function.

"MCALL function", page 6-6

● How to use

Programming Script MCALL(MR1000,"MacroA",DM1000,#50,MR2000)

Ladder Programming



■ Calling self-hold type macro

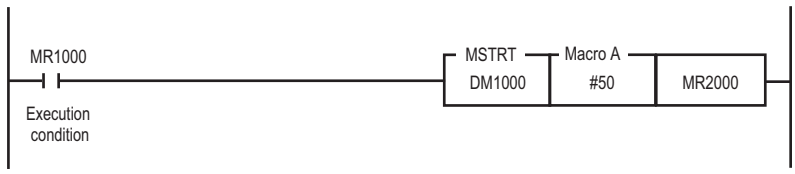
The macro that is registered as self-hold type can call MSTRT function. In this case, the called macro must have MEND instructions (functions).

"MSTRT function", page 6-7
"MEND function", page 6-8

● How to use

Programming Script MSTRT(MR1000, "MacroA", DM1000, #50, MR2000)

Ladder Programming



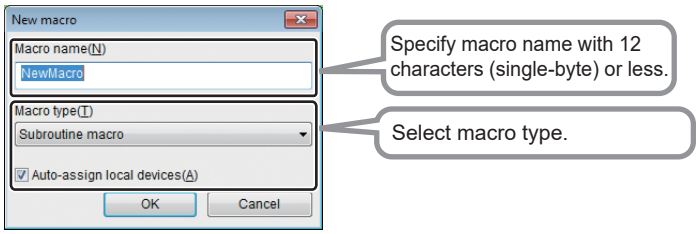
Programming Macro by Using Script

Newly make macro can be used with scripts by programming with ladder or scripts with script.

1 For the menu, select "Modules/macro (M)" → "New macro (M)".

Other procedure • Click the icon.
•From right-click menu, select "Edit macro (M)" → "New macro (N)"

2 Specify macro name and macro type.



Point

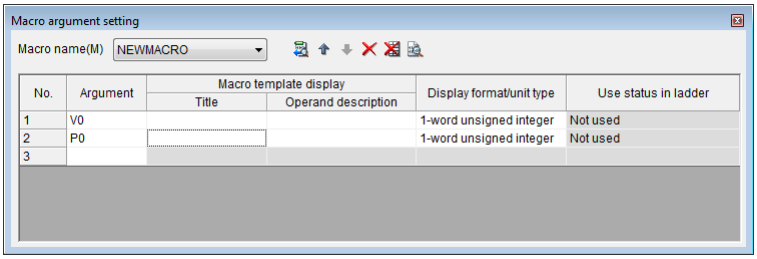
If macro name contains "+" / "-", "The bit under macro execution status" cannot be programmed in script (The script identifies "+ /-" to operators).
For details on "Bits in Macro Execution Status", refer to the user's manual of the CPU unit used.

Reference

- Some characters cannot be used in macro name.
- Macro type can also be changed after making a macro.
- When assigning local device manually, the checkbox shall be OFF.

3 Set macro argument.

Set the argument used by the macro to be registered from "Macro argument setting" dialog box.



4 Program the new macro.

To add the new macro in Workspace. Macro can be programmed by ladder, scripts or the combination of adder and scripst.

Reference

For detailed information on unit device contained in script program and the setting of macro argument device, please see "Macro" in the User's Manual of the CPU unit used.

Point

The script has no functions equivalent to END instruction and ENDH instruction and shall use ladder programming END and ENDH instructions.

Script Debug Function

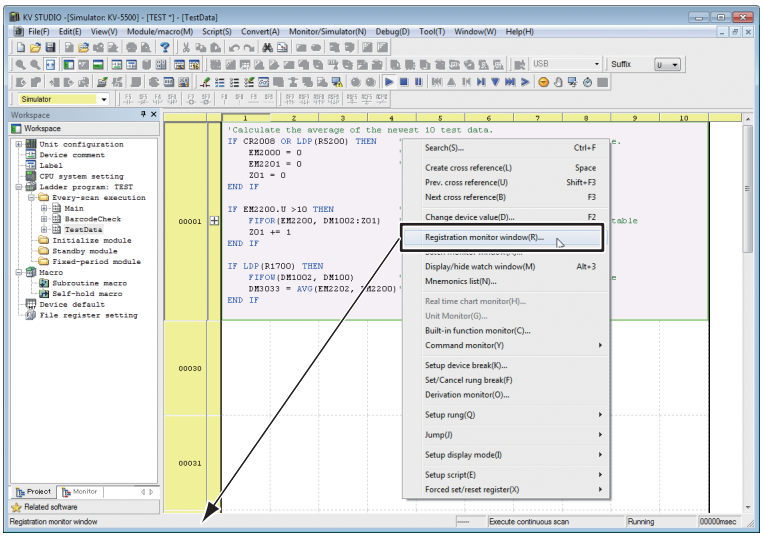
Monitor Function

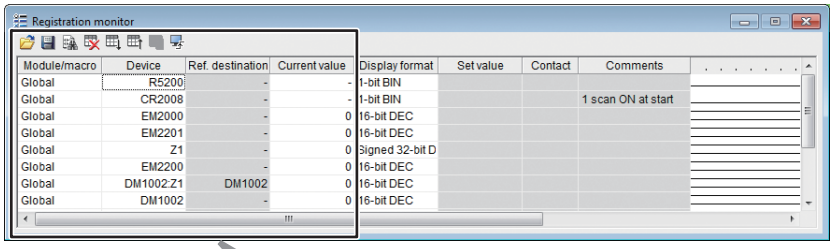
Monitor function allows to debug script program under online state or during starting simulator.

■ Register monitor

The current value of every device programmed in script can be monitored.

1 When KV STUDIO is in Monitor status or Simulator status, click the script to be monitored, select "Registration monitor window" from the right-click menu.





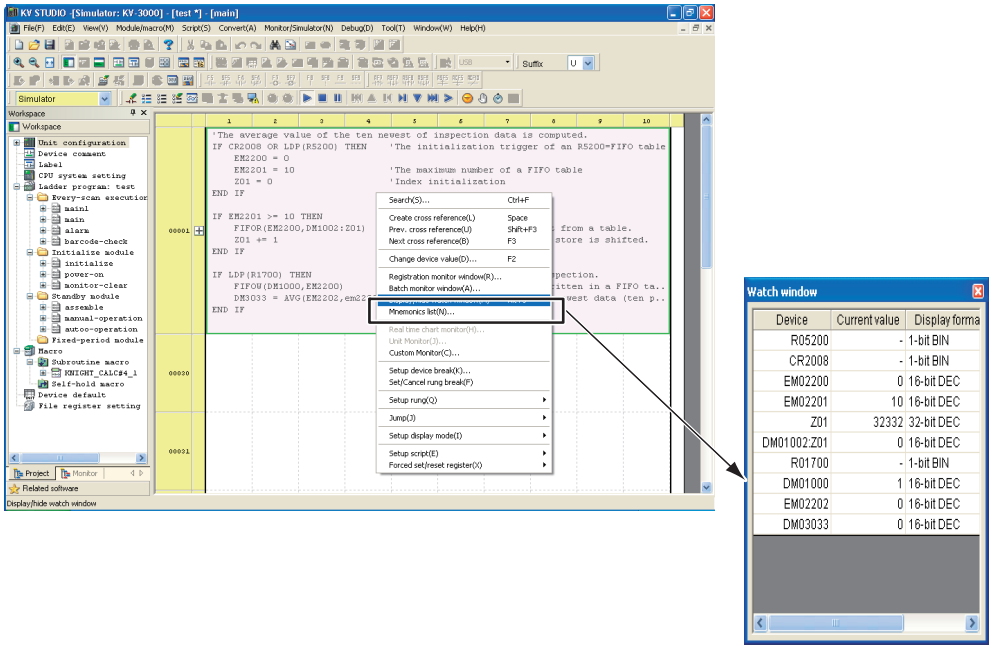
By registering all device programmed in script to “Registration monitor” window, current value and bit status can be monitored, as well as setting value, comments and time chart.

■ Watch window

Through the monitor window, you can monitor the current values of all devices or variables in the script by simply selecting the script.

1 When KV STUDIO is in Monitor or Simulator status, click the script to be monitored, select "Watch window " from the right-click menu.

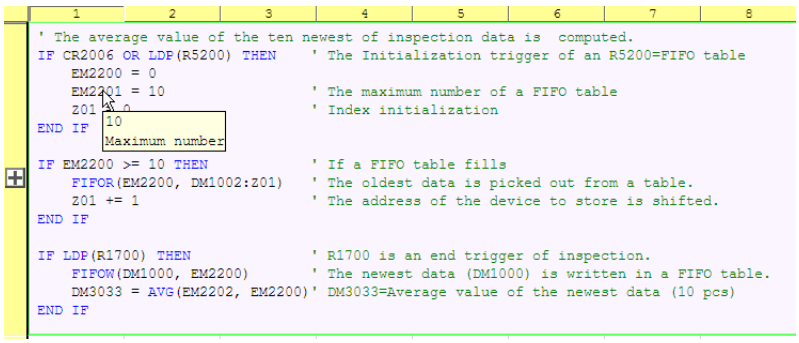
Other procedure



If programming other scripts, the device displayed in Monitor Window will be automatically switched.

■ Monitoring with tool tip

When KV STUDIO is in the Monitor or Simulator status, if the mouse cursor is moved to the device to be monitored, the current value and device comment will appear at the tool tip.



Input Auxiliary Function and Other Info

About Input Auxiliary Function

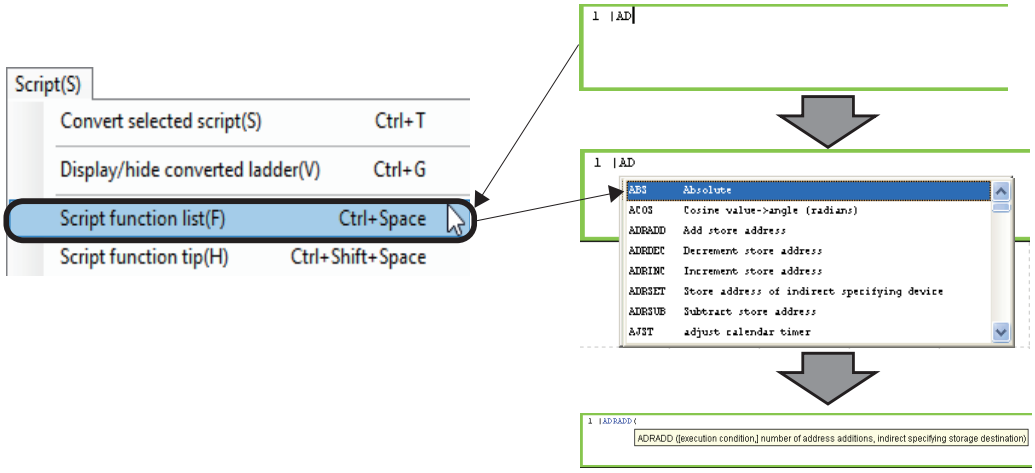
KV STUDIO provides various convenient functions and input auxiliary functions for Script Programming.

Script Function List

If function name is unknown, optional function can be called as follows after inputting function in midway.

● Call function list

From the menu, select "Script (S)" → "Script function list (F)".



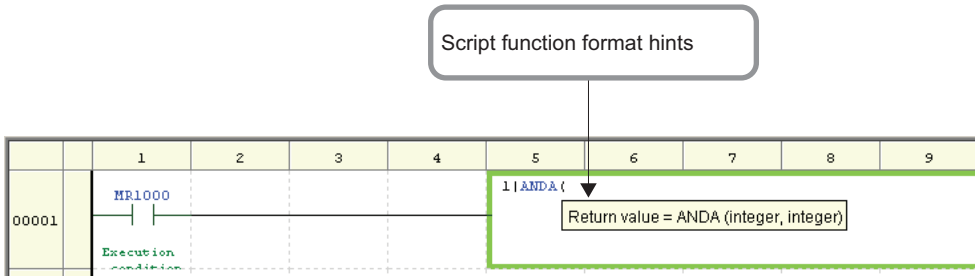
Select the function to be used and press ↵ key or double click, insert the selected function to the cursor position in Script Editor, and the script function format prompt is displayed.

Other procedure
- Ctrl key + Space key
- From the right-click menu, select "Script function list (F)"

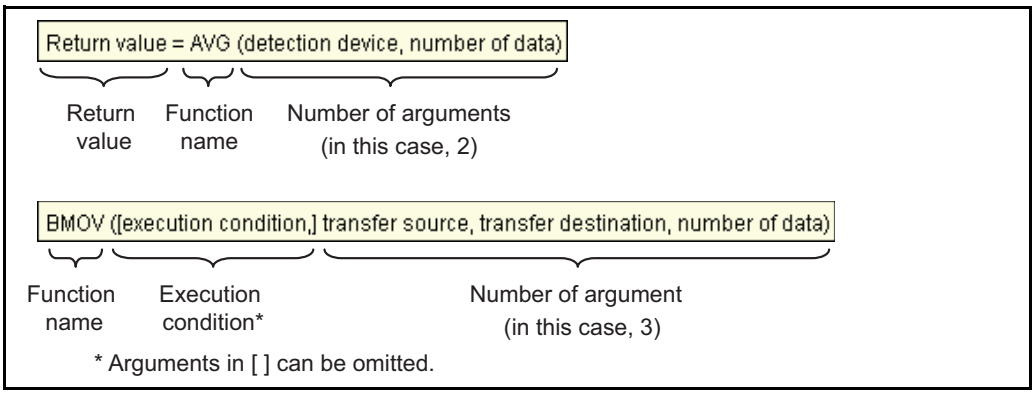
Reference
- Optional script function can be also called when any content not input. In this case, all optional functions are displayed in alphabetic sequence.
- In this case, all optional functions are displayed in alphabetic sequence.

Script function format hints

When inputting function, "Script function format hints" can simply inform messages such as argument and return value, etc.

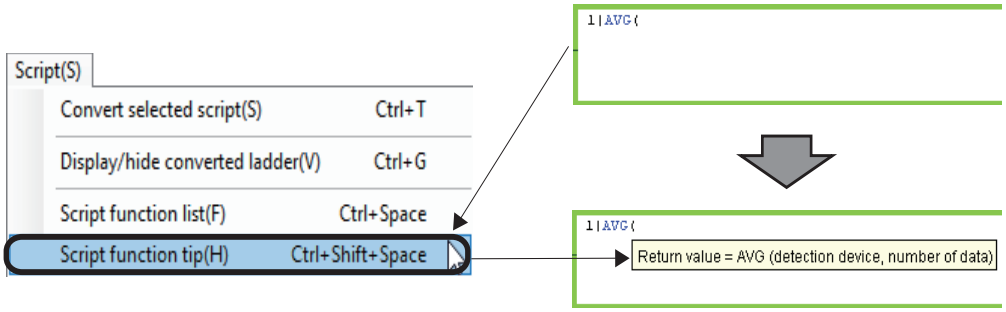


After inputting "Function name", the "Script function format hints" will be automatically displayed.



● Call script function format hints

Place mouse cursor "I" to the right side of "(", and select "Script (S)" →"Script function hints (H)".



Other Ctrl key+ Shift key+ Space key

Reference "Script function format hints (H)" can be also called during script programming.

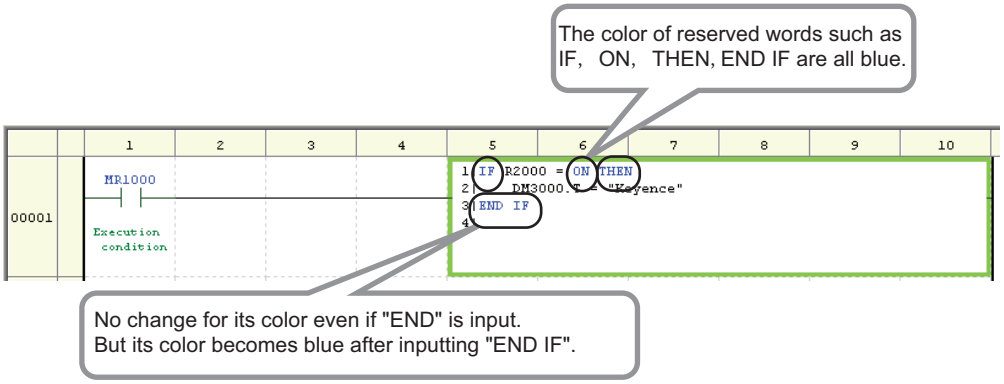
Point **When calling script function format hints, the cursor must be placed to the right side of "(" or start position of function. When the cursor is at other positions, the script function format hints cannot be called.**

Reserved word

The word (reserved word) such as control statement and function, etc. registered to KV script will be blue during programming. If character color is not changed, it indicates an input error.

Reference

The character color of reserved word can be changed via the "option setting" dialog box of KV STUDIO.



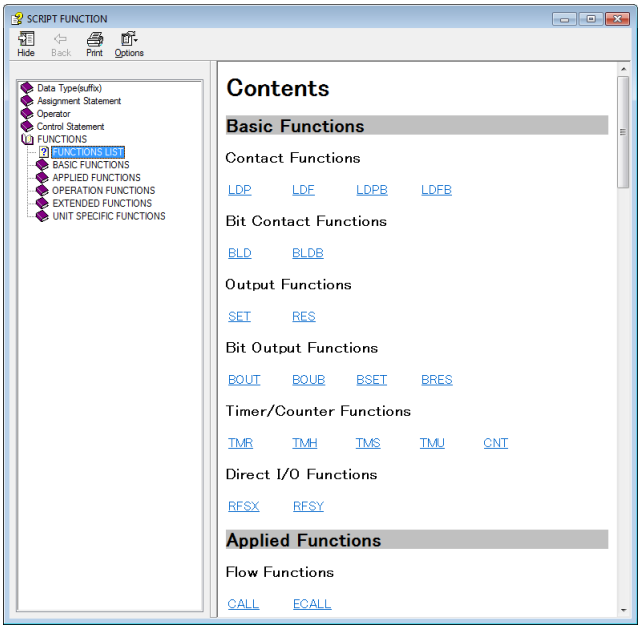
HELP Function

For details of function format and how to use, please see "Script function help".

■ Start script function help

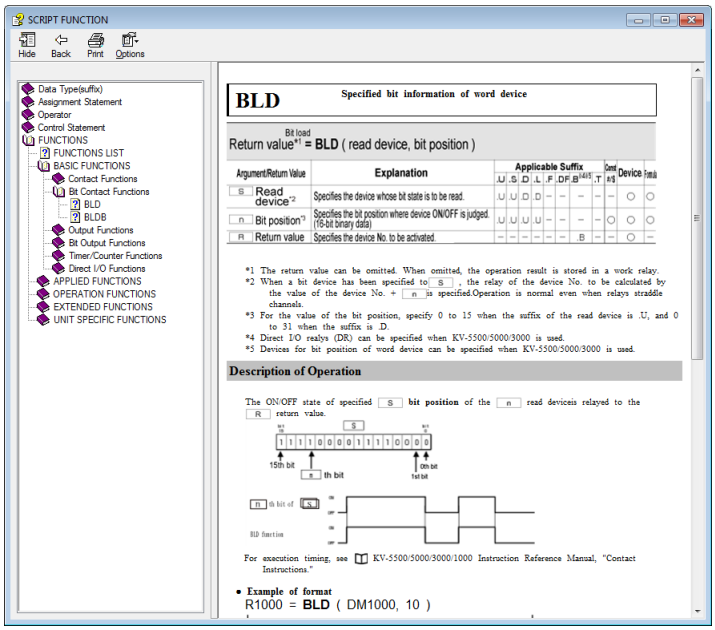
1 From the menu, select "Help (H)" → "Script Function Help (L)".

"Script function help" directory appears.



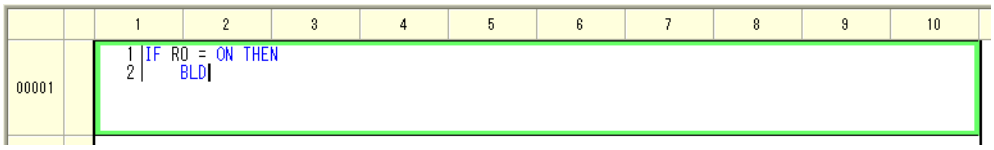
2 **Select the function that you want to inquire.**

To display detailed function descriptions.

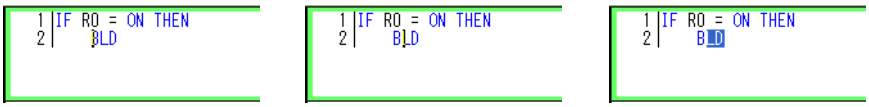


■ Display help of the specified function on Script Edit Screen

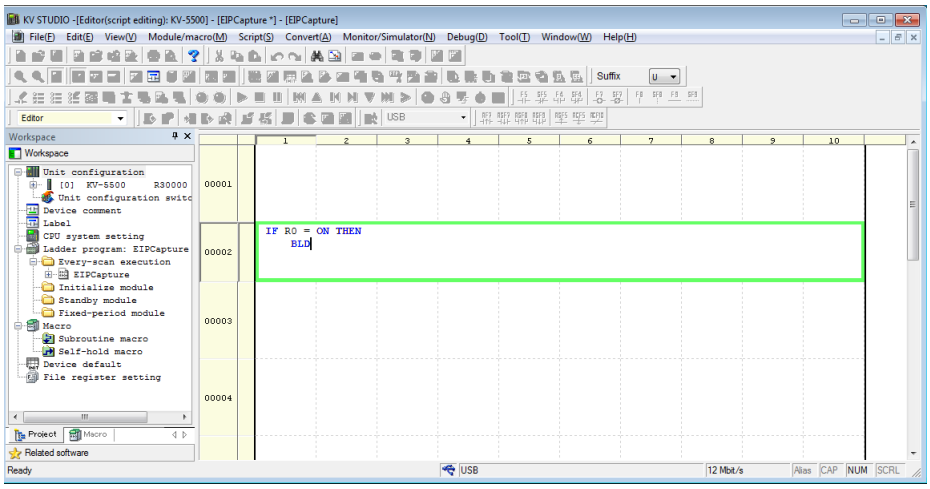
1 **Move the cursor to the function that you want to inquire.**



The cursor position as follows, shall be before and after or in middle of function, or a part of function is within the selected range.

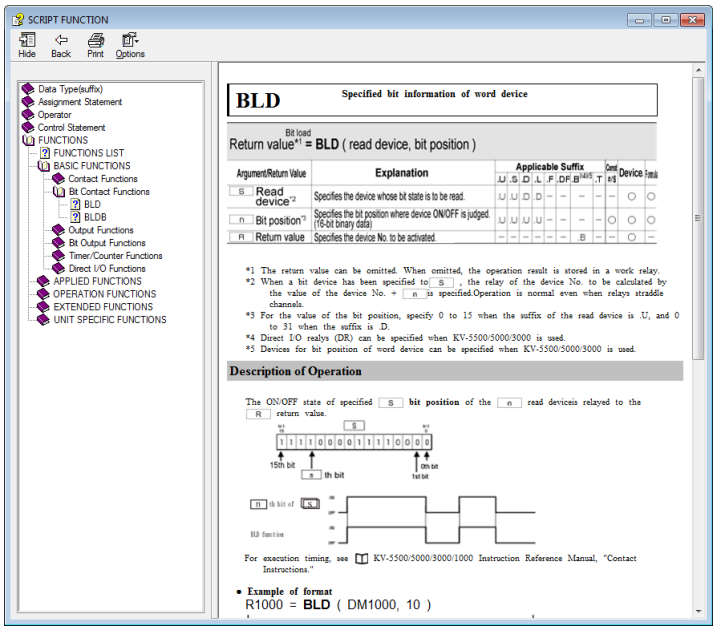


2 **Click button.**



Other procedure To select function, press F1 key.

Display the help screen of the selected function.

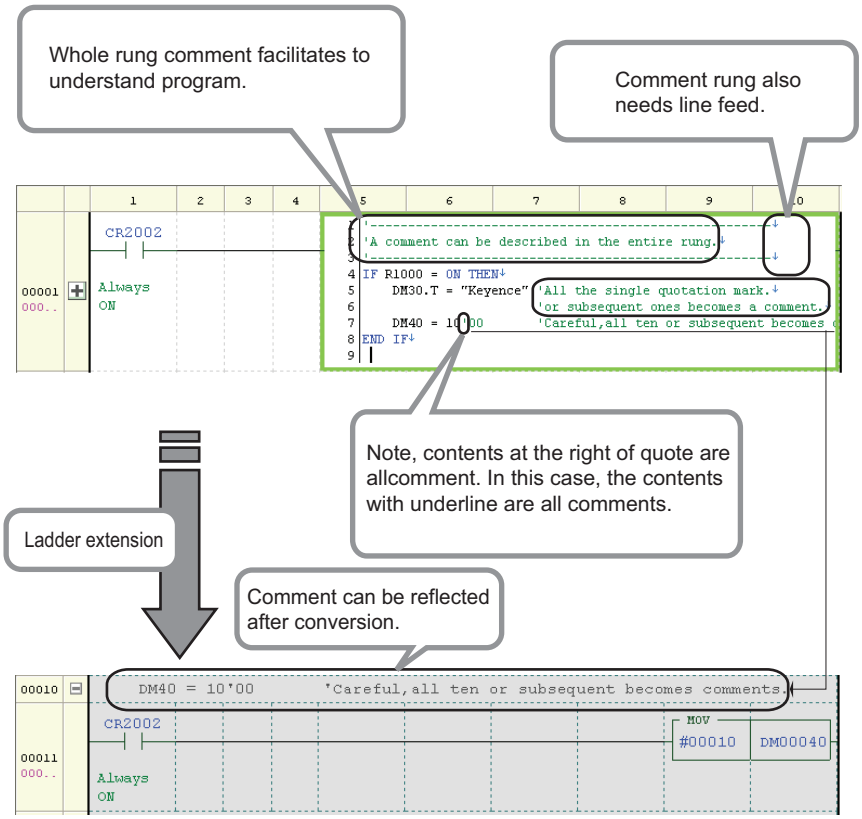


About Comments Function

KV script can program the content from single quote(') (or "REM ") to line feed (carriage return entry) at any position of script program as comment. Comment programming can improve efficiency of debug and program modification. It is recommended that comment shall be programed together with script.

● Comment writing method

Whole rung comment can be made on right side of statement (operation formula).



Reference

- Even increase of comment line will not influence program size and scanning time.
- The number of characters entered on a single rung is up to 1024 characters (halfwidth) including scripts and comments.
- The following methods can be used to change multiple selected rungs into a single comment rung (KV STUDIO version 11.30 or later).
  - "Set line disable (Q)" on the right-click menu
  - "ST/Script (S)" → "Set line disable (Q)" on the menu
  - Ctrl + Q
- The following methods can be used to clear the selected multiple-rung comment (KV STUDIO version 11.30 or later).
  - "Cancel line disable (W)" on the right-click menu
  - "ST/Script (S)" → "Cancel line disable (W)" on the menu
  - Ctrl + W

Write Easy-reading Script

When programming scripts, understandable programming method enables improving debug efficiency. Similar to ladder program, script programs finished shall be easy-to-understand in modification, maintenance and reuse. So, the written script shall be understandable and clear.
So, it can be effectively used without programmer.

● Sample Program

Abstruse program

```
IF CR2008 OR LDP(R5200) THEN
EM2200=0
EM2201=10
Z01=0
END IF
IF EM2200>=10 THEN
FIFOR(EM2200,DM1002:Z01)
Z01+=1
END IF
IF LDP(R1700) THEN
FIFOW(DM1000,EM2200)
DM3033=AVG(EM2202,EM2200)
END IF
```

Intelligible program

Calculate 10 average values of data.

```
IF CR2008 OR LDP(R5200) THEN        'R5200 = FIFO table initialization trigger
  EM2200 = 0                        'number of data stored in current FIFO table
  EM2201 = 10                       'maximal number of FIFO table
  Z01 = 0                           'index initialization
END IF

IF EM2200 >= 10 THEN                'when FIFO table is full
  FIFOR(EM2200, DM1002:Z01)         'delete the earliest data from table
  Z01 += 1                          'address of storing device
END IF

IF LDP(R1700) THEN                  'R1700 check end trigger
  FIFOW(DM1000, EM2200)             'the latest data (DM1000 ) is written into FIFO table
  DM3033 = AVG(EM2202, EM2200)      'DM3033 =average value of the latest data (10)
END IF
```

Tips for programming

- Use Indent.
- Add Space before and after operators.
- Add line feed (blank line).
- Add line comment and sentence end comment.

Reference

- Automatic indent function is available.
  Select "Tool (T)" → "Option (O)" from the menu, set automatic indentation of "Script" tab to ON.
- If the line of script is long, a new line can be started by adding an underbar (_) to the end.

Inline Mnemonics

Mnemonics is a language to enable operation of the created program on CPU unit.
The program made with KV script and expansion ladder, will be converted to mnemonics when transferring to CPU unit.
For detail of operation diagram, see User's Manual of the CPU unit used.

With inline mnemonics, mnemonics can be programmed directly in script program, i.e. mnemonics language can be mixed with script language.
Plus "!! " attached at the start of mnemonics is used to discriminate from script.

Point

The following inline mnemonics cannot be used:

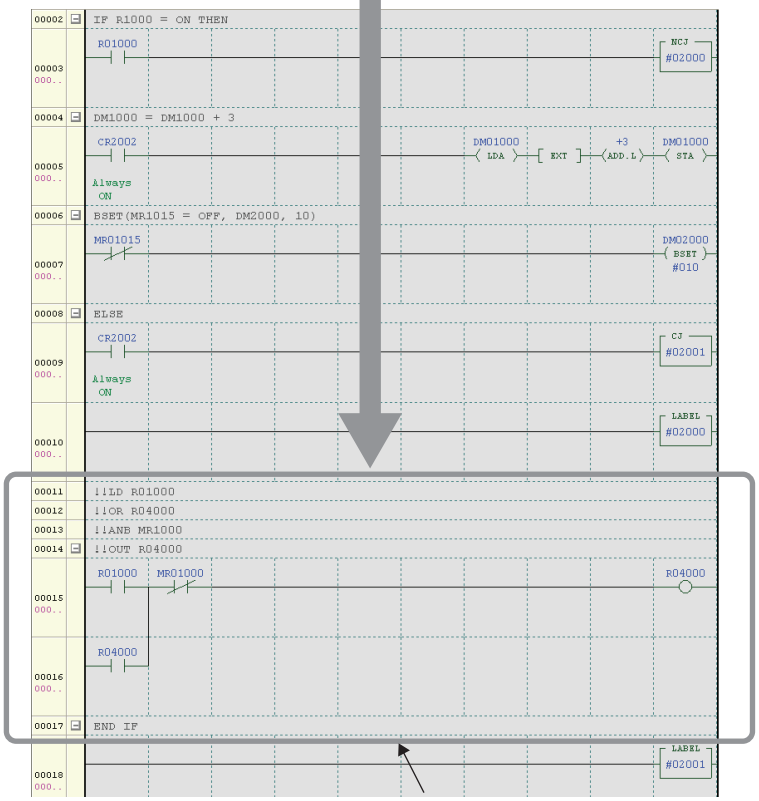
- **!!END**
- **!!ENDH**

Programmin

```
IF R1000 = ON THEN
    DM1000 = DM1000 + 3
    BSET(MR1015 = OFF, DM2000, 10)
ELSE
    !!LD     R01000
    !!OR     R04000
    !!ANB    MR1000
    !!OUT    R04000
END IF
```

Inline mnemonics

Ladder



Inline mnemonics ladder

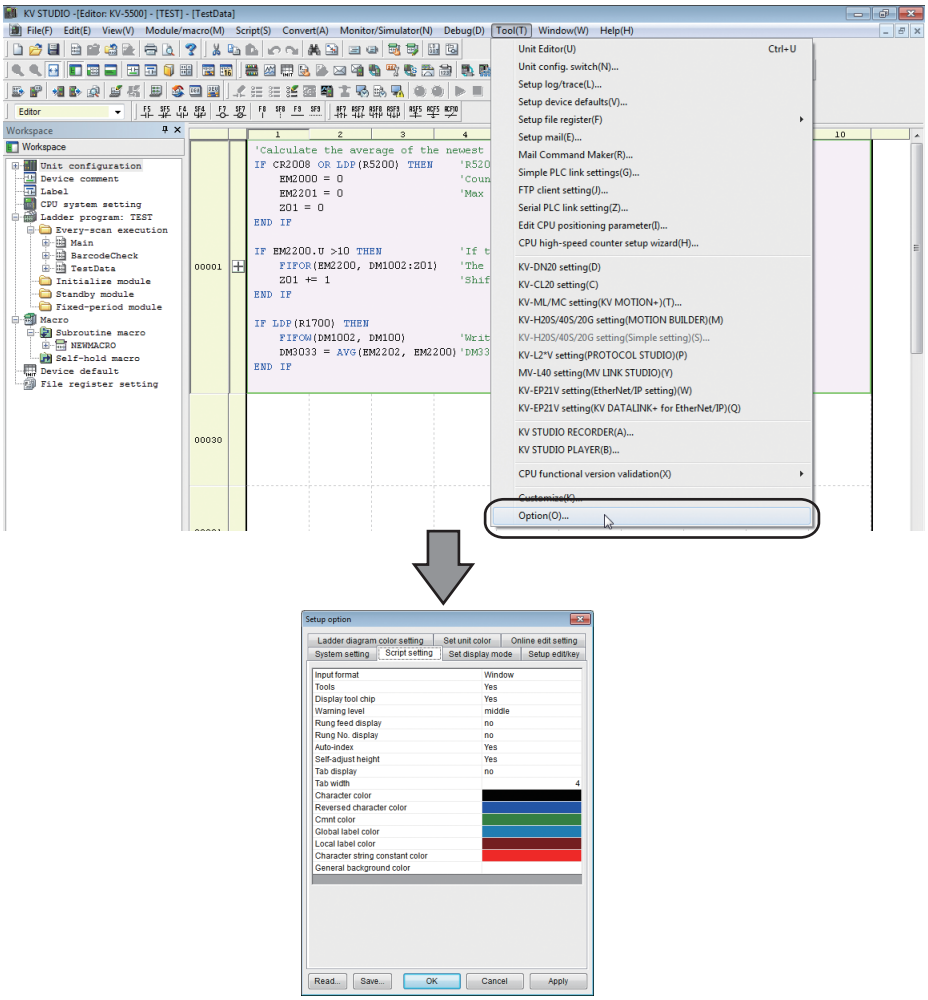
Precautions on Edit Screen and Customization

■ Precautions when programming script

- Script program cannot use 2-byte (Chinese characters) code (but, comment part and text string constant, global label/local label can use Chinese characters).
- The number of characters input on one rung is within 1024 characters (single-byte), including scripts and comments.
- Ladder of KV script extended cannot be directly edited.

■ Customizing script window

From the menu, select "Tool (T)" → "Option (O)" → "Script setting" to customize edit window.
For details, see 📖 KV STUDIO User's Manual.



About Script Work Memory (Work Devices)

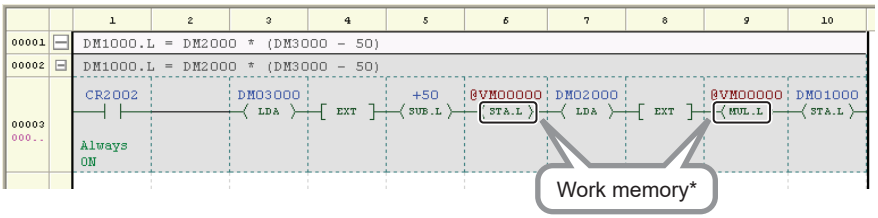
When converting the script program to Ladder, work memories can be used for CPU unit KV-8000/7500/7300/5500/5000/3000 or KV Nano Series, work memory and work devices for CPU unit KV-1000.

Work memory (work device)

Work memory (work device) refers to the device for temporarily storing partial operation result such as complicated operation formula.
In script program, during script conversion, KV STUDIO can allocate work devices from the pre-determined area automatically, so, reuse of devices doesn't occur during use.

Programming Script DM1000.L = DM2000 * (DM3000 - 50)

Ladder diagram



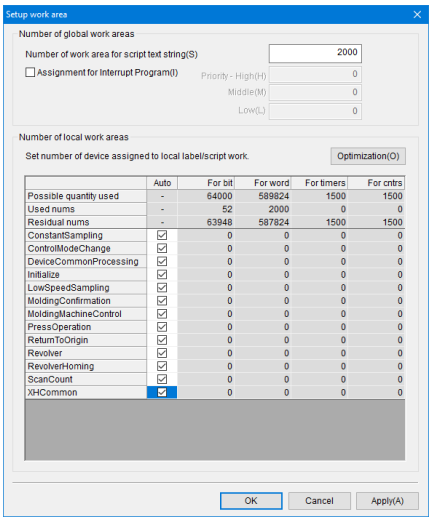
*When CPU unit is KV-1000, work devices such as EM64536 can be used.

Description The result obtained by carrying out [DM3000-50] operation is stored in work memory (@VM0) temporally, and then multiplied with DM2000, the final result is stored in left DM1000.L.

■ Application area of work memories

When CPU is KV-8000/7500/7300/5500/5000/3000 or KV Nano Series, the script uses work memories.
Work memories used for scripts can be checked via "Setup work area" dialog box.

"KV STUDIO User's Manual"



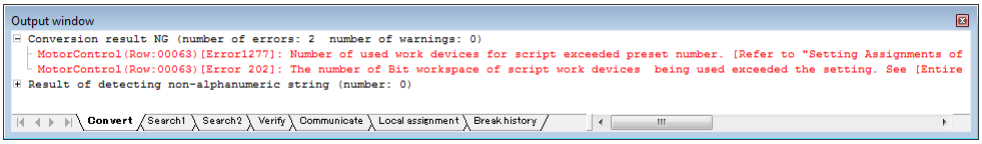
■ Application area of work devices

When CPU unit is KV-1000, the script uses work devices.

Work devices take the pre-determined area as "bit workspace", "word workspace" and "text string workspace". According to program content, quantity of work devices will be inadequate sometimes.

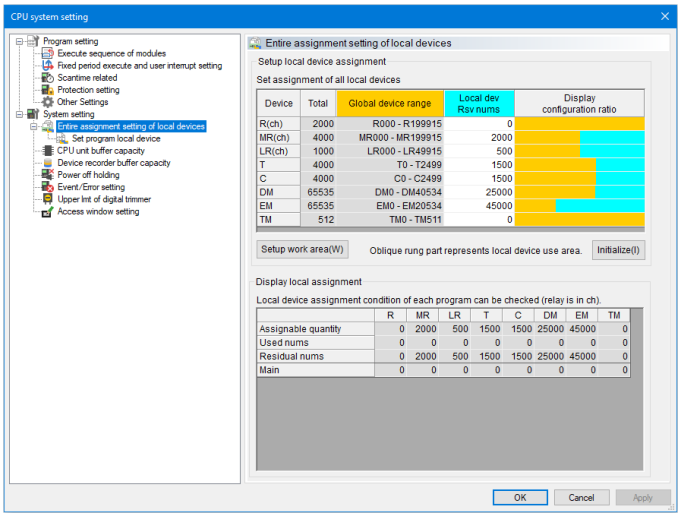
● When work devices are insufficient

When work devices are insufficient, the following message will be displayed in output window during conversion of ladder program.



In the status, the program cannot be transferred to PLC, so the area of work devices shall be enlarged as follows.

1 From the menu, select "Convert (A)" → "Entire assignment setting of local devices (A)", and "CPU system setting" dialog box will appear.



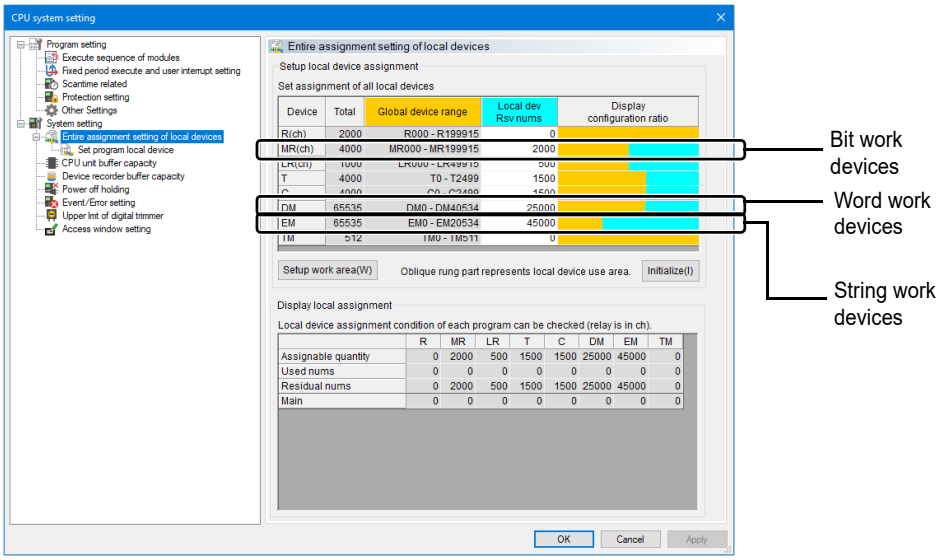
Other procedure
- Double click "CPU system setting" in workspace.
- Double click error message in popping up window
- Click button.

Reference

The setting for the reserved work devices for the script is only displayed in projects where"KV-1000" is selected for the model (CPU unit).

2 Edit the column for the reserved work devices for the script.

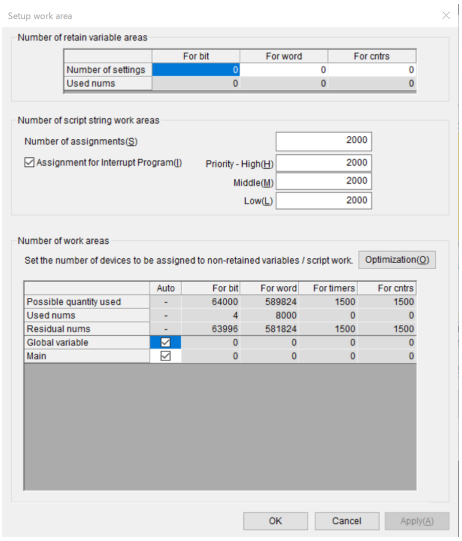
According to error message of output window, change assignment of every workspace.



Reference Three types of work devices in workspace: "bit work device" for temporarily storing bit status, "word working device" for temporarily storing operation result, and "text string work devices" for temporarily storing text string.

Point **Work devices for text string can be reused for many times in a project, so, care shall be taken when programming script using text string via interrupt program.**

When using KV-8000 and KV-7500/7300 of the function version 2.2 or later, click [Working Area Setting(W)] to assign the working area for interrupting the program.



XYM Marking

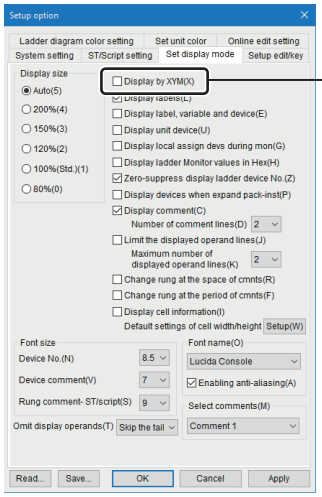
Script program can be made via XYM marking.

■ Script program via XYM mark.

1 From the menu, select "Tool (T)" → "Option (O)".

"Setup option" dialog box appears.

2 Click the "Set display mode" label.



Select “Display by XYM(X)” checkbox and click “OK”.

3 Program with XYM marking.

```
1 IF D1000 > K20 THEN
2     D0 = K50
3     SET(X10)
4 END IF
5
```

Reference

After selecting "Display by XYM (X)", programming can be enabled with normal Device No. (mixable).

If the XYM marking is canceled after editing the script with XYM marking, an error will occur when the script using X or Y is converted again. For M and L, the device number changes from a decimal to a CH marking such as R, so note that the device number in the converted results will change.

Example) M100=ON

The results of converting this script to a M100 ladder are:

- When using XYM marking, the script is converted to MR604.
- When not using XYM marking, the script is converted to MR100.

MEMO

3

DATA PROCESSING

Devices 3-2
Index Modification and Indirect Specifying 3-54
Data Type (suffix) 3-62

Devices

Device refers to element (variable) used in internal relay and data memory, etc. of CPU unit, enabling to supply data used for operation and functions and store function execution results.
The device varies with the models (CPU units).

Device List (When using KV-8000)

■ KV-8000 bit device

Device name	Range*1	Number of points	Main functions	When start operation (Power ON, PRG →RUN)	Local device (@)*1	Index Modification (:Z/:#)	Indirect Specifying (*)*8	
							Reference	Store
Input Relay*10	R000 ～ R199915*2	Total 32000	Device used for capturing ON/OFF info from peripherals.	Clear/hold*3*12	○	○	○	×
Output relay*10			Device used for sending ON/OFF info to peripherals					
Internal auxiliary relay*10			Used in CPU and expansion unit					
Link relay	B00 ~ B7FFF*10	32768 points	Devices that can be only used in CPU	Clear/hold*3	×	○	○	×
Internal auxiliary relay	MR000 to MR399915	64000 points		Clear	○	○	○	×
Latch relay	LR000 to LR99915	16000 points		Hold	○	○	○	×
Timer (contact)	T0 to T3999	4000 points	10ms, 1ms, 10ms, 100ms down counter, 10ms contact device of up/down counter	Clear	○	○	○	×
Counter (contact)	C0 to C3999	4000 points	Contact device for up counter, up/ down counter	Clear/hold*7	○	○	○	×
Control relay	CR000 to CR7915	1280 points	Devices used for controlling PLC function or capturing the status from peripherals.	Hold	×	×	×	×

■ KV-8000 series word device

Device name	Range*1	Number of points	Main functions	When start operation (power supply PRG →RUN)	Local device (@)*1	Index Modification (:Z/:#)	Indirect Specifying (*)*8	
							Reference	Store
Data memory	DM0 to DM65534	65535 points	Device for storing numerical data (16-bit)	Clear/hold*3	○	○	○	○
Expansion data memory	EM0 to EM65534	65535 points		Clear/hold*3	○	○	○	○
File register	Dial mode ZF0 to ZF524287	524288 points		Clear/hold*3	×	○	○	○
	Current group FM0 to FM32767	(32768×16 bank)			○*11			
Link register	W0 ~ W7FFF*10	32768 points		Clear/hold*3	×	○	○	○
Temporary memory	TM0 to TM511	512 points	Device for storing numerical data (16-bit)	Clear	○	○	○	○
Timer (current/ setting value)	T0 to T3999	4000 points	10ms, 1ms, 10ms, 100ms down counter, 10ms current/ setting value of up/down counter (32-bit)	current value: Clear*5 current value: Hold	○	○	○	×
Counter (current/ setting value)	C0 to C3999	4000 points	Current/setting value of up counter and up/ down counter (32-bit)	Clear/hold*7	○	○	○	×
Index register	Z1 ~ Z23*6	23 points	Devices that can be used in index modify (32-bit)	Clear	×	×	×	×
Control memory	CM0 to CM7599	7600 points	Devices used for controlling PLC function or capturing the status from peripherals (16-bit).	Hold	×	×	×	×

*1 Since they are used for local device, their reserved number is limited by the usable range of global devices. "Local device list", page 3-5

*2 R000 to R199915 can be specified as extended I/O relay with Unit Monitor.

*3 Clear/Hold settings can be made in the "Power Failure Holding" option.

*4 "KV-8000 Series User's Manual "CR/CM table""

*5 Up/down timers (UDT) hold the current value.

*6 Z11 and Z23 are reserved for the system, so you cannot use these devices.

*7 Hold/Clear setups can be made in the "Power Failure Holding" option. (only for contact, current value)

8 "Store" is a device that can be described as "(device)". "Lookup" is the device that can perform indirect specifying.
"KV-8000 Series User's Manual" "Indirect specifying"

*9 Specify all device No. in HEX.

*10 "DR" represents direct processing.
"KV-8000 Series User's Manual "About direct processing"

*11 The local device cannot be used for bank switching.

*12 For the input relay, clear/hold cannot be set.

■ Macro operand device

The macro operand devices used only in macros are shown as follows.

Attribute		Symbols	Range	Description	Indirect Specifying (*)	Local device (@)	Index modification (:)
Device		P	P0 to P9	Devices are used as argument.	×	×	○
Value		V	V0 to V9	Device values or constants are used as argument for processing.	×	×	×
Unit	Leading R	UR	UR0 to UR9	Leading device (R) assigned to the expansion unit is processed as an argument.	×	×	○
	Leading DM	UM	UM0 to UM9	Leading data memory (DM) assigned to the expansion unit is processed as an argument.	×	×	○
	No.	UV	UV0 to UV9	Unit number of the expansion unit is processed as an argument.	×	×	×

■ Function block argument

The arguments used in the function block only are as shown below.

Argument type		Argument name / device		Description	Indirect Specifying (*)	Local device (@)	Index modification (:)
IN		Any label		Device values or constants are used as argument for processing.	×	×	×
OUT		Any label		Written to the device that specified the operation results.	×	×	×
IN-OUT		Any label		Devices are used as argument.	×	×	○
Unit	Leading R	UR	UR0 to UR7	Leading relay (R) assigned to the unit is processed as an argument.	×	×	○
	Leading DM	UM	UM0 to UM7	Leading data memory (DM) assigned to the unit is processed as an argument.	×	×	○
	No.	UV	UV0 to UV7	Unit number is processed as an argument.	×	×	×

■ Const

Const	Range	Main functions
DEC(#)	#0 to #65535 (unsigned 16-bit) -32768 to +32767 (signed 16-bit) #0 to #4294967295 (unsigned 32-bit) -2147483648 to +2147483647 (signed 32-bit)	Used for specifying DEC constant. *XYM Marking is represented by "K".
Single precision floating point type real number	-3.4E+38 ≦ N ≦ -1.4E-45 N=0 +1.4E-45 ≦ N ≦ +3.4E+38 (Significant digit : 7 digit)	Used for specifying single precision floating point type real number.
Double precision floating point type real number	-1.79E+308 ≦ N ≦ -2.23E-308 N=0 +2.23E-308 ≦ N ≦ +1.79E+308 (Significant digit : 16 digit)	Used for specifying double precision
floating point type real number.	HEX($) $0000 to $FFFF (16-bit) $00000000 to $FFFFFFFF (32-bit)	Used for specifying HEX constant. * XYM Marking is represented by "H".
Fixed text string(" ")	Example "Type ABC", "MON/08:30:15"	Used to fixed text string

■ Internal register

Device name	Range	Number of points	Main functions	When start operation (Power ON, PROG -> RUN)
Internal register	-	1 points	Device for temporary data storage (16-bit/32-bit/64-bit)	Clear

Local device list

The number of local devices/global devices and their settings when a project is created are shown as follows. Number of reserved devices can be changed through “CPU system setting” → “Assignment of all local devices setting” on the “Workspace”.

Device name*1	Local device	Global device	Total
	Reserved devices in the whole number project*2	Counts/range	
Relay（R）*3	None	32000 points (2000 CH) R000 to R199915	32000 points (2000 CH)
Internal auxiliary relay (MR)*3	32000 points (2000 CH)	32000 points (2000 CH) MR000 to MR199915	64000 points (4000 CH)
Latch relay (LR)*3	8000 points (500 CH)	8000 points (500 CH) LR000 to LR49915	16000 points (1000 CH)
Timer (T)*4	1500 points	2500 points T0 to T2499	4000 points
Counter (C)*4	1500 points	2500 points C0 to C2499	4000 points
Data memory (DM)	25000 points	40535 points DM0 to DM40534	65535 points
Expansion data memory (EM)	45000 points	20535 points EM0 to EM20534	65535 points
Temporary data memory (TM)	None	512 points TM000 to TM511	512 points

*1 Link relays (B), link registers (W), file registers (ZF), control relays (CR), control memories (CM), and index registers (Z) cannot be used as a local device.
When using @FMxxx, local devices in workspace will be occupied.

*2 When a project is created, the device area used for global devices can be guaranteed as a whole (suppose 10 modules are used and 50 counts of macro calling.)
The number of reserved devices can be changed as required.

*3 Number can be set in “ch” unit.

*4 "@+ (device code)" can be used only when the device code is specified as an operand for timer/counter instructions. (Example for inputting mnemonic : TMR @0 #100).

Point

- **If relays (R) and temporary data memories (TM) are used as a local device, the number of relays and data memories must be specified.**
- **The following devices must be used as a global device:**
  - **Devices assigned to expansion units**
  - **The devices referenced by such peripherals as touch panel**
  - **Common devices in the whole project**
  - **Devices for inter-module data transmission**
- **Where possible, local devices should be coded from 0 and use consecutive numbers.**
- **Since R and DM are assigned to the expansion unit, as a global device, there must be an area to be used. When using local devices, please use MR or EM first.**
- **If local devices or global devices are used outside the specified range, then a conversion error will occur.**
- **For the KV-8000 Series, when an index device or indirect specification is used to specify outside of the usable range of local devices or global devices, an operation error will occur.**
- **When various instructions using the leading device as the operands and index modification or indirect specifying are used, the range of the local devices and global devices that can be used should not be exceeded.**

Device list (When using KV-7000)

■ Bit Device

Device name	Range*1	Number of points	Main functions	When start operation (Power ON, PRG →RUN)	Local device (@)*1	Index Modification (:Z/:#)	Indirect Specifying (*)*8	
							Reference	Store
Input Relay*10	R000 ～ R199915*2*13	Total 32000	Device used for capturing ON/OFF info from peripherals.	Clear/hold*3*12	○	○	○	×
Output relay*10			Device used for sending ON/OFF info to peripherals					
Internal auxiliary relay*10			Used in CPU and expansion unit					
Link relay	B00 ~ B7FFF*10	32768 points	Devices that can be only used in CPU	Clear/hold*3	×	○	○	×
Internal auxiliary relay	MR000 to MR399915	64000 points		Clear	○	○	○	×
Latch relay	LR000 to LR99915	16000 points		Hold	○	○	○	×
Timer (contact)	T0 to T3999	4000 points	10µs, 1ms, 10ms, 100ms down counter, 10ms contact device of up/down counter	Clear	○	○	○	×
Counter (contact)	C0 to C3999	4000 points	Contact device for up counter, up/ down counter	Clear/hold*7	○	○	○	×
Control relay	CR000 to CR7915	1280 points	Devices used for controlling PLC function or capturing the status from peripherals.	Hold	×	×	×	×

■ KV-7000 series word device

Device name	Range*1	Number of points	Main functions	When start operation (power supply PRG →RUN)	Local device (@)*1	Index Modification (:Z/:#)	Indirect Specifying (*)*8	
							Reference	Store
Data memory	DM0 to DM65534	65535 points	Device for storing numerical data (16-bit)	Clear/hold*3	○	○	○	○
Expansion data memory	EM0 to EM65534	65535 points		Clear/hold*3	○	○	○	○
File register	Dial mode ZF0 to ZF524287	524288 points		Clear/hold*3	×	○	○	○
	Current group FM0 to FM32767	(32768×16 bank)			○*11			
Link register	W0 ~ W7FFF*10	32768 points		Clear/hold*3	×	○	○	○
Temporary memory	TM0 to TM511	512 points	Device for storing numerical data (16-bit)	Clear	○	○	○	○
Timer (current/ setting value)	T0 to T3999	4000 points	10ms, 1ms, 10ms, 100ms down counter, 10ms current/setting value of up/down counter (32-bit)	current value: Clear*5 current value: Hold	○	○	○	×
Counter (current/ setting value)	C0 to C3999	4000 points	Current/setting value of up counter and up/ down counter (32-bit)	Clear/hold*7	○	○	○	×
Index register	Z1 ~ Z12*6	12 points	Devices that can be used in index modify (32-bit)	Clear	×	×	×	×
Control memory	CM0 to CM5999	6000 points	Devices used for controlling PLC function or capturing the status from peripherals (16-bit).	Hold	×	×	×	×

*1 Since they are used for local device, their reserved number is limited by the usable range of global devices. "Local device list", page 3-9

*2 R000 to R199915 can be specified as extended I/O relay with Unit Monitor.

*3 Clear/Hold settings can be made in the "Power Failure Holding" option.

*4 "KV-8000 Series User's Manual "CR/CM table""

*5 Up/down timers (UDT) hold the current value.

*6 Z11 and Z12 are reserved for the system, so you cannot use these devices.

*7 Hold/Clear setups can be made in the "Power Failure Holding" option. (only for contact, current value)

8 "Store" is a device that can be described as "(device)". "Lookup" is the device that can perform indirect specifying.
"KV-7000 Series User's Manual "Indirect specifying"

*9 Specify all device No. in HEX.

*10 "DR" represents direct processing.
"KV-7000 Series User's Manual "About direct processing"

*11 The local device cannot be used for bank switching.

*12 For the input relay, clear/hold cannot be set.

*13 For CPU function versions older than 2.2, the number of points for R is 16000 and the range is R00000 to R99915.

■ Macro operand device

The macro operand devices used only in macros are shown as follows.

Attribute		Symbols	Range	Description	Indirect Specifying (*)	Local device (@)	Index modification (:)
Device		P	P0 to P9	Devices are used as argument.	×	×	○
Value		V	V0 to V9	Device values or constants are used as argument for processing.	×	×	×
Unit	Leading R	UR	UR0 to UR9	Leading device (R) assigned to the expansion unit is processed as an argument.	×	×	○
	Leading DM	UM	UM0 to UM9	Leading data memory (DM) assigned to the expansion unit is processed as an argument.	×	×	○
	No.	UV	UV0 to UV9	Unit number of the expansion unit is processed as an argument .	×	×	×

■ Function block argument

The arguments used in the function block only are as shown below.

Argument type		Argument name / device		Description	Indirect Specifying (*)	Local device (@)	Index modification (:)
IN		Any label		Device values or constants are used as argument for processing.	×	×	×
OUT		Any label		Written to the device that specified the operation results.	×	×	×
IN-OUT		Any label		Devices are used as argument.	×	×	○
Unit	Leading R	UR	UR0 to UR7	Leading relay (R) assigned to the unit is processed as an argument.	×	×	○
	Leading DM	UM	UM0 to UM7	Leading data memory (DM) assigned to the unit is processed as an argument.	×	×	○
	No.	UV	UV0 to UV7	Unit number is processed as an argument.	×	×	×

■ Const

Const	Range	Main functions
DEC(#)	#0 to #65535 (unsigned 16-bit) -32768 to +32767 (signed 16-bit) #0 to #4294967295 (unsigned 32-bit) -2147483648 to +2147483647 (signed 32-bit)	Used for specifying DEC constant. *XYM Marking is represented by "K".
Single precision floating point type real number	-3.4E+38 ≦ N ≦ -1.4E-45 N=0 +1.4E-45 ≦ N ≦ +3.4E+38 (Significant digit : 7 digit)	Used for specifying single precision floating point type real number.
Double precision floating point type real number	-1.79E+308 ≦ N ≦ -2.23E-308 N=0 +2.23E-308 ≦ N ≦ +1.79E+308 (Significant digit : 16 digit)	Used for specifying double precision
floating point type real number.	HEX($) $0000 to $FFFF (16-bit) $00000000 to $FFFFFFFF (32-bit)	Used for specifying HEX constant. * XYM Marking is represented by "H" .
Fixed text string ("")	Example "Type ABC"、"MON/08:30:15"	Used to fixed text string

■ Internal register

Device name	Range	Number of points	Main functions	When start operation (Power ON, PROG -> RUN)
Internal register	-	1 points	Device for temporary data storage (16-bit/32-bit/64-bit)	Clear

Local device list

The number of local devices/global devices and their settings when a project is created are shown as follows. Number of reserved devices can be changed through “CPU system setting” → “Assignment of all local devices setting” on the “Workspace”.

Device name*1	Local device	Global device	Total
	Reserved devices in the whole number project*2	Counts/range	
Relay（R）*3	None	32000 points (2000 CH) R000 to R199915	32000 points (2000 CH)
Internal auxiliary relay (MR)*3	32000 points (2000 CH)	32000 points (2000 CH) MR000 to MR199915	64000 points (4000 CH)
Latch relay (LR)*3	8000 points (500 CH)	8000 points (500 CH) LR000 to LR49915	16000 points (1000 CH)
Timer (T)*4	1500 points	2500 points T0 to T2499	4000 points
Counter (C)*4	1500 points	2500 points C0 to C2499	4000 points
Data memory (DM)	25000 points	40535 points DM0 to DM40534	65535 points
Expansion data memory (EM)	45000 points	20535 points EM0 to EM20534	65535 points
Temporary data memory (TM)	None	512 points TM000 to TM511	512 points

*1 Link relays (B), link registers (W), file registers (ZF), control relays (CR), control memories (CM), and index registers (Z) cannot be used as a local device.
When using @FMxxx, local devices in workspace will be occupied.

*2 When a project is created, the device area used for global devices can be guaranteed as a whole (suppose 10 modules are used and 50 counts of macro calling.)
The number of reserved devices can be changed as required.

*3 Number can be set in “ch” unit.

*4 "@+ (device code)" can be used only when the device code is specified as an operand for timer/counter instructions. (Example for inputting mnemonic : TMR @0 #100).

Point

- **If relays (R) and temporary data memories (TM) are used as a local device, the number of relays and data memories must be specified.**
- **The following devices must be used as a global device:**
  - **Devices assigned to expansion units**
  - **The devices referenced by such peripherals as touch panel**
  - **Common devices in the whole project**
  - **Devices for inter-module data transmission**
- **Where possible, local devices should be coded from 0 and use consecutive numbers.**
- **Since R and DM are assigned to the expansion unit, as a global device, there must be an area to be used. When using local devices, please use MR or EM first.**
- **If local devices or global devices are used outside the specified range, then a conversion error will occur.**
- **For the KV-7000 Series, when an index device or indirect specification is used to specify outside of the usable range of local devices or global devices, an operation error will occur.**
- **When various instructions using the leading device as the operands and index modification or indirect specifying are used, the range of the local devices and global devices that can be used should not be exceeded.**

Device table (for KV-5500 / 5000/3000/1000)

■ KV-5500/5000/3000 bit device

Device name	Range*1	Number of points	Main functions	When start operation (Power ON, PROG → RUN)	Local device (@)*1	Index Modification (: Z/: #)	Indirect designation (*)*9	
							Reference	Store
Input Relay	R000 to R015 (R1000~R99915*3)	Input/ output/ Internal auxiliary relay total 15936 points	Device used for capturing ON/OFF info from peripherals.	—	○	○	○	×
Output relay	R500 to R507 (R1000~R99915*3)		Device used for sending ON/OFF info from peripherals.	Clear/hold*4	○	○	○	×
Internal auxiliary relay	R508~R915*2 R1000~R99915*3		Used in CPU and expansion unit	Clear/hold*4	○	○	○	×
Link relay	B00~B3FFF*10	16384 points	Devices that can be only used in CPU	Clear/hold*4	×	○	○	×
Internal auxiliary relay	MR000 to MR99915	16000 points		Clear	○	○	○	×
Latch relay	LR000 to LR99915	16000 points		Hold	○	○	○	×
Timer (contact)	T0 to T3999	4000 points	10ms, 1ms, 10ms, 100ms down counter, 10ms contact device of up/ down counter	Clear	○	○	○	×
Counter (contact)	C0 to C3999	4000 points	Contact device for up counter, up/ down counter	Clear / hold*8	○	○	○	×
High-speed counter comparator (contact)	CTC0 to CTC3	4 points	Used for contact of high-speed counter	Clear / hold*8	×	×	×	×
Control relay	CR000 to CR3915	640 points	Devices used for controlling PLC function or capturing the status from peripherals.	Hold	×	×	×	×

Reference

Some input, output and internal auxiliary relays can be used for interrupt and high speed counter.

"KV-5500/5000/3000 series User's Manual", "3-6 Device and Constants"

■ KV-5500/5000/3000 word device

Device name	Range*1	Number of points	Main functions	When start operation (Power ON、PROG→RUN)	Local device (@)*1	Index Modification (: Z/: #)	Indirect designation(*)*9	
							Reference	Store
Data memory	DM0 to DM65534	65535 points	Devices that can store numerical data (16 bit)	Clear/hold*4	○	○	○	○
Expansion data memory	EM0 to EM65534	65535 points		Clear/hold*4	○	○	○	○
File register	Dial mode ZF0 to ZF131071	131072 points		Hold	×	○	○	○
	Memory: FM0 to FM32767	(32768x4Bank) point			○*12	○	○	○
Link register	W0~W3FFF*10	16384 points		Clear/hold*4	×	○	○	○
Temporary memory	TM0 to TM511	512 points	Used for storing numerical data, simple indirect specifying (16 bit)	Clear	○	○	○	○
Timer (current/ setting value)	T0 to T3999	4000 points	10ms, 1ms, 10ms, 100ms down counter, 10ms current/setting value of up/down counter (32-bit)	Current value: Clear*6 Setting value: Hold	○	○	○	×
Counter (current/ setting value)	C0 to C3999	4000 points	Up / down counter current value / setting (32-bit)	Clear / hold*8	○	○	○	×
High-speed counter (current value)	CTH0 to CTH1	2 points	High-speed counter current value (32 bit)	Clear / hold*8	×	×	×	×
High-speed counter comparator (setting value)	CTC0 to CTC3	4 points	High-speed counter set value (32 bit)	Clear/hold*4	×	×	×	×
Index register	Z1~Z12*7	12 points	Devices that can be used in index modify (32-bit)*11	Clear	×	×	×	×
Control memory	CM0 to CM5999	6000 points	Devices used for controlling PLC function or capturing the status from peripherals (16-bit).	Hold	×	×	×	×

*1 Because this is used as a local device, the applicable range of global devices is limited by the number of reserved local devices.
"Local device list (When KV-5500 / 5000/3000)", page 3-15
*2 R100 to R415, R508 to R915 are reserved for the system and cant be used.
*3 R1000 to R99915 can be set to expansion I/O relays via Unit Editor.
*4 Power OFF hold function in CPU system setting enables set clear/hold.
*5 "Control Memory Entries CM (For KV-8000)", page A-5
*6 Up/down timer (UDT) keeps current value.
*7 Z11 and Z12 are reserved for the system, so you cannot use these devices.
*8 Power OFF hold function in CPU system setting enables set clear/hold. (only for contact, current value)
9 "Store" is the device that may be described as "(device) ". "Lookup" is the device that can perform indirect specifying.
"What is indirect specifying", page 3-57
*10 Device No. is specified in HEX number.
*11 The range of value that can be processed by index register is "-131072 to +131071".
*12 The local device cannot be used for bank switching.

■ KV-1000 bit device

Device name	Range*1	Number of points	Main functions	When start operation (Power ONPROG-> RUN)	Local software (@)*1	Index Modification (: Z/: #)	Indirect designation(*)*10	
							Reference	Gner
Input Relay*3	R000 to R015 (R1000~R59915*3)	Input Output, Internal auxiliary 9536 relay points	Device for obtaining ON / OFF information from peripheral devices	-	○	○	○	×
Output relay*3	R500 to R507 (R1000~R59915*3)		Device used for sending ON/ OFF info to peripherals	Clear*4	○	○	○	×
Internal auxiliary relay	R508~R915*2 (R1000~R59915*3)		Used in CPU and expansion unit	Clear*4	○	○	○	×
	MR000~ MR99915*4	16000 points	Devices that can be only used in CPU	Clear	○	○	○	×
Internal auxiliary relay (Latch relay)	LR000~ LR99915	16000 points		Hold	○	○	○	×
Timer (contact)	T0 to T3999	Timer Total counter 4000 points	10ms, 1ms, 10ms, 100ms down counter, 10ms contact device of up/ down counter	Clear	○	○	○	×
Counter (contact)	C0 to C3999	256 points	Contact device for up counter, up/down counter	Maintain*9	○	○	○	×
High-speed counter Comparator (contact)	CTC0 to CTC3	4 points	Used for contact of high-speed counter	Maintain*9	×	×	×	×
Control relay	CR000 to CR3915	640 points	Devices used to control PLC functions and obtain device status	-	×	×	×	×

Reference

Some input, output and internal auxiliary relays can be assigned for interrupt and high speed counter.

KV-1000 User's Manual, “Chapter 4 Interrupt"
KV-1000 User's Manual, “Chapter 5 High-speed Counter"

■ KV-1000 series of word devices

Device name	Range*1	Number of points	Main functions	When start operation (power supply ONPROG ->RUN)	Local software (@)*1	Index Modification (:Z/:#)	Indirect Specifying (*)*10	
							Reference	Gner
Data memory	DM0 to DM65534	65535 points	Device for storing numerical data (16-bit)	Maintain*4	○	○	○	○
Link register	EM0~ EM65534*5	65535 points		Maintain*4	○	○	○	○
	FM0~ FM32776*5	32767 points		Maintain*4	○	○	○	○
Temporary memory	TM0 to TM511	512 points	Device for storing numerical data (16-bit)	Clear	○	○	○	○
Timer (current/ setting value)	T0 to T3999	Timer Total counter 4000 points	10ms, 1ms, 10ms, 100ms down counter, 10ms current/ setting value of up/down counter (32-bit)	Current value: Clear*7 Setting value: Hold	○	○	○	×
Counter (current/ setting value)	C0 to C3999		Up counter, up/down counter (32 bit)	Maintain*9	○	○	○	×
High-speed counter (current value)	CTH0 to CTH1	2 point*6	High-speed counter current value (32 bit)	Maintain*9	×	×	×	×
High-speed counter Comparator (setting value)	CTC0 to CTC3	4 point*6	High-speed counter set value (32 bit)	Hold	×	×	×	×
Index register	Z1~Z12*8	12 points	Devices that can be used in index modify (32-bit)	Clear	×	×	×	×
Control memory	CM0~ CM11998	11999 points	Device for controlling PLC functions and obtaining device status (16 bits)	-	×	×	×	×

*1 Global device range is limited to the number reserved for local device.
"Local device list (At KV-1000)", page 3-16
*2 R100 to R415 are reserved for system and cannot be used.
*3 R1000 to R59915 can be set to expansion I/O relays via Unit Editor.
*4 MEMSW instruction allows to set clear/holding.
*5 According to assignment setting of work devices used in script, some areas are reserved for system.
*6 " CR/CM List", page A-2
*7 Up/down timer (UDT) to keep current value.
*8 Z11 and Z12 are reserved for the system, so you cannot use these devices.
*9 MEMSW instruction allows to set hold/clear. (only for contact, current value)
10 "Store" is the device that may be described as "(device)". "Lookup" is the device that can perform indirect specifying.
"What is indirect specifying", page 3-57

■ Macro argument device

The macro argument devices used only within the macro are as follows.

Attribute		Symbol	Range	Description	Indirect Specifying (*)	Local device (@)	Index modification (:)
Device		P	P0 to P9	Devices are used as argument.	×	×	○
Value		V	V0 to V9	Devices are used as argument. Device values or constants are used as argument for processing.	×	×	×
Unit	Start R	UR	UR0 to UR9	Leading device assigned to extended unit processed as argument.	×	×	○
	Start DM	UM	UM0 to UM9	The leading device assigned to extended unit is processed as an argument.	×	×	○
	No.	UV	UV0 to UV9	Treat the unit number of the expansion unit as an argument.	×	×	×

■ Const

Const	Range	Main functions
10 DEC(#)	(#) 0 to (#) 65535 (unsigned 16-bit) (#) -32768 to +32767 (signed 16-bit) (#) 0 to (#) 4294967295 (unsigned 32-bit) (#) -2147483648 to +2147483647 (signed 32-bit)	Used to specify DEC constant. ("#" can be omitted.) Reference When set to XYM display mode, the display will be "K". (K can be omitted.)
Single precision floating point real numbers	（#） -3.4E+38 ≦ N ≦ （#） -1.4E-45 N = （#） 0 （#） +1.4E-45 ≦ N ≦ （#） +3.4E+38 (Significant digit : 7 digit)	Used to specify single precision floating point type constant. ("#" can be omitted.)
Double precision floating point real numbers	（#） -1.79E+308 ≦ N ≦ （#） -2.23E-308 N = （#） 0 （#） +2.23E-308 ≦ N ≦ （#） +1.79E+308 (Significant digit : 16 digit)	Used to specify double precision floating point type constant. ("#" can be omitted.)
floating point type real number ($).	$0 to $FFFF (16-bit) $0 to $FFFFFFFF (32-bit)	Used to specify HEX constants. Reference When set to XYM display mode, the display is "H".
Fixed text string ("")	Example "Type ABC", "MON/08:30:15"	Used to fixed text string

■ Internal register

The status of the bit device is described in the following representation method.

Device name	Range	Number of points	Main functions	When the operation starts (power ON, PROG → RUN)
Internal register	-	1 points	Device for temporary data storage (16-bit/32-bit/64-bit)	Empty

Local device list (When KV-5500 / 5000/3000)

The number of local devices/global devices and their settings when a project is created are shown as follows. Number of reserved devices can be changed through “CPU system setting” → “Assignment of all local devices setting” on the “Workspace”.

Device name*1	Local device	Global device	Total
	Reserved devices in the whole number project*2	Counts/range	
Relay(R) *3	None	16000 points (1000 CH) R000 to R99915	16000 points (1000 CH)
Internal auxiliary relay (MR)*3	8000 points (500 CH)	8000 points (500 CH) MR000 to MR49915	16000 points (1000 CH)
Latch relay (LR)*3	8000 points (500 CH)	8000 points (500 CH) LR000 to LR49915	16000 points (1000 CH)
Timer (T)*4	1500 points	2500 points T0 to T2499	4000 points
Counter (C)*4	1500 points	2500 points C0 to C2499	4000 points
Data memory (DM)	25000 points	40535 points DM0 to DM40534	65535 points
Expansion data memory (EM)	45000 points	20535 points EM0 to EM20534	65535 points
Temporary data memory (TM)	None	512 points TM000 to TM511	512 points

*1 Link relay (B), link register (W), file register (ZF), high-speed counter (CTH), high-speed counter comparator (CTC), control relay (CR), control storage (CM), index register (Z) Cannot be used as a local device.
When using @FMxxx, local devices in workspace will be occupied.

*2 When a project is created, the device area used for global devices can be guaranteed as a whole (suppose 10 modules are used and 50 counts of macro calling.)
The number of reserved devices can be changed as required.

*3 Number can be set in "ch" unit.

*4 "@+ (device code)" can be used only when the device code is specified as an operand for timer/counter instructions.
(Example for inputting mnemonic : TMR @0 #100).

Point

- **If relays (R) and temporary data memories (TM) are used as a local device, the number of relays and data memories must be specified.**
- **The following devices must be used as a global device:**
  - **KV-5500/5000/3000 I/O unit**
  - **The devices referenced by such peripherals as touch panel**
  - **Common devices in the whole project**
  - **Devices for inter-module data transmission**
- **Where possible, local devices should be coded from 0 and use consecutive numbers.**
- **Since R and DM are assigned to the expansion unit, as a global device, there must be an area to be used. When using local devices, please use MR or EM first.**
- **If local devices or global devices are used outside the specified range, then a conversion error will occur.**
- **When various instructions using the leading device as the operands and index modification or indirect specifying are used, the range of the local devices and global devices that can be used should not be exceeded.**

Local device list (At KV-1000)

The number of local devices/global devices and their settings when a project is created are shown as follows. Number of reserved devices can be changed through “CPU system setting” → “Assignment of all local devices setting” on the “Workspace”.

Device name*1	Local device	Global device	Total
	Reserved devices in the whole number project*2	Counts/range	
Relay（R）*3	None	9600 points (600 CH) R00000 to R59915	9600 points (600 CH)
Internal auxiliary relay (MR)*3	8000 points (500 CH)	7200 points (450 CH) MR000 to MR44915	16000 points (1000 CH)
Latch relay (LR)*3	8000 points (500 CH)	8000 points (500 CH) LR000 to LR49915	16000 points (1000 CH)
Timer (T)*5	1500 points	Timer / counter total 1000 points T0000 to T0999 C0000 to C0999	Timer / counter total 4000 points
Counter (C)*5	1500 points		
Data memory (DM)	25000 points	40535 points DM0 to DM40534	65535 points
Expansion data memory (EM)	45000 points	19535 points EM0 to EM19534	65535 points*4
Expansion data memory (FM)	20000 points	11767 points FM00000 to FM11766	32767 points*4
Temporary data memory (TM)	None	512 points TM000 to TM511	512 points

High-speed counters (CTH), high-speed counter comparator (CTC), control relays (CR), control memories (CM), and index registers (Z) cannot be used as a device.

*2 When a project is created, the device area used for global devices can be guaranteed as a whole (suppose 10 modules are used and 50 counts of macro calling.)
The number of reserved devices can be changed as required.

*3 Number can be set in “ch” unit.

*4 This number includes the workspace used for script devices.

*5 "@+ (device code)" can be used only when the device code is specified as an operand for timer/counter instructions.

(Example for inputting mnemonic : TMR @0 #100).

Point

- **If relays (R) and temporary data memories (TM) are used as a local device, the number of relays and data memories must be specified.**
- **The following devices must be used as a global device:**
  - **KV-1000 I/O unit**
  - **The devices referenced by such peripherals as touch panel**
  - **Common devices in the whole project**
  - **Devices for inter-module data transmission**
- **Where possible, local devices should be coded from 0 and use consecutive numbers.**
- **Since R and DM are assigned to the expansion unit, as a global device, there must be an area to be used. When using local devices, please use MR or EM first.**
- **If local devices or global devices are used outside the specified range, then a conversion error will occur.**
- **When various instructions using the leading device as the operands and index modification or indirect specifying are used, the range of the local devices and global devices that can be used should not be exceeded.**

Device table (for KV Nano)

■ KV Nano Series bit device

Device name	Range*1	Number of points	Main functions	When start operation		Local software (@)*1	Index Modification (:Z/:#)	Indirect designation(*)*2	
				Power ON	PROG->RUN			Reference	Gner
Input Relay*3	R000~R203*4 (R1000~R59915*5)	Input Output, Internal auxiliary relay total 9600 points	Device for obtaining ON / OFF information from peripheral devices	Clear	Clear	○	○	○	×
Output relay*3	R500~R607*6 (R1000~R59915*5)		Device used for sending ON/ OFF info to peripherals			○	○	○	×
Internal auxiliary relay	R1000~R59915*3		Used in CPU and expansion unit			○	○	○	×
Link relay	B00~B1FFF*8	8192 points	Devices that can be only used in CPU	Clear	Clear	×	○	○	×
Internal auxiliary relay	MR000 to MR59915	9600 points		Clear	Clear	○	○	○	×
Latch relay	LR000 to LR19915	3200 points		Hold	Hold	○	○	○	×
Timer (contact)	T0 to T511	512 points	10ms, 1ms, 10ms, 100ms down counter, 10ms contact device of up/ down counter	Of T0 to T9, only devices used by the UDT instruction are held	Of T0 to T9, only devices used by the UDT instruction are held	○	○	○	×
Counter (contact)	C0 to C255	256 points	Contact device for up counter, up/down counter	Clear/hold*7	Hold	○	○	○	×
High-speed counter Comparator (contact)	CTC0~CTC7*9	8 points*9	Used for contact of high-speed counter	Hold	Hold	×	×	×	×
Control relay	CR000 to CR8915	1440 points	Devices used to control PLC functions and obtain device status	Hold	Clear	×	×	×	×

*1 Because this is used as a local device, the applicable range of global devices is limited by the number of reserved local devices. "Local device list", page 3-20

2 "Store" is the device that may be described as "(device) ". "Lookup" is the device that can perform indirect specifying. KV Nano Series User's Manual, "Indirect Specifying"

*3 For direct processing, described with "DR".

*4 The range of devices that can be used varies according to the unit.
KV-N14 □□：R000 ~ R007、KV-N24 □□：R000 ~ R013
KV-N40 □□：R000 ~ R107、KV-N60 □□：R000 ~ R203
KV-NC32T: R000 to R015

*5 R1000 to R59915 can be set to expansion I/O relays via Unit Editor.
Relays not assigned to input and output relays less than R1000 are reserved for the system and cannot be used.

*6 The range of devices that can be used varies according to the unit.
KV-N14 □□：R500 ~ R505、KV-N24 □□：R500 ~ R509
KV-N40 □□：R500 ~ R515、KV-N60 □□：R500 ~ R607
KV-NC32T: R500 to R515

*7 The clear / hold setting can be set by the "power failure hold setting" set by the CPU system. (only for contact, current value)

*8 Device No. is specified in HEX number.

*9 The range of devices that can be used varies according to the unit.
KV-N14 □□：CTC0 ~ CTC3 (4 point)、KV-N24 □□：CTC0 ~ CTC3 (4 point)
KV-N40 □□：CTC0 ~ CTC5 (6 point)、KV-N60 □□：CTC0 ~ CTC7 (8 point)
KV-NC32T：CTC0 ~ CTC5 (6 point)

■ KV Nano series of word devices

Device name	Range*1	Number of points	Main functions	When start operation		Local software (@)*1	Index Modification (:Z/:#)	Indirect Specifying (*)*9	
				Power ON	PROG->RUN			Reference	Gner
Data memory	DM0 to DM32767	32768 points	Device for storing numerical data (16-bit)	Clear/hold*3	Clear/hold*3	○	○	○	○
Link register	W0~W3FFF*5	16384 points		Clear/hold*3	Hold	×	○	○	○
Temporary memory	TM0 to TM511	512 points	Device for storing numerical data (16-bit)	Clear	Clear	○	○	○	○
Timer (current/setting value)	T0 to T511	512 points	10ms, 1ms, 10ms, 100ms down counter, 10ms current/setting value of up/down counter (32-bit)	Of T0 to T9, only devices used by the UDT instruction are held	Of T0 to T9, only devices used by the UDT instruction are held	○	○	○	×
Counter (current/setting value)	C0 to C255	256 points	Up counter, up/down counter (32 bit)	Clear/hold*4	Clear/hold*4	○	○	○	×
High-speed counter (current value)	CTH0~CTH3*6	4 points*6	High-speed counter current value (32 bit)	Hold	Hold	×	×	×	×
High-speed counter Comparator (setting value)	CTC0~CTC7*6	8 points*6	High-speed counter set value (32 bit)	Hold	Hold	×	×	×	×
Index register	Z1~Z12*7	12 points	Devices that can be used in index modify (32-bit)	Clear	Hold	×	×	×	×
Control memory	CM0 to CM8999	9000 points	Device for controlling PLC functions and obtaining device status (16 bits)	Maintain*8	Maintain*8	×	○	○	×

*1 Because this is used as a local device, the applicable range of global devices is limited by the number of reserved local devices.
"Local device list", page 3-20

2 "Store" is the device that may be described as "(device) ". "Lookup" is the device that can perform indirect specifying.

*3 "KV Nano Series User's Manual", "Indirect Specification"

*4 Power OFF hold function in CPU system setting enables set clear/hold. It can hold up to 3072 words together with other devices that can set the power-down holding function.

*5 The hold / clear setting can be set in the "power-off hold setting" set in the CPU system. (only for contact, current value)

*6 Specify the device number by hexadecimal number.

*7 The range of devices that can be used varies according to the unit.
KV-N14 □□：CTH0 ~ CTH1 (2 points) CTC0 ~ CTC3 (4 points)、KV-N24 □□：CTH0 ~ CTH1 (2 points) CTC0 ~ CTC3 (4 points)
KV-N40 □□：CTH0 ~ CTH2 (3 points) CTC0 ~ CTC5 (6 points)、KV-N60 □□：CTH0 ~ CTH3 (4 points) CTC0 ~ CTC7 (8 points)
KV-NC32T：CTH0 ~ CTH2 (3 points) 、CTC0 ~ CTC5 (6 points)

*8 Z11 and Z12 are reserved for the system, so you cannot use these devices.

*9 Only for devices that need to be held. For details on the target devices, see "KV Nano Series User's Manual", "List of CR and CM Devices" .

■ Macro argument devices

The macro argument devices used only within the macro are as follows.

Attribute		Symbol	Range	Description	Indirect Specifying (*)	Local device (@)	Index modification (:)
Device		P	P0 to P9	Devices are used as argument.	×	×	○
Value		V	V0 ~V9	Devices are used as argument. Device values or constants are used as argument for processing.	×	×	×
Unit	Start R	UR	UR0 to UR9	Leading device assigned to extended unit processed as argument.	×	×	○
	Start DM	UM	UM0 to UM9	The leading device assigned to extended unit is processed as an argument.	×	×	○
	No.	UV	UV0 to UV9	Treat the unit number of the expansion unit as an argument.	×	×	×

■ Const

Const	Range	Main functions
DEC (#)	#0 to #65535 (Unsigned 16-bit) - 32768 to +32767 (Signed 16-bit) #0 to #4294967295 (Unsigned 32-bit) -2147483648 ~ +2147483647 (Signed 32-bit)	Used to specify DEC constant. * Displayed as "K" when using the XYM display mode.
Single precision floating point real numbers	- 3.4E+38 ≦ N ≦- 1.2E - 38 N ＝ 0 +1.2E - 38 ≦ N ≦ +3.4E+38 (Significant digit : 7 digit)	Used to specify single precision floating point type constant. A denormal (-1.2E-38 ≦ N ≦ - 1.4E-45、 +1.4E-45 ≦ N ≦ +1.2E-38) cannot be specified. If the result is denormal, 0 is returned.
Double precision floating point real numbers	-1.79E+ 308 ≦ N ≦- 2.23E -308 N ＝ 0 +2.23E- 308 ≦ N ≦ +1.79E+308 (Significant digit : 16 digit)	Used to specify double precision floating point type constant.
HEX ($)	$0000 to $FFFF (16 bits) $00000000 to $FFFFFFFF (32 bits)	Used to specify HEX constants. * Displayed as "H" when using the XYM display mode.
Fixed text string ("")	Example "Type ABC", "MON/08:30:15"	Used to fixed text string

■ Internal register

The status of the bit device is described in the following representation method.

Device name	Range	Number of points	Main functions	When the operation starts (power ON, PROG → RUN)
Internal register	-	1 points	Device for temporary data storage (16-bit/32-bit/64-bit)	Empty

Local device list

The number of local devices/global devices and their settings when a project is created are shown as follows. Number of reserved devices can be changed through “CPU system setting” → “Assignment of all local devices setting” on the “Workspace”.

Device name*1	Local device	Global device	Total
	Reserved devices in the whole number project*2	Counts/range	
Relay（R）*3	None	9600 points (600 CH) R000 to R59915	9600 points (600 CH)
Internal auxiliary relay (MR)*3	1600 points (100 CH)	8000 points (500 CH) MR000 to MR49915	9600 points (600 CH)
Latch relay (LR)*3	320 points (20 CH)	3560 points (180 CH) LR000 to LR17915	3200 points (200 CH)
Timer (T)*4	100 points	412 points T0 to T411	512 points
Counter (C)*4	50 points	206 points C0 to C205	256 points
Data memory (DM)	5000 points	27768 points DM0 to DM27767	32768 points
Temporary data memory (TM)	None	512 points TM000 to TM511	512 points

*1 Link relays (B), link registers (W), high-speed counters (CTH), high-speed counter comparators (CTC), control relays (CR), control memory entries (CM), and index registers (Z) cannot be used as local devices.

*2 When a project is created, the device area used for global devices can be guaranteed as a whole (suppose 10 modules are used and 50 counts of macro calling.)
The number of reserved devices can be changed as required.

*3 Number can be set in “ch” unit.

*4 "@+ (device code)" can be used only when the device code is specified as an operand for timer/counter instructions.

(Example for inputting mnemonic : TMR @0 #100).

Point

- **If relays (R) and temporary data memories (TM) are used as a local device, the number of relays and data memories must be specified.**
- **The following devices must be used as a global device:**
  - **KV Nano Series I/O relays**
  - **The devices referenced by such peripherals as touch panel**
  - **Common devices in the whole project**
  - **Devices for inter-module data transmission**
- **Where possible, local devices should be coded from 0 and use consecutive numbers.**
- **Since R and DM are assigned to the expansion unit, as a global device, there must be an area to be used. When using local devices, please use MR or EM first.**
- **If local devices or global devices are used outside the specified range, then a conversion error will occur.**
- **When various instructions using the leading device as the operands and index modification or indirect specifying are used, the range of the local devices and global devices that can be used should not be exceeded.**

Device notation method

● Writing index modification (modification according to index registers and according to index constants)

• Modify via index register

Continuously enter：Z□ (□: numerical value 1 to 10) after device (R (DR) / MR / LR / B / T / C / DM / EM / FM / ZF / W / TM / *DM / *EM / *FM / *ZF / *W / *TM).

Example

DM0:Z1	If the value of Z1 is 100, this means DM100.
R1000:Z1	If the value of Z1 is 100, this means R1604.

Point

- **Device Name KEYENCE**
- **The modification by index register towards CM is only possible for KV Nano series.**

• Modify via index constant

After the device (R (DR)/MR/LR/B/T/C/DM/EM/FM/ZF/W/TM/*DM/*EM/*FM/*ZF/*W/*TM) Then enter: #□（□：- 524287 ~ +524287）.

Example

DM0:#1	This means DM1.
R1000:#1	This means R1001.

Point

- **Device Name KEYENCE**
- **The modification by index register towards CM is only possible for KV Nano series.**

• Indexing by operation expression

Enter the operation expression after the device (R (DR)/MR/LR/B/T/C/DM/W/TM/CM/*DM/* W/*TM).

Example

DM0: (DM100 + 20) If the value of DM100 is 50, it means DM70

R1000: (DM0 * 4) If the value of DM0 is 2, it means R1008

Point

- **Indexing restriction by operation expression can only be used in KV script.**
- **The EM, FM can not be used in KV Nano series.**
- **Only in the KV Nano series, it is possible to add arithmetic expression limits to CM.**

● Writing indirect specification (*)

Enter * in front of device (DM / EM / FM / ZF / W / TM). It is to be used together with indirect designation command.

Example

*DM0 *TM100

"What is indirect specifying", page 3-57

Point

The EM, FM can not be used in KV Nano series.

● Writing local devices

Enter @ in front of device (R/MR/LR/T/C/DM/EM/FM/TM).

These devices are only valid in modules and macros. You can also perform indirect specification and index modification.

Example

@DM0 @R10000

Point

The EM, FM can not be used in KV Nano series.

● XYM marking

The designation of device by XYM notation of the list below is possible for device (R(DR)/MR/LR/DM/EM/FM).

Device name	KEYENCE Marking	XYM Marking	Device Number during
Relay	R（DR）	X（DX）	0000 to 999F*1
		Y（DY）	0000 to 999F*1
Internal auxiliary relay	MR	M	00000 to 63999*1*2
Latch relay	LR	L	0000 to 999F*1
Data memory	DM	D	00000 to 65534*1
Expansion data memory	EM	E	00000 to 65534
File register	FM	F	00000 to 32767

*1 The scope of device number that can be used differs for KV Nano series.
X(DX):0000 to 599F
Y(DY):0000 to 599F
M:00000 to15999
‘ L:0000 to199F
D:00000 to32767

*2 The scope of device number that can be used differs for KV-5500/5000/3000 series.
M:00000 to15999

To display it in XYM notation, select “Tool” → “Option” from the KV STUDIO menu and put a check on “Display in XYM” with “Configuration of Display Mode”.

Point

The EM, FM can not be used in KV Nano series.

● Entry specifying bits of a word device

After the device (R (DR) / MR / LR / B / T / C / DM / EM / FM / ZF / W / TM / CM), continue to input. □ (□: the value of 0 to 15).

Example

DM0.0 (Indicates the first DM0 bit)
EM10.4 (Indicates the fifth EM10.4)

"Processing Word Device as Bit Device", page 3-46

Point

The EM, FM can not be used in KV Nano series.

● Suffix entry

Soft element (R (DR) / MR / LR / B / T / C / DM / EM / FM / ZF / W / TM / CM / * DM / * EM / * FM / * ZF / * W / * TM) Then continue to input. □ (□: U, S, D, L, F, DF, B, T).

Example

DM0.F (Values DM0 - DM1 are calculated as floating point data)
R1000.U (calculated as 16-bit data between R1000 - R1015)

"What is indirect specifying", page 3-57

Point

- **Suffix entries can be used only in KV scripts.**
- **The EM, FM can not be used in KV Nano series.**

● Constant value type specification

With the KV script, by adding a suffix to the decimal, a range with constants can be set.

Type	Suffix	Data range	Program example
Unsigned 16-bit	Constant U	0 to 65535	910U, 505U
Signed 16-bit	Constant S	− 32768 to + 32767	23S, − 600S
Unsigned 32-bit	Constant D	0 to 4294967295	1234567D, 5D
Signed 32-bit	Constant L	− 2147483648 to + 2147483647	123L, − 89335963L
Single precision floating point type (32-bit)	Constant F	− 3.4E + 38 ≦ N ≦ − 1.4E − 45 N = 0 + 1.4E − 45 ≦ N ≦ + 3.4E + 38 (* For the KV Nano Series, an absolute value in a range of 1.4E - 45 to 1.2E - 38 cannot be used.)	100F, −3.4E38F,0.12F
Double precision floating point type (64-bit)	Constant DF	− 1.79E + 308 ≦ N ≦ − 2.23E − 308 N = 0 + 2.23E − 308 ≦ N ≦ + 1.79E + 308	100DF, − 1.7E123DF 0.0256DF

* The type of hexadecimal constant handled in KV script is determined by the number of digits. (suffixes cannot be entered)
4 digits or less…16bit unsigned (.U)
5 digits or more…32bit unsigned (.D)

Bit Device

■ Bit status

The status of the bit device is described in the following representation method.

Bit load	Representation	Program example
ON（TRUE）	ON, 1, TRUE	MR1000=ON, MR1001=1, MR1002=TRUE
OFF（FALSE）	OFF, 0, FALSE	MR2000=OFF, MR2001=0, MR2002=FALSE

Reference **These 2 status called "Boolean value".**

I/O relay R

Overview

Input relay is the device that acquires ON/OFF information from peripheral equipment into PLC. Output relay is the device that sends ON/OFF instruction or arithmetic operation result from PLC to PLC peripheral equipment.

Denotation

Device No. composition as follows.

Example) Channel No. (000 to 999*1) Contact No. (00 to 15)

R00012

Device name Device No. (00000 to 99915*2)

*1 When CPU unit is KV-1000 or KV Nano Series, the range is "000 to 599"
*2 When CPU unit is KV-1000, the range is "000 to 59915"

Reference

- Input allows zero suppression when programming.
  Example) R12
- Suffix can be added when programming
  Example) R12.B
- When suffix is omitted, the type will be processed as .B (bit type), unless type declaration is available.
- Processed as word device.
  Example) R2000.U (16 bits occupied from R2000)
- When CPU unit is KV-8000/7500/7300/5500/5000/3000 or KV Nano Series, if adding "D" to the device name, it will operate as direct I/O relay.
  Example) DR500 (R500 operates as direct output relay)
  "Direct I/O Relay", page 3-47

Range

- R000 to R015 * (special for input)
- R500 to R507 (special for output)
- R1000 to R199915 * (can be assigned as the expansion input unit)

* The relay numbers that can be used differ by model.

	Special for input	Special for output	Assigned number of the expansion input unit
KV-8000	-	-	R1000 to R199915
KV-7500/7300	-	-	R1000 ~ R199915*1
KV-5500/5000/3000	R000 to R015	R500 to R507	R1000 to R99915
KV-1000	R000 to R015	R500 to R507	R1000 to R59915
KV-N14□□	R000 to R007	R500 to R505	R1000 to R59915
KV-N24□□	R000 to R013	R500 to R509	R1000 to R59915
KV-N40□□	R000 to R107	R500 to R515	R1000 to R59915
KV-N60□□	R000 to R203	R500 to R607	R1000 to R59915
KV-NC32T	R000 to R015	R500 to R515	R1000 to R99915

*1 For CPU function versions older than 2.2, the range is R00000 to R99915.

Usage

Input Relay

- Used peripheral equipment connected, including button and switch (for example Selector Switch, limit switch, photoelectric switch, digital switch), etc.
- Input relay can be used as high speed inputs such as interrupt, high speed counter, positioning function etc.

(The input relays that can be used differ per unit.)

KV-5500/5000/3000/1000	R000 ~ R015
KV-N14 □□	R000 ~ R007
KV-N24 □□	R000 ~ R013
KV-N40 □□ /KV-N60 □□	R000 ~ R107
KV-NC32T	R000 ~ R015

Output relay

- Used the peripheral equipment connected, including solenoid valve, electromagnetic switch, display (signal indicator, numerical indicator), etc.
- Output relay can be used as high speed output of high speed counter and positioning function, etc.

(The output relays that can be used differ per unit.)

KV-5500/5000/3000/1000	R500 ~ R503
KV-N14 □□ /KV-N24 □□	R500 ~ R503
KV-N40 □□	R500 ~ R505
KV-N60 □□	R500 ~ R507
KV-NC32T	R500 ~ R505

Description

- Assign 1 bit to every input for input relay.
- Assign 1 bit to every output for output relay.
- NO or NC contact can be specified.
- In the same project, the input relay with same Device No. can be programed for many times.
- Of R000 to R915, relays that are not used by inputs or outputs are reserved for the system and cannot be used.

Default

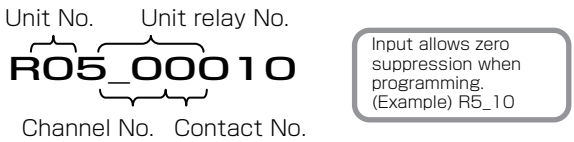
Default setting of output relay: shutoff and cleared when powering ON and when PPROG → RUN.

- When CPU unit is KV-8000/7500/7300/5500/5000/3000 or KV Nano Series, power off hold function in CPU system setting allows to set up holding status.
- When CPU unit is KV-1000, MEMSW instruction allows to set up holding status.

Reference

- About representation method of relay (R) unit device

In KV STUDIO , except general representation method, the relay can also be represented according to Unit No.. If representing with devices used for unit, you can select "View (V)" → "Display devices for unit (Y)" from the menu.



Unit number : Starting from CPU, according to the unit connection sequence, assign00, 01, 02, 03, 04...., and the like.

Unit relay No. : Number of relay points occupied from 00000 are assigned to relay No.

Unit No.	00※	01	02	03	04	05
KV-U7	CPU unit	KV-B16XA 16-input	KV-C32XA 32-input	KV-B16TA 16-output	KV-C32TA 32-output	KV-H20 10CH occupied
Unit relay No.	R0000 to R00015 (input) R00500 to R00507 (output)	00000～00015	00000～00115	00000～00015	00000～00115	00000～00915

* Devices from R000 to R915 used by the CPU (base) unit cannot be used as unit devices.

Link relay B

Overview

This device cannot be used when the compatible model (CPU unit or base unit to use) is the KV-1000.
It is used as link relay when connecting via FL-net, etc. Different from other relays, device No. is specified in HEX.

Denotation

Device No. composition as follows.

Example)

Channel No. (000 to 7FF) Contact No. (0 to F)

B1EE7

Device name Device No. (0000 to 7FFF)

Reference

- Input allows zero suppression when programming.
  Example) B7A
- Suffix can be added when programming
  Example) B7A.B
- When suffix is omitted, the type will be processed as .B (bit type), unless type declaration is available.
- Processed as word device.
  Example) B80.U (16 bits occupied starting from B80,)

Point

- **Please use HEX number (0 to F) to specify.**
- **Local device (@) cannot be used.**

Range

B0000~B7FFF*

Usage

It is used as link relay when connecting via FL-net, etc.

Description

- These relays cannot be assigned as the unit device of expansion unit.
- In the same project, the link relay with same Device No. can be programmed for many times.

Default

Default setting of link relay: shutoff and cleared when powering ON and when PPROG→RUN. Hold mode setting can be performed utilizing “Power off hold function" in CPU system setting.

* The relay numbers that can be used differ by model.

KV-8000	B0000 to B7FFF
KV-7500/7300	B0000 to B7FFF
KV-5500/5000/3000	B0000 to B3FFF
KV-1000	---
KV Nano Series	B0000 to B1FFF

Internal auxiliary relay R/B/MR/LR

■ R

Overview

R unassigned to actual I/O relay and expansion unit relay is used as internal auxiliary relay.

Denotation

Device No. composition as follows.

Example)

Channel No. (000 to 999*1) Contact No. (00 to 15)

R00012

Device name Device No. (00000 to 99915*2)

*1 When CPU unit is KV-1000 or KV Nano Series, the range is "000 to 599"

*2 When CPU unit is KV-1000 or KV Nano Series, the range is "000 to 59915"

Reference

- Input allows zero suppression when programming.
  Example) R12
- Suffix can be added when programming
  Example) R12.B
- When suffix is omitted, the type will be processed as .B (bit type), unless type declaration is available.
- Processed as word device.
  Example) R2000.U (16 bits occupied from R2000)

Range

- R1000 to R199915* (can be assigned as the expansion input unit No.)

  * The relay numbers that can be used differ by model.

KV-8000	: R000 to R199915
KV-7500/7300	: R000 to R199915*1
KV-5500/5000/3000	: R1000 to R99915
KV-1000	: R1000 to R59915
KV Nano Series	: R1000 to R59915

  *1 For CPU function versions older than 2.2, the range is R00000 to R99915.

■ B

Overview

This device cannot be used when the compatible model (CPU unit to use) is the KV-1000. When FL-net is not connected, B that is not assigned to link relay can be used as internal auxiliary relay.

Denotation

Device No. composition as follows.

Example)

Reference

- Input allows zero suppression when programming.
  Example) B7A
- Suffix can be added when programming
  Example) B7A.B
- When suffix is omitted, the type will be processed as .B (bit type), unless type declaration is available.
- Processed as word device.
  Example) B80.U (16 bits occupied from B80)

Point

- **Please use HEX number (0 to F) to specify.**
- **Local device (@) cannot be used.**

Range

B0000~B7FFF*

Description

- These relays cannot be assigned as the unit device of expansion unit.
- In the same project, the link relay with same Device No. can be programmed for many times.

Default

Under default status, output is off when powering ON and when status is cleared during PPROG ←→ RUN. Power off hold function in CPU system setting allows to set up holding status.

* The relay numbers that can be used differ by model.

KV-8000	B0000 to B7FFF
KV-7500/7300	B0000 to B7FFF
KV-5500/5000/3000	B0000 to B3FFF
KV-1000	---
KV Nano Series	B0000 to B1FFF

■ MR

Overview

The device that can be used only inside CPU.

Denotation

Device No. composition as follows.

Example)

Reference

- Input allows zero suppression when programming.
  Example) MR2809
- Suffix can be added when programming
  Example) MR2809.B
- When suffix is omitted, the type will be processed as .B (bit type), unless type declaration is available.
- Processed as word device.
  Example) MR2900.U (16 bits occupied from MR2900)

Range

MR00000~MR399915*

Point

By default, the range for the KV-8000/7000 Series is MR200000 to MR399915, and the range for the KV-5000/3000/1000 and KV Nano Series is MR50000 to MR99915.* indicates it cannot be used because it is already assigned to local device.
(The reserved area can be changed using "whole local assignment setting" set in CPU system setting)

Description

- These relays cannot be assigned as the unit device of expansion unit.
- In the same project, the MR with same Device No. can be programmed for many times.

Default

When powering ON, or PROG->RUN, status will be cleared, output will be OFF.

* The relay numbers that can be used differ by model.
KV-8000：
MR00000 to MR399915 (MR200000 to MR399915 are local devices in the initial state)
KV-7500/7300:
MR00000 to MR399915 (MR200000 to MR399915 are local devices in the initial state)
KV-5500/5000/3000/1000:
MR00000 to MR99915 (MR50000 to MR99915 are local devices in the initial state)
KV Nano Series:
MR00000 to MR59915 (MR50000 to MR59915 are local devices in the initial state)

■ LR (Latch relay)

Overview

This device could only be used in CPU to hold status.

Denotation

Device No. composition as follows.

Example)

Reference

- Input allows zero suppression when programming.
  Example) LR2308
- Suffix can be added when programming
  Example) LR2308.B
- When suffix is omitted, the type will be processed as .B (bit type), unless type declaration is available.
- Processed as word device.
  Example) LR2400.U (16 bits occupied from LR2400)

Range

LR00000~LR99915*

Description

- These relays cannot be assigned as the unit device of expansion unit.
- In one project, the latch relay with same Device No. can be programmed for many times.

Default

Hold the status before the last power OFF.

* The relay numbers that can be used differ by model.
KV-8000/7500/7300/5500/5000/3000/1000：LR00000 to LR99915
KV Nano Series: LR00000 to LR19915

Timer (contact) T

Overview

Timer including contact (1 bit) and current value (32 bit). Timer is a device for turning contact ON, when current value of down timer is equal to "0".

Denotation

Device No. composition as follows.

Example)

Device No. (0000 to 3999*)

T0501.B

Device name Suffix

Reference

- Input allows zero suppression when programming.
  Example) T501.B
- Suffix can be added when programming When used as timer contact, suffix.B must be attached; when processing current value of timer, suffix.D can be attached.
- When suffix is omitted, the type will be processed as .B (bit type), unless type declaration is available.

Range

T0000.B to T3999.B*

Description

4 timer functions available: 10ms timer, 1ms timer, 10ms timer, 100ms timer.

Default

When powering ON, or PROG → RUN , status will be cleared, output will be OFF.

Current value is identical to setting value when powering ON or PR O G → RUN.

Setting value is held.

* The relay numbers that can be used differ by model.
KV-8000/7500/7300/5500/5000/3000/1000：T0000.B to T3999.B
KV Nano Series: T000.B to T511.B

■ How to program the timer when timer function is operated via scripts

Specifying via execution condition of timer for timer function argument (timer reset relay), timer No. and setting value (time before time up), enables programming the timer's operation.

Example)

Timer device (T0000 to T3999*) Setting vale (0 to 4294967295, device No.)

TMR (R1000, T501, 1000)

Timer function name (100ms down timer) Timer execution condition

[Operation difference of every timer]

Timer type (function)	Device No.	Setting value	Time before ON
100ms Down Counter (TMR)	T0~T3999*1	0~4294967295	n x100ms
10ms Down Counter (TMH)			n x10ms
1ms Down Counter (TMS)			n x 1ms
10μs down Timer (TMU)*2			n x 10μs

*1 The device numbers that can be used differ according to the model.
KV-8000/7500/7300/5500/5000/3000/1000：T0000.B to T3999.B
KV Nano Series: T000.B to T511.B

*2 This function cannot be used with the KV-1000.

Reference

The device (unsigned 32-bit data) and formula can be specified in setting value.

"Timer / Counter function", page 5-22

■ Timer error

Timer error as follows:

TMR (100ms Timer) : within± (100ms + 1 scanning time)

TMH (10ms Timer) : within± (10ms + 1 scanning time)

TMS (1ms Timer) : within ± (1ms + 1 scanning time)

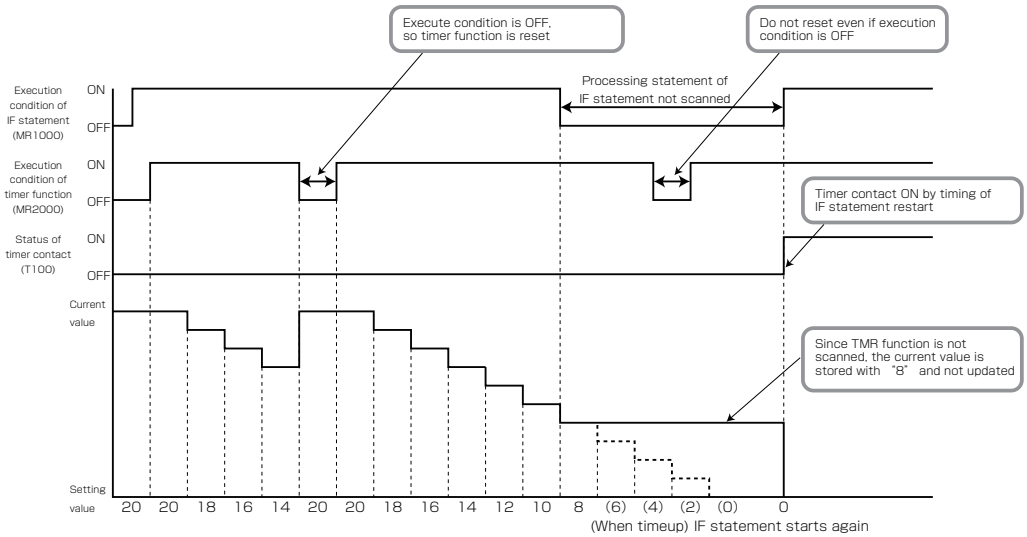
TMU (10ms Timer) : within ± (10ms+1 scanning time)

■ Precautions when using a timer function

When using timer function in IF statement or Box Script, if IF statement or Box Script is not executed during timer function execution, the current value of timer will be not updated.

If time measurement is continued, and every instruction executed, the elapsed time will be reflected in current value of timer . If reaching setting time, timer contact will be ON rapidly after execution again.

```
Programmin IF MR1000 = ON THEN
              TMR (MR2000 = ON, T100, 200)
           END IF
```



■ How to process setting/current value of the timer

The method of reading via script or changing setting value/current value of timer in ladder as follows.

		How to Read	How to Change
Current value		Storage DM = timer device Example) DM1000.D＝T100.D	Timer device=changed value (constant, device) Example) T100.D＝DM2000.D
Setting value	Device specifying	Storage DM= setting value storage device	Setting value storage device=changed value (constant, device)
	Const	The set value cannot be read.	Setting values can't be changed.

Point

When correcting, if current value exceeds setting value, the current value is corrected to the value identical to the setting value.

Counter (contact) C

Overview

Counter including contact (1 bit) and current value (32 bit).
Counter is a device in which contact is ON when current value is the same as setting value (counting is finished).

Denotation

Device No. composition as follows.
Example)

Device No. (0000 to 3999*)

C0551.B

Device name　　Suffix

Reference

- Input allows zero suppression when programming.
  Example) C551.B
- Suffix can be added when programming When used as timer contact, suffix.B must be attached; when processing current value of timer, suffix.D can be attached.
- When suffix is omitted, the type will be processed as .B (bit type), unless type declaration is available.

Range

C0000.B to C3999.B*

* Depending on the model, the Device No. that can be used varies.
  KV-8000/7500/7300/5500/5000/3000/1000：C0000.B to C3999.B
  KV Nano Series: C000.B to C255.B

■ How to program the counter when counter function is operated via scripts

Specifying via execution condition (counter reset) of the counter for counter function argument, counter No., setting value and count input relay enables programming the counter's operation.

Counter device No.　Setting value

CNT(R1000,C600,50,MR1

Counter function　counter execution condition　Count input relay

When count input relay OFF->ON (differential type), the current value of counter function will be incremented by "1"; when the current value reaches setting value, counter contact is ON.

Reference

The device (unsigned 32-bit data) and formula can be specified via setting value.
"Timer / Counter function", page 5-22

■ How to process setting/current value of the counter

The method of reading via script or changing setting value/current value of counter in ladder as follows.

		How to Read	How to Change
Current value		Storage DM = timer device Example) DM1000.D＝C200.D	Timer device=changed value (constant, device) Example) C200.D＝DM2000.D
Setting value	Device specifying	Storage DM= setting value storage device	Setting value storage device=changed value (constant, device)
	Const	The set value cannot be read.	Setting values can't be changed.

Point

When correcting, if current value exceeds setting value, the current value is corrected to the value identical to the setting value.

High-speed counter comparator (contact) CTC

Overview

High-speed counter compare including contact (1 bit) and setting value.
High speed counter comparator is a device that turns ON when current value of high speed counter is the same as setting value.

Point

When CTC is used as interrupt element, CTC (contact) is not ON.

Range

CTC0 to CTC7*

Reference

- Suffix can be added when programming When processed as contact, suffix.B must be attached; when processing setting value, suffix.D must be attached.
- When suffix is omitted, the type will be processed as .B (bit type), unless type declaration is available.

Description

Interrupt program can be also executed with CTC as interrupt condition.

Default

Under default status, the value is held.

- When the compatible model (CPU unit or base unit to use) is the KV-5500/5000/3000 or KV Nano, power off hold function in CPU system setting to set clear status (non-hold).
- When CPU unit is KV-1000, can use MEMSW instruction allows to set clear status (non-hold).

* The device numbers that can be used differ by model.

KV-5500/5000/3000/1000	: CTC0 to CTC3
KV-N14 □□ /KV-N24 □□	: CTC0 to CTC3
KV-N40 □□	: CTC0 to CTC5
KV-N60 □□	: CTC0 to CTC7
KV-NC32T	: CTC0 to CTC5

KV-5500/5000/3000 series PLC User's Manual, "2-8 High-speed Counter"
KV Nano Series User's Manual "4-4 High-speed Counter"

Control relay (contact) CR

Overview

Control relay is an internal auxiliary relay to control PLC function and acquire PLC status.

Range

CR0000 to CR7915

Reference

- Suffix can be added when programming
  Example) CR2002.B
- When suffix is omitted, the type will be processed as .B (bit type), unless type declaration is available.

Description

"CR/CM List", page A-2

* The device numbers that can be used differ by model.

KV-8000	: CR0000 to CR7915
KV-7500/7300	: CR0000 to CR7915
KV-5500/5000/3000/1000	: CR0000 to CR3915

KV Nano Series: CR0000 to CR8915

Word device

Data memory/Extended data memory DM/EM/FM

Overview

Data memory and extended data memory are used to store lookup data or 16-bit/point operation result during function operation.
When CPU unit is KV-5500/5000/3000, FM is used as File Register.
"File register ZF/FM", page 3-37

Point

EM and FM cannot be used when the compatible model (base unit to use) is the KV Nano Series.

Denotation

Device No. composition as follows.

Example)

Device name Device No.

DM06143

Reference

- Input allows zero suppression when programming.
  Example) DM6143
- Suffix can be added when programming
  Example) DM1000.S (signed 16 bit), DM2000.D (Unsigned 32 bit)
- When suffix is omitted, the type will be processed as .U (signed 16-bit), unless type declaration is available.
- Processed as the bit device.
  Example) DM1100.Bthe lowest bit of DM1100
- When the compatible model (CPU unit or base unit to use) is the KV-8000/7500/7300/5500/5000/3000 or KV Nano, it can be processed as the bit device whose bit position is specified.
  Example) DM1200.12the 13th bit of Device of DM1200

Range

- DM00000 to DM65534
- EM00000 to EM65534*1
- FM00000 to FM32766*1*2

*1 When the compatible model (base unit to use) is the KV Nano, these devices cannot be used.

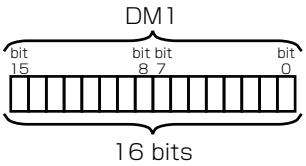
*2 When CPU unit is KV-5500/5000/3000, the range is FM00000 to FM32767.
"File register ZF/FM", page 3-37

Description

- Only Data Memory (DM) can be assigned as the unit device of expansion unit.

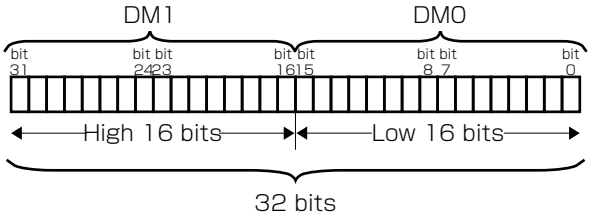
● When processing 16-bit data

- The bit width of the device is 16 bit per point. when processing unsigned DEC number, 0 to 65535 can be processed; when processing signed DEC number, -32768 to + 32767 can be processed; when processing HEX number, $0 to $FFFF can be processed.
- When processing16-bit data, suffix .U (can be omitted) or .S shall be added on Device No.



● When processing 32-bit data

- 2 devices can be used to process 32-bit data. For 32-bit data, low 16 bits will be stored in Data Memory of the specified Device No., and high 16 bits will be stored in Data Memory of the next Device No.. When processing unsigned DEC number, 0 to 4294967295 can be processed;
  when processing signed DEC number, - 2147483648 to + 2147483647 can be processed; when processing HEX number, $0 to $FFFFFFFF can be processed.
- When processing 32-bit data, suffix .D or .L shall be added on Device No..



Default

Under default status, the value is held.

- When CPU unit is KV-8000/7500/7300/5500/5000/3000 or KV Nano Series, power off hold function in CPU system setting to set clear status (non-holding).
- When CPU unit is KV-1000, MEMSW instruction allows to set clear status (non-hold).

Reference

· About the presentation of unit device for Data Memory (DM)
In KV STUDIO, except general representation, it can also be represented according to Unit No.
When unit device representation mode is used, select "View V" → "Display devices for unit (Y)" from the menu.

Unit No. :	Starting from CPU, according to the unit connection sequence, assign00, 01, 02, 03, 04...., and the like.
Unit DM No. :	Assign the occupied Data Memory from 0000.
Suffix :	When programming, Data Memory and expansion Data Memory may have suffix. When suffix is omitted, these memorizers are processed as unsigned 16-bit (.U).

Example)

Unit No.	00	01	02	03	04	05
KV-U7	KV-5000 CPU	KV-AD40 22 words	KV-DA40 18 words	KV-MC20V 402 words	KV-MC20V 402 words	KV-MC20V 402 words
Unit DM No.	0000～0019	0000～0021	0000～0017	0000～0401	0000～0401	0000～0401

The 5 th of KV-AD40 DM01_0004
The 53rd of KV-MC20V (Unit No. 03)..... DM03_0052
The 1st of KV-MC20V (Unit No. 05)..... DM05_0000

File register ZF/FM

Overview

When CPU unit is KV-8000/7500/7300/5500/5000/3000, it can be used.
File Register is divided into ZF processing all areas with consecutive number method and FM dividing the area into multiple (16 or 4) Banks and processing in Bank switching mode.
Refer to "File Register" in the User's Manual of the CPU unit used for details.

Point

When CPU unit is KV-1000, only FM (FM00000 to FM32766) can be processed.
"Data memory/Extended data memory DM/EM/FM", page 3-35
ZF/FM cannot be used when the compatible model (base unit to use) is the KVNano Series.

Denotation

Device No. composition as follows.
Example)

Device name Device No.
ZF0372

Reference

- Input allows zero suppression when programming.
  Example) ZF372
- Suffix can be added when programming.
  Example) ZF1000.S (signed 16-bit), FM12300.D (Unsigned 32-bit)
- When suffix is omitted, the type will be processed as .U (signed 16-bit), unless type declaration is available.
- Processed as the bit device.
  Example) FM1100.B.....the lowest bit of FM1100
- When CPU unit is KV-8000/7500/7300/5500/5000/3000, can be processed as the bit device whose bit position is specified.
  Example) ZF1200.12.....the 13th bit of ZF1200

Point

- Local device for ZF (@ZFxxx) cannot be used.
- Local device for FM (@FM0 to @FM32767) cannot be bank switched.
- Local device for FM (@FM0 to @FM32767) uses work memories.

Range

- ZF000000 to ZF524287*
- FM00000 to FM32767*

* The device numbers that can be used differ by model.

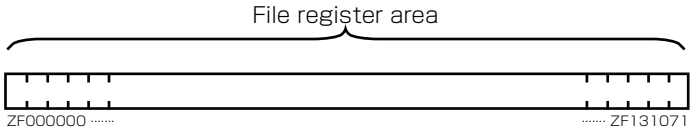
KV-8000	: ZF000000 to ZF524287, FM00000 to FM32767
KV-7500/7300	: ZF000000 to ZF524287, FM00000 to FM32767
KV-5500/5000/3000	: ZF000000 to ZF131071, FM00000 to FM32767
KV-1000	: FM00000 to FM32766
KV Nanoseries	: Cannot be used

"Data memory/Extended data memory DM/EM/FM", page 3-35

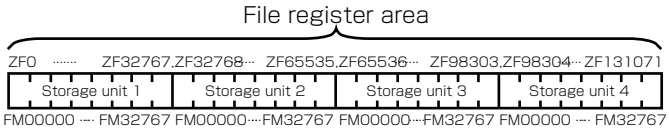
Description

- These relays cannot be assigned as the unit device of expansion unit.
- Device types and range are different between consecutive number method and Bank switching mode.

● ZF: when file register is processed with consecutive number method
Process whole area of File Register as 1 device.
When CPU unit is KV-1000 , it cannot be used.



● FM: when File Register is processed in Bank switching mode
File Register area will be separated into 16* blocks, processing is performed by switching banks.
When the corresponding model (using CPU unit) is KV-1000, only Bank1 FM0 to FM32766 can be used.

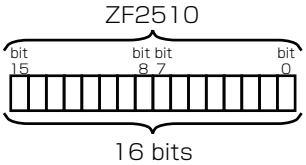


1 device can be used as 16* different devices via switching Bank.
But, it can be reused with ZF device, so, care must be taken during data processing.
For Bank switching method, please see "File Register" in the User's Manual of the CPU unit used for details.
* The number of banks that can be divided depends on the type of device.

KV-8000	: 16
KV-7500/7300	: 16
KV-5500/5000/3000	: 4

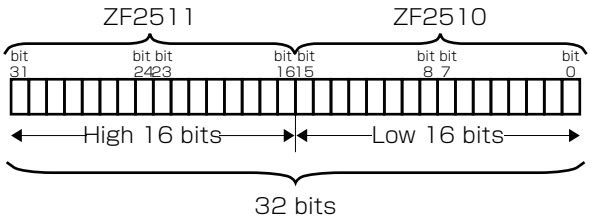
● When processing 16-bit data

- The bit width of the device is 16 bit per point, when processing unsigned DEC number, 0 to 65535 can be processed; when processing signed DEC number, -32768 to + 32767 can be processed; when processing HEX number, $0 to $FFFF can be processed.
- When processing16-bit data, suffix .U (can be omitted) or .S shall be added on Device No.



● When processing 32-bit data

- 2 devices can be used to process 32-bit data. For 32-bit data, low 16 bits will be stored in Data Memory of the specified Device No., and high 16 bits will be stored in Data Memory of the next Device No.. When processing unsigned DEC number, 0 to 4294967295 can be processed;
  when processing signed DEC number, - 2147483648 to + 2147483647 can be processed;
  when processing HEX number, $0 to $FFFFFFFF can be processed.
- When processing 32-bit data, suffix .D or .L shall be added on Device No..



Default

Under default status, the value is held.

- When CPU unit is KV-8000/7500/7300/5500/5000/3000, power off hold function in CPU system setting to set clear status (non-hold).
- When CPU unit is KV-1000, MEMSW instruction allows to set clear status (non-hold).

Link register W

Overview

This device cannot be used when the compatible model (CPU unit to use) is the KV-1000.

It is used as link register when connecting via FL-net, etc. Different from other word devices, Device No. is specified using HEX system.

Denotation

Device No. composition as follows.

Example)

Reference

- Input allows zero suppression when programming.
  Example) W7A
- Suffix can be added when programming.
  Example) W7A.S (signed 16 bit), W3E76.D (Unsigned 32 bit)
- When suffix is omitted, the type will be processed as .U (unsigned 16-bit), unless type declaration is available.
- Processed as the bit device. bit position can be also specified.
  Example) the 13th bit of W1200.12W1200

Point

- **Please use HEX number (0 to F) to specify.**
- **Local device (@) cannot be used.**

Range

W0000 to W7FFF*

* The device number that can be used depends on the type of device.

KV-8000	: W0000 to W7FFF
KV-7500/7300	: W0000 to W7FFF
KV-5500/5000/3000 Series	: W0000 to W3FFF
KV Nano Series	: W0000 to W3FFF

Usage

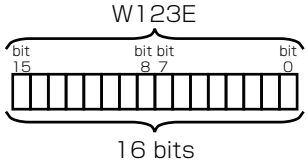
It is used as link relay when connecting via FL-net, etc.

Description

These relays cannot be assigned as the unit device of expansion unit.

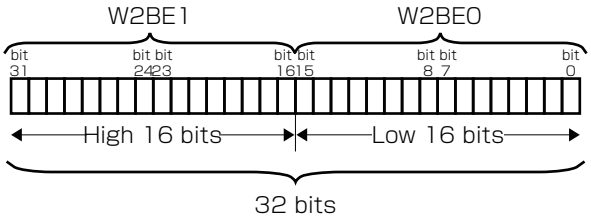
● When processing 16-bit data

- The bit width of the device is 16 bit per point, when processing unsigned DEC number, 0 to 65535 can be processed; when processing signed DEC number, -32768 to + 32767 can be processed; when processing HEX number, $0 to $FFFF can be processed.
- When processing 16-bit data, suffix .U(can be omitted) or .S shall be added on Device No.



● When processing 32-bit data

- 2 devices can be used to process 32-bit data. For 32-bit data, low 16 bits will be stored in Data Memory of the specified Device No., and high 16 bits will be stored in Data Memory of the next Device No.. When processing unsigned DEC number, 0 to 4294967295 can be processed; when processing signed DEC number, - 2147483648 to + 2147483647 can be processed; when processing HEX number, $0 to $FFFFFFFF can be processed.
- When processing of 32-bit data, suffix .D or .L shall be added on Device No..



Default

Under default status, the value is held.
Hold mode setting can be performed utilizing “Power off hold function" in CPU system setting.

Temporary data memory TM

Overview

Temporary data memory is a 16-bit word device that can conduct same processing with Data Memory. Some device areas are also used to temporarily store arithmetic operation data or arithmetic operation result data.

Denotation

Device No. composition as follows.

Example)

Device name Device No.

TM010

Reference

- TM000 to TM003 are used in operation instructions such as DIV, MUL in ladder, TM000 and TM001 used for processing 32-bit above data. It is recommended that these temporary data memories are not used for other purposes.
- Input allows zero suppression when programming.
  Example) TM10
- Suffix can be added when programming
  Example) TM10.D (Unsigned 32-bit)
- When suffix is omitted, if without type declaration, it is processed as .U (Unsigned 16- bit).

Range

TM000 to TM511

TM No.	Application
TM000 to TM001	64-bit data
TM002/TM003	Used to store the remainder after dividing
TM004 to TM511	Common *

* Simple indirect specifying cannot be used for script.

Default

When powering ON or PROG -> RUN, temporary data memoryis cleared.

Timer (current value) T

Overview

By adding suffix .D to timer device, timer device can process current value.

Range

T0000.D to T3999.D*

Reference

When timer device processes current value, suffix.D cannot be omitted. When suffix is omitted, if without type declaration, it is processed as .B (timer contact).

Description

Current value of timer device can be changed within 0 to 4294967295.

Point

When current value changes, if current value exceeds setting value, current value will be changed to the value identical to setting value.

"Timer (contact) T", page 3-31

* Depending on the model, the Device No. that can be used varies.
  KV-8000/7500/7300/5500/5000/3000/1000：T0000.D to T3999.D
  KV Nano Series: T000.D to T511.D

Counter (current value) C

Overview

By adding suffix.D to counter device, counter device can process current value.

Range

C0000.D to C3999.D*

Reference

When Counter device processes current value, suffix.D cannot be omitted. When suffix is omitted, if without type declaration, it is processed as .B (counter contact).

Description

Current value of counter device can be changed within 0 to 4294967295.

Point

When current value changes, if current value exceeds setting value, current value will be changed to the value identical to setting value.

* Depending on the model, the Device No. that can be used varies.
KV-8000/7500/7300/5500/5000/3000/1000：C0000.D to C3999.D
KV Nano Series: C000.D to C255.D

"Counter (contact) C", page 3-33

High-speed counter (current value) CTH

Overview

High speed counter is 32-bit counter used to collect short interval signal that cannot be collected by counter instruction.

Range

CTH0 to CTH3*

Reference

- Suffix can be added when programming
  Example) CTH0.D
- When suffix is omitted, if without type declaration, process as .B (Unsigned 32-bit).

Description

- A value within 0 to 4294967295 can be processed.
- In addition to external input counting, internal clock can also be counted.
- Maximal response frequency is duty ratio 50%, 1-phase, 100KHz, phase difference50KHz.

Default

Under default status, output is off when powering ON and when status is cleared during PPROG → RUN.

- When the compatible model (CPU unit or base unit to use) is the KV-5500/5000/3000 or KV Nano, power off hold function in CPU system setting allows to set hold mode.
- When CPU unit is KV-1000, MEMSW instruction allows to set hold mode.

* Depending on the model, the Device No. that can be used varies.
KV-5500/5000/3000/1000: CTH0 to CTH1
KV-N14**/KV-N24**: TH0 to CTH1
KV-N40**: TH0 to CTH2
KV-N60**: CTH0 to CTH3
KV-NC32T: CTH0 to CTH2

KV-5500/5000/3000 series PLC User's Manual, "2-8 High-speed Counter"
KV Nano series User's Manual, "4-4 High-speed Counter"

High-speed counter comparator (setting value) CTC

Overview

By adding suffix.D to CTC, setting value of high-speed counter (32-bit) can be processed.

Range

CTC0 to CTC7*

Reference

When high-speed counter compare device processes setting value, suffix .D cannot be omitted. When suffix is omitted, if without type declaration, it is processed as .B (contact).

Description

- The range should be 0 to 4294967295.
- 2 setting values (CTC) can be set for every high-speed counter (CTH) .
  - CTC0,CTC1 .....CTH0
  - CTC2,CTC3 .....CTH1
  - CTC4,CTC5 .....CTH2*
  - CTC6,CTC7 .....CTH3*

Default

The value is held.

* Depending on the model, the Device No. that can be used varies.
  KV-5500/5000/3000/1000: CTC0 to CTC3
  KV-N14**/KV-N24**: CTC0 to CTC3
  KV-N40**: CTC0 to CTC5
  KV-N60**: CTC0 to CTC7
  KV-NC32T: CTC0 to CTC5

KV-5500/5000/3000 series PLC User's Manual, "2-8 High-speed Counter"
KV Nano series User's Manual, "4-4 High-speed Counter"

Index register Z

Overview

32-bit word device that is used to specify No. attached to destination device during index modification. Just like data memory, it may store data and operation result.

Range

KV-8000 after CPU function version V2.0: Z1 to Z23
Models other than the above: Z1 to Z12

Reference

- Suffix can be added when programming
  Example) Z01.L (signed 32-bit)
  Z02.D (unsigned 32-bit)
- For programming when suffix omitted, if without type declaration, when the model is KV-8000/7500/7300/5500/5000/3000 or KV Nano series, it is processed as .L (signed32-bit); when the model is KV-1000, it is processed as .S (signed 16-bit).

Point

- **Z11 are reserved for system and cannot be used.**
- **.F, .DF, .T, .B cannot be specified in suffix.**
- **The bit of word device cannot be specified.**
- **Local device is not available. ZPUSH and ZPOP function should be used.**

"ZPUSH function", page 6-20
"ZPOP function", page 6-24

Default

When powering ON, or PROG-RUN, index register is cleared.

"Index modification", page 3-55

Control memory CM

Overview

Control memory is a specific word device to control PLC function and acquire PLC status.

Range

CM0000 to CM7599*

Reference

- Suffix can be added when programming.
  Example) CM1720.U
- When suffix is omitted, if without type declaration, it is processed as .U (Unsigned 16-bit).

* Depending on the model, the Device No. that can be used varies.

KV-8000	CM0000 to CM7599
KV-7500/7300/5500/5000/3000	CM0000 to CM5999
KV-1000	CM00000 to CM11998
KV Nano Series	CM0000 to CM8999

Description

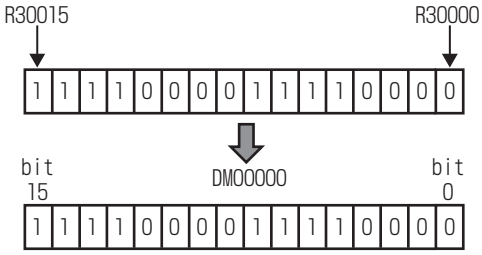
"CR/CM List", page A-2

Processing Bit Device in Word Unit

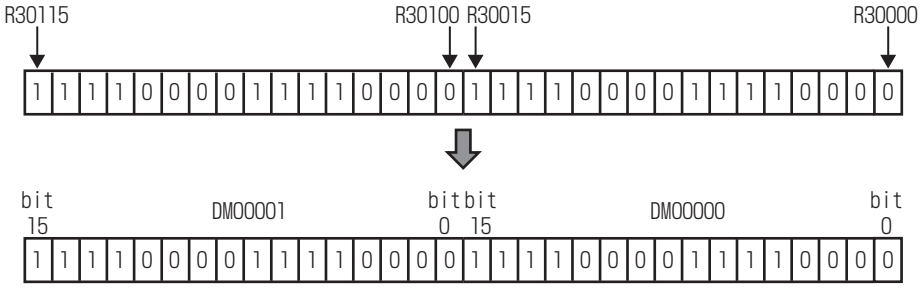
If the bit device without suffix.B is specified according to functions and assignment statements of word device used, it will be processed in word unit starting from the specified bit device.

Example)

DM0000.U = R3000.U

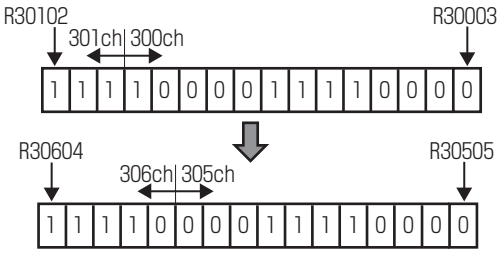


DM0000.D = R3000.D



If the bit device with suffix is not at the leading of a channel, straddle the next channel and process in word unit.

R30505.U = R30003.U



Processing Word Device as Bit Device

In general, word device is processed as the device of 16-bit/32-bit, but it can be processed as bit device after adding suffix of bit type.

Point

High-speed counter (CTH) and index register (Z) cannot be processed as the bit device.

● When processing as the bit device at the lowe rest bit

When adding suffix .B to word device, or specifying word device in arguments and assignment statements of contact output function used for processing bit devices etc., it will be processed as bit device at the lowe rest bit of word device.

When DM1000 value is #46563 ($B5E3)

DM1000=#46563($B5E3)

1	0	1	1	0	1	0	1	1	1	1	0	0	0	1	1

Lowest bit
1: Contact ON
0: Contact OFF

The lowest bit of DM1000 is 1, so, DM1000.B shall be processed as per "ON".

● When processing as the bit device at any bit.

This device cannot be used when the compatible model (CPU unit or base unit to use) is the KV-1000.
When bit position of ".0 to .15" is specified in word device, it is processed as bit device at specified bit position of word device.

Denotation

Example)

DM1234.12

Random word device Bit position (.0 to .15)

Reference

- Same bits are processed by DM123.B and DM123.0.
- Device comment displays comment set against word device.

Point

- **When CPU unit is KV-1000 , it cannot be used.**
- **Please use DEC (0 to 15) to specify the bit position.**
- **Local device (@) cannot be used.**
- **The devices whose bit can be specified are only limited to DM, W, EM, FM, ZF, TM and CM.**
- **The variables that can be specified by the execution bit are limited to the variables whose data format is "INT, DINT, UINT, UDINT" and the elements of the array type variables and structure type variables set to these data formats.**
- **For label, bit position cannot be specified. (during global label registration, the word device whose bit has been specified can be specified)**
- **For macro argument device, bit cannot be processed based on the position of the bit specified.**
- **Macro function argument cannot be set for the device whose bit position specified.**
- **Indirect specifying and index modification cannot be performed for the device whose bit position specified.**

When DM123 value is #59 ($3B)

bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1

When DM123 is treated as bit device at the specified bit position, bit status as follows.

Device	Bit status
DM123.0	ON
DM123.1	ON
DM123.2	OFF
DM123.3	ON
DM123.4	ON
DM123.5	ON
DM123.6 ~ DM123.15	OFF

Direct I/O Relay

When the compatible model (CPU unit or base unit to use) is the KV-8000/7500/7300/5500/5000/3000 or KV Nano, you can add "D" to the front end of device No. of I/O relay (R), and it works as direct I/O relay.

Denotation

Example)

Direct I/O relay

DR001

I/O relay

Point

- **When the corresponding model (using a CPU unit) is KV-1000, direct I/O relays cannot be used.**
- **When the corresponding model (basic unit used) is the KV Nano series, the direct input / output relay cannot be used for the expansion unit.**
- **No suffix other than .B could be added.**

Range

I/O relay

KV-5500/5000/3000	DR000 to DR015/	DR500 to DR503/ Expansion I/O relay
KV-N14 □□	DR000 to DR007/	DR500 to DR505
KV-N24 □□	DR000 to DR013/	DR500 to DR509
KV-N40 □□	DR000 to DR107/	DR500 to DR515
KV-N60 □□	DR000 to DR203/	DR500 to DR607
KV-NC32T	DR000 to DR015/	DR500 to DR515

For direct I/O relays, see 📖 "KV-5500/5000/3000 series User's Manual" or "KV Nano series User's Manual".

Variables

The current values of all devices and variables in a script can be monitored simply by selecting the script in the monitor window. KV STUDIO can use variables for programming in addition to devices.
Unlike devices, variables can be programmed with names defined by KV STUDIO, and the defined variables are automatically assigned to the work area of the CPU unit. Therefore, it is not necessary to create a device map, etc. to manage the device numbers assigned with functions. In addition, variables are also applicable to the VT5 series and Soft-VT, so there is no need to change the device number on the display unit side in combination with the device number on the PLC side.

Point

- **Variables are only available in KV STUDIO Ver11.0 and later. The corresponding models are only KV-8000 with CPU function version 2.0 and later.**
- **Variables that can be read and written from the display unit are limited to VT STUDIO Ver. 8.0 and later. The corresponding model is limited to the VT5 series and the PLC model is KV-8000 (Ethernet).**

For the setting method of variables, please refer to "Label Programming and Label Array", page 2-25.

Variables and data formats at a glance

When using variables, it is necessary to define the data format in advance. The CPU unit automatically allocates the necessary data size to the work area*1 according to the defined data format.
The work area sizes and variable specifications that can be used with KV-8000 are as follows.

Item		Details
Work area size	bit	64000 points
	word	589824 words
	Timer	4000 points
	Counter	4000 points
Variable specifications	Definable number	100000
	Maximum length of variable name	128 character

*1 You can manually assign devices to the CPU unit without using the work area. Please refer to "KV-8000 Series User Manual" for details.

Point

The start of BOOL type arrays may not be assigned to the start of the channel. When specifying a BOOL type array for an operand that must specify the start of the channel, specify this array assigned to the start of the channel with one of the following methods.
- **Assign the array to a device of the CPU unit.**
- **Create a structure with a BOOL type array member and assign this member to the start of the channel.**

The data format and data size of the variables that can be used with KV-8000 are as follows.

Data type		Data size (bit units)
Type name	Description	
BOOL	bit	1
UINT	1 word unsigned integer	16
INT	1-word signed integer	16
UDINT	2 word unsigned integer	32
DINT	2-word signed integer	32
REAL	Single precision floating point type	32
LREAL	Double precision floating point type	64
STRING	Text string	Round (number of characters * 2 + 1) × 8 to an integer multiple of 16
Timer	Timer	-
Counter	Counter	-
Array	-	Data format × array size
Structure	-	Based on definition*1
Function block	Function block	Based on definition*1

*1 Structures and function blocks are data formats that use multiple data formats. Therefore, the data size is determined according to the setting contents. Please refer to "KV-8000 Series User Manual" for details.

Constant

Constant and its range will vary with function used and suffixof device.

(#) DEC constant

Overview

DEC constant is expressed by 16-bit and 32-bit DEC.

Denotation

DEC value is expressed as it is. (signed representation also acceptable)

Example)

Value

06143

Reference

- Input allows zero suppression when programming.
- DEC constant can also be attached prefix "#" when programming. For signed DEC constant, only "+" can be omitted.
- In XYM display mode,"K" can be added to indicate.

Range

- Unsigned 16-bit: 0 to 65535
- Unsigned 32-bit: 0 to 4294967295
- Signed 16-bit: - 32768 to +32767
- Signed 32-bit: - 2147483648 to +2147483647

Reference

- Adding suffix via constant and programming, can expand the range of constant. (suffix is not added to constant in programming generally)

300U : 16-bit, unsigned (0 to 65535)
500S : 16-bit, signed (- 32768 to +32767)
3000D : 32-bit, unsigned (0 to 4294967295)
5000L : 32-bit, signed (- 2147483648 to +2147483647)
123F : Single precision floating point type
(-3.4E+38 ≦ N ≦ 1.4E-45
N = 0
1.4E-45 ≦ N ≦ 3.4E+38)
1.2345678DF (Number of effective digits: approx. 7 digits)
: Double precision floating point type
(-1.79E+308 ≦ N ≦ -2.23E-308
N = 0
2.23E-308 ≦ N ≦ 1.79E+308)
(Number of effective digits: approx. 16 digits)

Point

- **For constant suffix, "." (point) is not added when programming.**
- **For the KV Nano Series, an operation error occurs when using denormalized single precision floating point types.**

$ HEX constant

Overview

HEX constant is expressed by 16-bit and 32-bit HEX number.

Denotation

HEX constant takes "$" as prefix, attached with HEX value.

Example)

HEX symbol Value

$1AF7

Reference

- **Input allows zero suppression when programming.**
- **In XYM display mode, "K" can be added to indicate.**

Range

- 16 bit : $0 to $FFFF
- 32 bit : $0 to $FFFFFFFF

Reference

Due to HEX number without suffix, 16-bit/32-bit will be identified via the number of digits after"$".

$00FF(4 digits or less) ……16-bit

$0000 00FF (5 digits or more)……32-bit

Point

- **When programming, if HEX constant has no prefix "$" and contains A to F , error will occur; if it does not contain A to F, it will be considered as DEC number.**
- **When programming, sign (+/-) cannot be attached to HEX constant.**

Single precision floating point type constant

Overview

When programming, single precision floating point type real number can act as floating real number.

Denotation

- DEC constant is expressed by [sign + DEC number].

Example)

Value

1234.56

- 8-bit above is represented in exponential form.
- DEC constant is represented by suffix.

Example)

Value single precision floating point type

1234F

Reference

For floating type constant, suffix can be omitted.

Example)

1234.0

Indicates single precision floating point type

Reference

DEC constant can also contain prefix "#" when programming. For signed DEC constant, only "+" can be omitted.

Range

-3.4E+38 ≦ N ≦ -1.4E-45

N=0

+1.4E-45 ≦ N ≦ +3.4E+38 (Number of effective digits: approx. 7 digits)

(For the KV Nano Series, an absolute value in a range of 1.4E-45 to 1.2E-38 cannot be used.)

Double precision floating point type constant

Overview

When programming, double precision floating point type real number can act as floating real number.

Denotation

• DEC constant is expressed by [sign + DEC number].

Example)

Value

12345.6789

• 17-bit above is represented in exponential form.
• DEC constant is represented by suffix.

Example)

Value double precision floating point type

1234DF

Reference

DEC constant can also contain prefix "#" when programming. For signed DEC constant, only "+" can be omitted.

Range

-1.79E+308 ≦ N ≦ -2.23E-308
N=0
+2.23E-308 ≦ N ≦ +1.79E+308 (Number of effective digits: approx. 16 digits)

Fixed text string (text string type constant)

Overview

During programming, any text string can be bracketed using double quote (").

Denotation

("") indicates text string.

Example)

String

"Operation result"

Reference

• When processing (") as characters, please input (" ").
•When adding (") to other characters, use "+" and characters.

DM1000.T = " " + "Operation result" + " " "Operation result" is stored in DM1000.T.

Point

Text string will be processed as label if ("") not inputted.

Range

Up to 1999 characters can be processed.

Index Modification and Indirect Specifying

Devices that can be used with index modification and indirect specifying are listed below.

Device			KV-8000/7500/7300/5500/5000/3000		KV-1000		KV Nano	
Type	Symbols	Name	Index Modification	Indirect Specifying	Index Modification	Indirect Specifying	Index Modification	Indirect Specifying
bit Device	R	Relay	○	○*	○	○*	○	○*
	CR	Control relay	—	—	—	—	—	—
	T	Timer (contact)	○	○*	○	○*	○	○*
	C	Counter (contact)	○	○*	○	○*	○	○*
	MR	Internal auxiliary relay	○	○*	○	○*	○	○*
	LR	Latch relay	○	○*	○	○*	○	○*
	B	Link relay	○	○*	—	—	○	○*
Word Device	DM	Data memory	○	○	○	○	○	○
	EM	Expansion data memory	○	○	○	○	—	—
	FM	File register (Bank switching)	○	○	○	○	—	—
	ZF	File register (Consecutive numbering)	○	○	—	—	—	—
	W	Link register	○	○	—	—	○	○
	TM	Temporary memory	○	○	○	○	○	○
	TS	Timer (setting value)	○	○	○	○	○	○
	TC	Timer (current value)	○	○	○	○	○	○
	CS	Counter (setting value)	○	○	○	○	○	○
	CC	Counter (current value)	○	○	○	○	○	○
	CM	Control memory	—	—	—	—	○	○*

* Indirect specifying is possible, but the address cannot be stored with the "*(device)" description.

Point

- **Index modification is not possible for CR, CTC, CTH, Z, labels (label arrays), variable (variable array), and bit specifying devices of word devices.**
- **Indirect specifying is not possible for CR, CTC, CTH, Z, and bit specifying devices of word devices.**

Index modification

"Index modification" refers to the method for specifying numbered device by adding the value and constant of index register (Z) to corresponding device.
The range of the value that can be summarized is signed 32-bit value (-2147483648 to +2147483647).
The devices that can be used differ by unit. See "Devices that can be used with index modification and indirect specifying" on page 第 3-54 页.
Index modification can be used together with indirect specifying (*) or local device (@).

Point

- **Bits (label arrays) and word device bit designation devices cannot be index modified.**
- **Index modification cannot be performed for variables. However, index modification can be performed for variables manually assigned to global devices.**
- **When specify the device out of range that is capable of index modification according to argument, control relay CR2012 is ON and this function is not executed. But operation continues.**

■Modify by index register

Index modification is Used to change device no. in KV-1000 operation.
Programming method to modify device through index register is as follows:

(Device No.) : Z(01 to 10)

Example)

DM1000:Z01 — Specify DM0990 when value in Z01 is -10

R0000:Z05 — Specify R0100 when value in Z05 is +16.

Point

Versions after Z11 are reserved by the system and cannot be used.

Exampl

The current value stored in DM1000 will be saved as triggering signal at the rising edge of R0001.
Index register will be initialized during start of operation; the data is stored in memories starting from DM0000 at the rising edge of R0001(OFF → ON change).

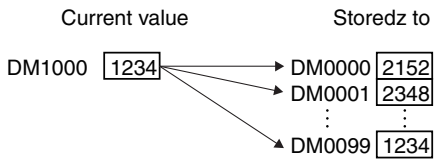
Scripting

```
IF LDP(CR2008) THEN
    Z01=0
END IF
IF LDP(R0001) THEN
    DM0000:Z01 = DM1000
    INC(Z01)
END IF
```

Set index register Z01 to 0 when starting operation.

Write value of DM1000 to destination device to be index-modified at rising edge of R0001.

Increment (+1) index register Z01



Point

Since the device (value) to be index-modified is processed as signed 32-bit data, the index register whose type is not declared will be automatically performed as".L (signed 32-bit)".

Reference

If 10 or more index registers (Z) are required in a project, index register function (ZPUSH/ZPOP) can be used for batch cleaning or restoring index register of every module or macro.

"ZPUSH function", page 6-20
"ZPOP function", page 6-24

■Modify via index constant

When constant is used for index modification

(Device No.): (#) (DEC constant)

("#" can be omitted.)

Example)

DM1000:10 — DM1000+10 words -> DM1010

R0000:16 — R0000+16 bits (1 channel) -> R0100.

Reference

If it is used when making ladder using unit device, the device No. assigned can be clearly identified. Besides, when making macro as argument device, it is valid when UR/UM is used.

Example)

UR0:2

UM1:32

■Modify via the device and expression

Script allows to perform index modification for the device other than index register.

In this case, index register (Z11/Z12) automatically reserved for system during ladder conversion is used .

Scripting

DM1000 : (DM10 + 2) = 100

Ladder

*during index modification via expression, must add bracket () to the expression.

■Modify the device with suffix

The suffix of the device for index modification can be programmed as follows:

DM1000.D:Z01.S

DM1100.F:(DM10+10)

■Invalid index modification

Index modification cannot be performed in the following cases:

- (DM1000: Z01): Z02 To modify the device after index modification
- (DM1000+3): Z02. To modify operation result
- DM1000: Z01+5 "( )" is not included index modification expression
- Z01: Z02 •••••••••••••••••••••• To modify the device that the index modification cannot be performed

What is indirect specifying

■What is indirect specifying(∗)

Indirect specifying (∗) refers to indirect specifying by programming the device storage device indicated value (address) instead of direct device specifying.

● Indirect specifying principle

Every device is managed by internal memory in CPU unit, each with separate address.

- Indirect specifying (∗) operates the address via special function, changing the device to be searched.
- Index modification is represented via [base address device + relative offset (index constant)], and indirect specifying represented via [“∗”+ specifying destination address storage device].
- Indirect specifying can be used with index modification (: Z /: (#) decimal constant) and local device (@).
  2 words must be used for address storage due to address managed as 32-bit.

Point

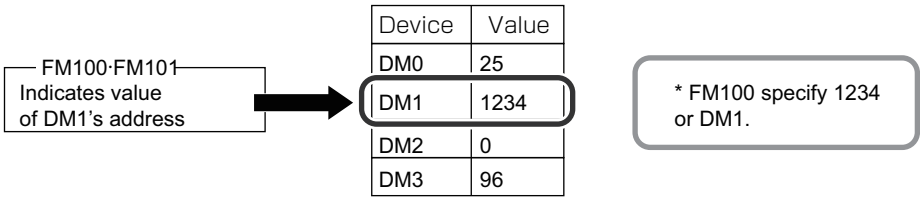
- **If the device outside range of applicability is addressed via indirect specifying, the action of function cannot be guaranteed.**
- **If the device outside range of applicability is addressed according to the condition operation formula, the operation result cannot be guaranteed.**

● Indirect specifying programming

If indirect specifying is programmed with "∗(device)" the device value or the device matching the address stored in the device will be searched.

Example)

When the address of DM1 is set to FM100 and programmed as ∗FM100



The devices that can be used differ by unit. For details, see “Devices that can be used with index modification and indirect specifying.”

Point

- **Variables and labels can also be specified indirectly. However, the following cannot be specified indirectly.**
  - **Subscripts for array variables**
  - **Elements of array variables**
  - **Member of structure type variable**
- **Word data bit position specification cannot be performed indirectly.**

Reference

- If indirect specifying device is a timer (T) or counter (C), contact, current value and setting value will depend on the function argument used and additional suffix.
- Use programming format of local device "∗@(device) ".
- Indirect specifying device is processed as .U (unsigned 16 bit).
- Type declaration can be also conducted for indirect specifying.

● Application method of indirect specifying

Indirectly specify according to the following steps.

1 Write the address of device to be searched with ADRSET function.

Example)

ADRSET(R1000,DM0,FM100.D)

Store address of DM0 to FM100/FM101 when execution condition (R1000) is ON.

Point

The devices that can be used differ by unit. For details, see "Devices that can be used with index modification and indirect specifying."

"ADRSET function", page 6-26

2 For every argument (device) of indirect specifying function, Prefix "∗" can be added to the device for storing current address.

Example)

EM1000.U = AVG(∗FM100.U,10)

∗Store average vaue of 10-word data starting from FM100 (DM0) to EM1000.U.

Point

Before adding prefix "∗" to address storage device, ADRSET function must be used to store the address of device to be indirectly addressed.

3 If changing the device to be searched, indirect specifying function (ADRINC/ADRDEC/ADRADD/ADRSUB) can be used to change the address stored currently.

Example)

ADRSET(EM300,FM100.D) — Set EM300's address to FM100.

ADRINC(R1001,FM100.D) — Increment (+1) DM0 address stored in FM100/FM101 when execution condition (R1001) is ON

∗FM100 = DM1000 — ∗ Store value of DM1000 in FM100 (EM300).

Point

When changing the address to be searched, do not use common operators(+, -,∗, /, etc.). Otherwise, Ladder fails to operate correctly.

● Type of indirect specifying device

For indirect specifying device (*[device]), similar to other device, data range that can be processed varies with suffix.

- The device type can be represented by adding suffix to indirect specifying device.

```
Scripting  *DM1000.F = 123.567      ' *DM1000 indirect specifying device is processed as .F
                                    ' (single precision floating point type).
```

- If using type declaration (TYPE) to specify the type of indirect specifying device, the suffix can be omitted when programming.

```
Scripting  TYPE *DM1000.F           ' Type declaration: *DM1000 is processed as .F.
           *DM1000 = 123.567        ' Even if the suffix is omitted, the device of *DM1000
                                    ' indirect specifying is also processed as .F.
```

- When programming, if the suffix of indirect specifying device without type declaration is omitted, the device will be processed as .U.

```
Scripting  *DM1000 = 65535          ' *DM1000 indirect specifying device without type
                                    ' declaration is processed as .U (unsigned 16 bit).
```

In this case, even if the device type of indirect specifying is not.U, if suffix of indirect specifying device is omitted, can be also processed as.U.

```
Scripting  ADRSET (MR2000, DM1000.D) ' MR2000 address is stored in DM1000.D.
           *DM1000 = 123             ' MR2000 is bit device but suffix is omitted, *DM1000
                                     ' (=MR2000) is processed as. U ( word device).
```

- If the type of indirect specifying device has been declared, the type of address storage device is declared as.D (unsigned 32-bit).

```
Scripting  TYPE *DM1000.F           ' *DM1000 is declared as .F, and DM1000 as .D.
           TYPE *DM2000.S           ' *DM2000 acts as .S declaration, or DM2000 as .D
                                    ' declaration.
           ADRSET (EM5000, DM1000)  ' DM1000 is processed as .D .
           *DM1000 = 123.567        ' *DM1000(=EM5000) is processed as .F .
           ADRSET (FM6000, DM2000)  ' DM2000 is processed as .D .
           *DM2000 = -555           ' *DM2000(=FM6000) is processed as.S .
                                    '
```

● Example for indirect specifying

During initial scanning, DM0 address will be stored to FM100/FM101.

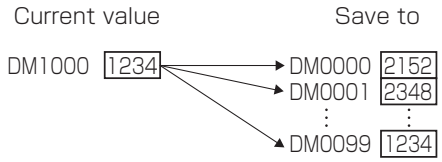
DM1000 current value is stored from DM0 in turn and triggered at rising edge of R0001.

```
ADRSET(LDP(CR2008),DM0,FM100)
IF LDP(R1) THEN
    *FM100 = DM1000
    ADRINC(FM100)
END IF
```

Write address of DM0 to [FM100/FM101] when starting operation.

Write value of DM1000 to index-specified destination device at the rising edge of R0001.

Increment (+1) address stored in [FM100/FM101].



Combination of index modification and indirect specifying

■How To Program

During index modification for indirect specifying device, you can program according to following method.

Example) (∗DM1000.U):Z01 — ∗Index-modify DM1000 by Z01.

∗(DM1000.U:Z01) — Indirect-specify DM1000 by Z01.

When programming with ( ) is omitted, the meaning is as follows:

Example) ∗DM1000.U:Z01 — ∗Index-modify DM1000 by Z01.

● Example for combination of index modification and indirect specifying

Taking rising edge of R1 as trigger signal, store current value of DM1000.

If R2 is ON at rising edge of R0, the data will be stored in turn from DM0; if it is OFF, the data will be stored in turn from EM0.

```
IF LDP(R0000) THEN
    Z01 = 0 ----------------------------------------①
    ADRSET(R2,DM0,FM100.D) ------------------②
    ADRSET(R2 = OFF,EM0,FM100.D) ----------③
END IF
IF LDP(R1) THEN
    ∗FM100:Z01 = DM1000 ----------------------④
    Z01 += 1----------------------------------------⑤
END IF
```

[When starting measurement]

① Write 0 in Index register Z01. (Index register initialization)

② If R2 is ON, write address of FM100/FM101 to DM0.

③ If R2 is OFF, write address of FM100/FM101 to EM0.

[When trigger occurs]

(4) Write FM100/FM101 value into device obtained via index modification Z01 (Z01 added), address device is stored in DM1000.

Example) when R2 is OFF,

∗FM100→EM0 → ∗FM100:Z01→EM23

Z01= 23

(5) Z01 increment "+ 1".

■Simple indirect specifying

KV script cannot use simple indirect specifying (#TM). When using simple indirect specifying, ladder program must be used.

Data Type (suffix)

For KV script, device type and constant type can be represented by adding suffix to the device.
For the device with the suffix ".U", ".S", ".D", ".L", ".F", ".DF", ".B" or ".T", the type of processing data varies with different suffixes.

Suffix

A suffix refers to a symbol attached to the end of the device, and can be divided into seven kinds as follows.

□.U　□.S　□.D　□.L　□.F　□.DF　□.B　□.T

*□The device with corresponding suffix is input in .

According to suffix, every device can process Data Memory value, etc. as follows:
Suffix □.U is processed as unsigned 16-bit data. (0 to 65535)
Suffix □.S is processed as signed 16-bit data. (-32768 to 32767)
Suffix □.D is processed as unsigned 32-bit data. (0 to 4294967295)
Suffix □.L is processed as signed 32-bit data. (-2147483648 to 2147483647)
Suffix □.F is processed as single precision floating point type real number data.

−3.4E+38 ≦ N ≦ −1.4E−45
N = 0
1.4E−45 ≦ N ≦ 3.4E+38
(Number of effective digits: approx. 7 digits)
(For the KV Nano Series, an absolute value in a range of 1.4E-45 to 1.2E-38 cannot be used.)

Suffix □.DF is processed as double precision floating point type real number data.

−1.79E+308 ≦ N ≦ −2.23E−308
N = 0
2.23E−308 ≦ N ≦ 1.79E+308
(Number of effective digits: approx. 16 digits)

Suffix □.B is processed as bit data.（ON:TRUE，OFF:FALSE）
Suffix □.T is processed as text string data.

Reference
The suffix specified by ladder program can be only added at the end of an instruction, not device (for script, suffix is only added at the end of the device, not function).

例

```
DM1000 (.U)
DM1000.S
DM1000.D
DM1000.L
DM1000.F
DM1000.DF
DM1000.B
DM1000.T
```

Point
When using variables and labels, you can also use the suffix like "AnalogData.L". However, if the data size of the suffix processing is larger than the data format set in the variable edit window / label edit window, a conversion error will occur.

● Suffix omitted

For programming of the device whose suffix omitted, the processing type depends on the device type.

Device type	Type processed when suffix is omitted (default)
R(DR),B,MR,LR,T,C,CTC,CR,UR	.B (bit type)
DM,W,EM,ZF,FM,TM,CM,V0,P0,UM0,UV0 indirect addressing (∗) device	.U (unsigned 16-bit data type)
Z*	KV-8000/7500/7300/5500/5000/3000 : .L (signed 32-bit data type) KV-1000 : .S (signed 16-bit data type)
CTH	.D (unsigned 32-bit data type)

∗ Default type varies with different CPU units.

Point **When the suffix is omitted for a variable, it is processed according to the data format set in the variable edit window.**

Programming Script

```
DM1000 = 2000                    'When omitted, data is processed as unsigned
                                 16-bit (.U).
DM1100.D = 75535                 'DM1100 is processed as unsigned 32-bit (.D).
IF DM1000 = 3000 THEN            '
   DM3000.L = DM1100 + 1000      'Right side is operated as signed 32-bit.
END IF                           '
                                 '
DM1000.L = −21474800             'DM1000 is processed as signed 32-bit (.L)
IF DM1000 > 0 THEN               '
   DM3000.D = DM1000.S ∗ −1      'Right side is developed to .L operation,
END IF                           converted
                                 to .D and then replaced.*
```

∗ The signed 32 bit data (.L) will be replaced with unsigned 32-bit data (.D). If "Set script warning level" is set too higher, the warning will be sent during script conversion.

Reference The type of device can also be declared in advance via "TYPE" programming.
Type declaration is valid only in type statement script.
For details, see "About Type Declaration (TYPE)".

■ About .U and .S

Suffix □.U processes BIN data as unsigned 16-bit data.
Suffix □.S processes BIN data as signed 16-bit data.
Correspondingly, the decimal representation relation between BIN data and corresponding suffix is shown below.

Device (DM1000)	Constant type	1111111111111111 and bit data
DM1000.U	Unsigned DEC	65535
DM1000.S	Signed DEC	-1

- When 16-bit BIN data (1111111111111111) stored in DM1000 is compared with decimal constant 100, if suffix is .U, the value of DM1000 is compiled to 65535.

Programming Script
```
IF DM1000.U > 100 THEN
INC (DM2000)
END IF
```

Correspondingly, if DM1000 > 100 is TRUE, add "1" to DM2000

- If suffix is .S, DM1000 value is compiled to "-1".

Programming Script
```
IF DM1000.S > 100 THEN
DEC (DM2000)
END IF
```

Correspondingly, subtraction is not conducted for DM2000 due to DM1000 < 100, DM2000.
For control statements, see "Control statement".

■ About .D and .L

Suffix □.D processes BIN data as unsigned 32-bit data.
Suffix □.L processes BIN data as signed 32-bit data.

Correspondingly, the decimal representation relation between BIN data and corresponding suffix is shown below.

Device (DM1000)	Constant type	11111111111111111111111111111111 and bit data
DM1000.D	Unsigned DEC	4294967295
DM1000.L	Signed DEC	-1

- When comparing 32 bit BIN data (11111111111111111111111111111111) stored in DM1000/DM1001 with decimal constant 100, if suffix is .D, the value of [DM1000/DM1001] is compiled as 4294967295.

Programming Script
```
IF DM1000.D > 100 THEN
DM2000 = DM2000 + 1
END IF
```

Correspondingly, DM1001 (high bit)/DM1000 (low bit) (4294967295) > 100. Contact is ON, DM2000 plus "1".

- If the suffix is .L, the value of [DM1000/DM1001] is processed as "-1".

Programming Script
```
IF DM1000.L > 100 THEN
DM2000 = DM2000 - 1
END IF
```

Correspondingly, DM1001 (high bit)/DM1000 (low bit) (-1) < 100. Contact is OFF, DM2000 minus "1".

Point **During processing of 32-bit data (.D,.L,.F), device No. of even number must be used.**

■About .F

□.F suffix will process BIN data as single precision floating point type real number. The data is 32-bit type.

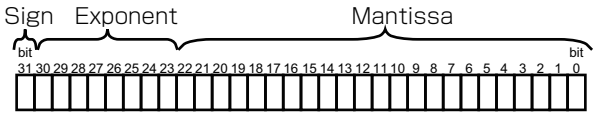
Single precision floating point type real number data is indicated with specific bit (internal representation).

So, direct bit operation and analysis on internal representation must be avoided.

Single precision floating point type real number value is a value represented according to the following method.

$$-1.234\times2^{-3}$$

Sign Mantissa Exponent



Sign (1 bit)	Exponent (8-bit)
0＝plus	00000000（$00）＝not used
1＝minus	00000001（$01）＝–126
	00000010（$02）＝–125
Mantissa section (23 bit)	00000011（$03）＝–124
0～8388607	⋮
	01111110（$7E）＝–1
	01111111（$7F）＝0
	10000000（$80）＝1
	10000001（$81）＝2
	11111100（$FC）＝125
	11111101（$FD）＝126
	11111110（$FE）＝127
	11111111（$FF）＝not used

Point

Suffix .D, .L and .F shall be added to the device when 32-bit data is processed by word devices such as DM/W/EM/FM/TM, etc.
In this case, the device uses 2 words with continuous No..
The device to be specified is that the suffix is added to low 16bits.
Example) Use DM1000.D → "DM1001, DM1000".

■About .DF

□.DF suffix will process BIN data as double precision floating point type real number. The data is 64-bit type.

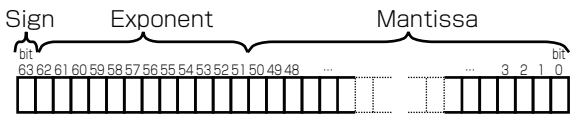
Double precision floating point type real number data is indicated with specific bit (internal representation).

So, direct bit operation and analysis on internal representation must be avoided.

Double precision floating point type real number value is a value represented according to the following method.

$$-1.2345678 \times 2^{-102}$$

Sign　Mantissa　Exponent



Sign (1 bit)	Exponent (12-bit)
0=plus	000000000000 ($ 000) = not used
1=minus	000000000001 ($ 001) = −2046
	000000000010 ($ 002) = −2045
Mantissa section (51 bit)	000000000011 ($ 003) = −2044
0 to 2251799813685248	⋮
	011111111110 ($ 000) = −1
	011111111111 ($ 7FF) = 0
	100000000000 ($ 800) = 1
	100000000001 ($ 801) = 2
	⋮
	000000000000 ($ 000) = 2045
	000000000000 ($ 000) = 2046
	000000000000 ($ 000) = 2047
	111111111111 ($ FFF) = not used

Point

Suffix .DF shall be added to the device when double precision floating point type real number is processed by word devices such as DM/W/EM/FM/TM, etc.

In this case, the device uses 4 words with continuous No.. When using .DF, use even-numbered devices.

The device to be specified is that the suffix is added to low 16bits.

Example) DM1000.DF → [DM1003•DM1002•DM1001•DM1000] .

.DF can be used with KV-8000/7000 CPU units, KV-5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.

■About .B

□.B suffix processes BIN data as bit data.

Example) Set the bit device MR1000 to ON (1, TRUE).

Programming Script SET(MR1000.B)

Reference If the word device is bit processed (suffix .B), and the lowest significant digit is the bit target. As mentioned in above example, DM10.B is specified in the argument of SET function, the least significant bit of DM10 changes to "1". The TRUE / FALSE judgment of the word device is as follows.

```
Example) IF DM10.B = ON THEN
             Processing statement
         END IF
```

Example) Set the bit device MR1000 to OFF (0, FALSE).

Programming Script RES(MR1000.B)

■About .T

Suffix □.T processes BIN data as text-string data.

Example) Continuous devices from DM100 stores "KEYENCE" text string.

Programming Script DM100.T = "KEYENCE" + CHR ($0D) + CHR ($0A)

above example, data storage status as follows.

Data storage status

bit 15	bit 0	
K(4BH)	E(45H)	:DM100
Y(59H)	E(45H)	:DM101
N(4EH)	C(43H)	:DM102
E(45H)	C_R(0DH)	:DM103
L_F(0AH)	NUL(00H)	:DM104

Reference NUL will be automatically stored into text string (00H).

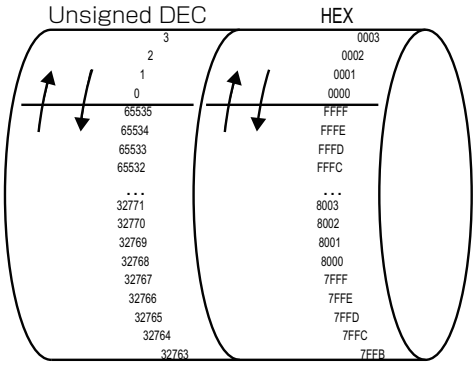
Point **Up to 1999 characters can be processed.**

■Value change when exceeding max./min. value

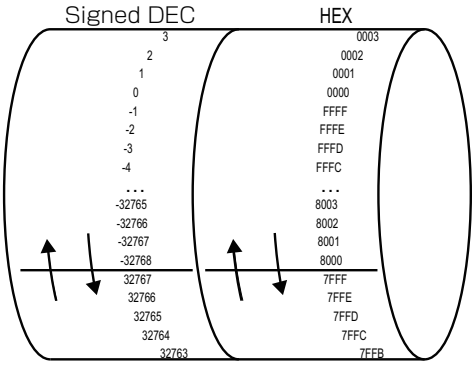
When arithmetic operation result exceeds maximum value or minimum value, according to data type, data change is shown below respectively.

● For 16-bit data

Value change when □.U larger than 65535

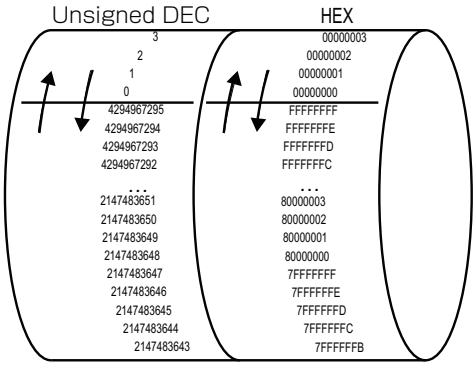


Value change when □.S larger/less than max./min. value

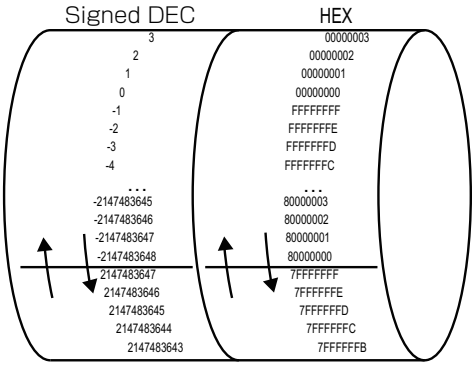


● For 32-bit data

Value change when □.D larger than 4294967295

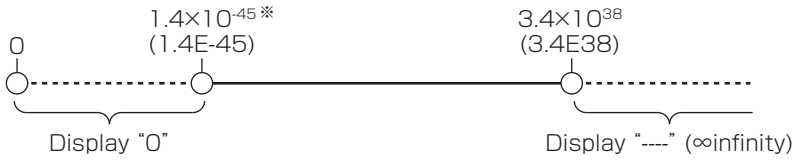


Value change when □.L larger/less than max./min. value



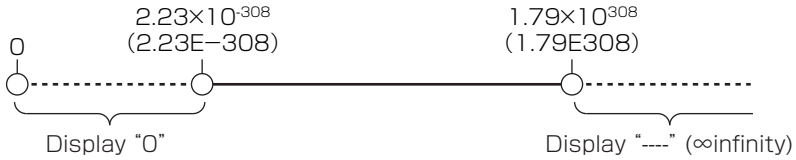
Reference

- For single precision floating point type data
  Overflow : when absolute value is larger than 3.4×10^{38}, it becomes "∞ (infinity)".
  Underflow : when absolute value is less than 1.4×10^{-45}*, it becomes "0".



* For the KV Nano Series, 1.2x10-38 (1.2E-38).

- For double precision floating point type data
  Overflow : when absolute value is larger than 1.79×10^{308}, it becomes "∞ (infinity)".
  Underflow : when absolute value is less than 2.23×10^{-308}, it becomes "0".



About Type Declaration (TYPE)

In KV script, the device whose suffix programming omitted will be processed as the pre-determined type (varies with different device). After type declaration, the type of device whose suffix programing omitted can be changed.
Type declaration is only valid in box script (or area script) of stated type.

- When programming via [device type + suffix (Example: DM.F)], the type of device shall be processed as the device with declared suffix.

Programming Script

TYPE DM1000.D	DM1000 will be processed as .D in the script
TYPE EM.F	Emnnnn will be processed as .F in the script
DM1000 = 10000 ∗ DM2000.D	Even if suffix is omitted, DM1000 can be still processed as .D
EM3000 = 123.456 / DM1200.F	All EM devices are processed as F

- Device type of specialized range can be also declared.

Programming Script

TYPE DM1200 - DM1600.F	DM1200 to DM1600 processed as .F
TYPE DM2000.L - DM2100	Programming method in another specified range
TYPE DM2200.D - DM2300.D	Programming method in another specified range
DM1200 = 12.5 * DM1100	Although DM1200 suffix is omitted, shall be also processed as .F
DM1210 = SIN(DM1300)	Argument and return value are processed as .F

- Even if the device with type declaration, other suffix will still be addressed firstly.

Programming Script

TYPE EM.F	Afterwards, all EM are .F in this script
EM1000.S = DM1100 * -10	But, after other suffix is specified,
EM3000.D = 1234 * DM1200	the specified suffix will be precedential
EM3000 = EM3000 + 10000	After suffix omitted, the device is processed as TYPE statement type.

Point

- **Type declaration can be programmed at the start of every script.**
- **All type declarations are invalid except the script with type declaration. Correspondingly, type shall be declared again in other scripts, or corresponding suffix shall be specified.**
- **Note: when using of global label, type declaration shall be precedential based on data format defined when registering a label.**

Example) TYPE DM.D
When programming above script, data format of all global labels of assignment DM is .D.

About Operation Type

For value operation via KV script, operators (+,-,∗, /) are used in programming instead of special instructions used like ladder language.

■Data range processed by operation formula

Each item of the operation formula is handled as .L (signed 32-bit), .F (single precision floating point type) or .DF (double precision floating point type). Thus, all devices of the formula are automatically sign extended to .L, .F or .DF.

● When operation formula does not contain floating point data (.F, .DF, constant value)

Example) DM3000.L = (DM1000.U + 30) * DM2000.U

The device in the operation formula is sign extended to .L once, so the above operation formula is calculated as

DM3000.L = (TOL(DM1000.U) + 30) * TOL(DM2000.U)

Example) DM3100.L = (DM1100.D + 30) * DM2100.U

In which, .D (unsigned-32 bit) data exists, symbol is expanded to .L, data precision will be impaired sometimes, this shall be noted.

Above operation formula is calculated as follows:

DM3100.L = (DM1100.L + 30) * DM2100.L

When "Script warning level setting" is "high", the following warning is displayed in output window.

Warning : data may lose in "+" operation. (unsigned 32-bit->signed 32-bit)

● When operation formula contains floating point data (.F, .DF, constant value)

Example) DM3000.F = (DM1000.S + 30) * DM2000.F + DM2500.U

If the operation formula has no .DF (double precision floating point type) but has an .F (single precision floating point type), .F is calculated as .L until it is used as an arithmetic element. Once .F becomes an arithmetic element, it is operated as .F.

In the above operation formula, data other than .L and .F is sign extended to .L or .F once, so the above operation formula is calculated as:

DM3000.F = (TOL(DM1000.S) + 30) * DM2000.F + TOF(DM2500.U)

Example) DM3000.DF = (DM1000.S + 30) * DM2000.F + DM2500.U + DM2600.DF - DM2700.U

If the operation formula has a .DF (double precision floating point type), .DF is calculated as .L or .F until .DF is used as an arithmetic element. Once D.F becomes an arithmetic element, it is operated as .DF.

In the above operation formula, data other than .L, .F and .DF is sign extended to .L, .F or .DF once, so the above operation formula is calculated as:

DM3000.DF = TOF(TOL(DM1000.S) + 30) ∗ DM2000.F + TOF(DM2500.U) + DM2600.DF - TODF(DM2700.U)

● When the result storage location is other than .L

Example) DM3200.S = (DM1200.U + 30) * DM2200.U

The result of calculations that do not include floating-point data must be in .L format. Therefore, when the format of the result storage location on the left is specified to be other than .L, data accuracy may be reduced. This should be noted.

Above operation formula is calculated as follows:

DM3200.S =TOS ((DM1200.U + 30) * DM2200.U)

When "Script warning level setting" is "high", the following warning is displayed in output window.

Warning :data may lose due to assignment "=". (signed 32-bit -> signed 16-bit)

Even if the device storing the results is .F, if the right-hand side contains .DF, the precision will drop.

Point

The warning on precision impairment of operation formula is not displayed in output window by default.

To display warning message, script warning level must be set to "high".

"Script Conversion Output Message", page 2-52

PRAGMA Directive

If the operation formula contains a single precision floating point type data (.F, constant), the operation is executed after converting the type to a single precision floating point type (.F). However, by using a PRAGMA directive, compilation can be instructed to convert the type to a double precision floating point type (.DF).
By operating with double precision, a drop in precision, including in the interim operation results, can be prevented.

Point **The PRAGMA directive instruction cannot be used with the KV-1000.**

■ Explanation of PRAGMA Directive Usage and Operation

The PRAGMA directive is written in pairs, such as #PRAGMA DOUBLE ON and #PRAGMA DOUBLE OFF in the same manner as the control syntax for IF statements and FOR statements. The interval starting with #PRAGMA DOUBLE ON and ending with #PRAGMA DOUBLE OFF is the effective range of the double precision operation.

● Operation formula

Programming Script

```
DM1000.F = 123.45
DM1004.S = -1998

DM2000.DF = DM1000.F * 3.141592        ' ...①
DM2004.DF = DM1004.S / 25              ' ...②

#PRAGMA DOUBLE ON
   DM2010.DF = DM1000.F * 3.141592     ' ...③   } Effective interval of PRAGMA
   DM2014.DF = DM1004.S / 25           ' ...④   } DOUBLE
#PRAGMA DOUBLE OFF
```

Operation description

The right-hand sides (1), (2) and (3), (4) are the exactly same operation formula, but (3), (4) specify to convert the floating point type operation to a double precision floating point type with a PRAGMA directive. (1), (2) are operated with a single precision floating point type and (3), (4) are operated with a double precision floating point type.

(1) The operation formula contains only a single precision floating point type (.F, constant) so it is operated using the single precision floating point type. The operation results are converted to the left-hand side type (double precision floating point type (.DF), and stored in DM2000 to DM2003 (4 words).

(2)The operation formula contains only an integer type (.S, constant), so the integer type value is converted into a signed 32-bit integer (.L) and then operated. The operation results are converted into double precision floating point type (.DF), and stored in DM2004 to DM2007 (4 words).

(3)The operation formula contains only a single precision floating point type (.F), but PRAGMA DOUBLE Is valid so each is converted into a double precision floating point type (.DF) and operated. The operation results are stored in DM2010 to DM2013 (4 words).

(4) Although PRAGMA DOUBLE is valid, the floating decimal point type data is not included in the calculation formula, so the calculation is performed according to the same rules as (2), and the operation result is converted into a double precision floating decimal point type (.DF) and then stored in DM2014 ~ DM2017 (4 words).

Watch window

Device	Current value	Display format
D1000	+123.45	FLOAT
D1004	-1998	Signed 16-bit DEC
D2000	+387.829528808594	DOUBLE FLOAT
D2004	-79	DOUBLE FLOAT
D2010	+387.829522812622	DOUBLE FLOAT
D2014	-79	DOUBLE FLOAT

The operation results differ for (1) (**DM2000**) and (3) **(DM2010)**.

Point

- **The size of the double precision floating point type (.DF) is 4 words (64 bits).**
- **When operating integers, they are first converted into signed 32-bit integers (.L) regardless of the validity of the PRAGMA directive. To operate the integers as floating point types, for example with case (4) above, part of the target device must be extended into a float point type such as:**

  DM2014.DF = TODF(DM1004.S) / 25

  Please expand some target devices to a floating decimal point type.

  In addition, as shown below, the same effect is obtained even if the constant is marked as a decimal.

  DM2014.DF = DM1004.S / 25.0
- **The operation precision increases by operating as a double precision type, but the execution speed is slower than the single precision type. Using the PRAGMA directive only where operation precision is required is the key point.**

Point

The PRAGMA directive cannot be written to cross other control statements (where control blocks cross).

"Precautions on use of nesting", page 4-32

[When FOR statement (repeated control statement) and PRAGMA directive intersect]

```
FOR <Word device (1) value> TO <end value>          'control statement (1)
    Process statement (1)
    FOR <word device (2) value> TO <end value>      'control statement (2)
        Process statement (2)
        #PRAGMA DOUBLE ON                           'PRAGMA directive
            Operation Formula
    NEXT
        #PRAGMA DOUBLE OFF
NEXT
```

Crossing of control blocks

● Operation of functions

Programming Script

```
    DM1000.DF = SIN(DM100.F)          ' ...①

#PRAGMA DOUBLE ON
    DM1010.DF = SIN(DM100.F)          ' ...②
#PRAGMA DOUBLE OFF
```

Effective interval of PRAGMA DOUBLE

Operation description

The right-hand sides of (1) and (2) are the exact same operation formula, but (2) specifies to convert the floating point type operation to a double precision floating point type with a PRAGMA directive. When a PRAGMA directive is issued, even when a single precision floating point type (.F) is specified for the function's argument, if the function specifications support double precision operation, the type will be converted to a double precision floating point type and executed.

(1)The argument is a single precision floating point type (.F) so it is operated using the single precision (SIN.F). The operation results are converted to the left-hand side type (double precision floating point type (.DF)) and stored in DM1000 to DM1003 (4 words).

(2)The argument's single precision floating point type (.F) is converted to a double precision floating point type (.DF) and operated with double precision (SIN.DF). The operation results are stored in DM1010 to DM1013 (4 words).

The script for (2) is written in the following manner without the PRAGMA directive.

Programming Script

```
DM1010.DF = SIN(TODF(DM100.F))
```

Point

Even if there is a function in the PRAGMA directive, the following functions will not be extended to double precision floating point type data.

- **TOU function (converted to .U (unsigned 16-bit integer))**
- **TOS function (converted to .S (signed 16-bit integer))**
- **TOD function (converted to .D (unsigned 32-bit integer))**
- **TOL function (converted to .L (signed 32-bit integer))**
- **TOF function (converted to .F (single precision floating point type data))**

4

SCRIPT SYNTAX AND PRECAUTIONS

Assignment Statement 4-2
Order of Priority low 4-6
Control statement 4-10
Precautions on Script Programming 4-38

Assignment Statement

Assignment statement is the statement whereby the result or value on the right side of equal sign (=) is assigned to the left side of equal sign.

How to Use An Assignment Statement

Contents (devices and variable expressions) that can be written on the right side of an assignment statement include values, strings and bits, etc. as shown below, and are written using constants, expressions or functions (with return value).

● Numerical value assignment statement

Assign right numerical constant or value calculation result to the left.

Programming Script

```
DM1000.U = 123                          'Assignment of numerical constant
DM1100.L = DM2000.U + 123               'Assignation of numerical operation result
DM1200.U = DCNT(DM2100,EM1000,20)       'Assignment of function processing result
```

● String assignment statement

Assign right character string constant or character string calculation result to the left.

Programming Script

```
DM100.T = "ABCD"                        'Assignment of character string constant
DM200.T = "current" + DM300.T + "use"   'Assignation of character string connection result
```

● Bit assignment statement

- Assign right bit constant or bit calculation (conditional formula evaluation) result to the left.

Programming Script

```
R1000 = ON                              'Assignment of bit constant
                                        (Boolean value)
R1000 = R1100 OR (R1200 AND R1300)      'Assignment of bit operation result
R1000 = DM100.S > 100                   'Assignment of conditional formula
                                        evaluation
R1000 = R2000 = R3000                   'Assignment of comparative result
```

- If multiple equal signs(=) are used on one line, the initial (the most left) equal sign(=) is processed as <=>; of assignment, the later equal signs(=) are processed as comparative sign(=).
  in this case, the left side of equal sign(=) assigned is processed as the left, and the right side processed as the right side.

(Example) **R1000 = R2000 = R3000 = R4000 → R1000 = (R2000 = (R3000 = R4**

↑ Assign ↑ Compare ↑ Compa

Left side←:→Right side

- Except ON, OFF, bit status can be also represented with 1, 0 and TRUE, FALSE.

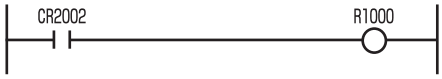
Programming Script

```
R1000 = 1
```

Operation description

R1000 Set R1000 to ON.

Ladder diagram Conversion



- Bit data can only be assigned to Boolean values.

  R1000 = TRUE . . . . . R1000 = ON
  R1000 = FALSE . . . . . R1000 = OFF

- The bit device can not be assigned to constants other than 1, 0 and assigned to result of value formula.

  × :R1000 = DM1000 + DM2000 . . . . . "+" Not used

- The bit device can be assigned to logical operation result and comparison operation result (results are Boolean value).

  ○: R1000 = R2000 AND R2100 . . . . . Logical operators allowed
  R1000 = DM1000 >= 500 . . . . . Equal sign, inequality sign allowed (calculation result is only formula of Boolean value)

● Operation assignment statement

Operation assignment statement is a assignment statement programed with right device omitted in four arithmetic operation for left assignment statement device.
There are 4 operation assign statements as follows.

+=, –=, ∗=, /=

Left side	right side	operation assignment statement
DM1000.U = DM1000.U + 1	→	DM1000.U += 1

Besides, when using operation assignment statement, right operation is based on type operation on the left side.

Programming Script

```
DM1000.U += 3                    'DM1000.U = DM1000.U + 3 (Right side as .U operation)*
DM1100.S –= 3                    'DM1100.S = DM1100.S – 3 (Right side operates as. S)*
DM1200.D ∗= 3                    'DM1200.D = DM1200.D ∗ 3 (Right side operates as. D)*
DM1300.L /= 3                    'DM1300.L = DM1300.L / 3 (Right side operates as.L )*
DM2000.F += DM1500–3             'DM2000.F = DM2000.F + TOF (DM1500 – 3)
DM2100.D += _                    'DM2100.D = DM2100.D + TOD (SRA (EM1000, 8)∗2)
SRA (EM1000, 8)*2
```

* formula operates as .L in general, but assignment statement shall be operated according to type operation on the left side.

The following script operation results are same (.L range), but LD converted is different.

Programming Script

```
DM1000.U = 10
DM1000.U = DM1000.U + 10         'Programming method A: operation result is 20
DM1000.U += 10                   'Programming method B: operation result is 20
```

Character string can be only added.

Programming Script

```
DM3000.T +="ABC"                 'And DM3000.T = DM3000.T + "ABC" Meaning same
```

■Precautions on using assignment statement

● When the type on right side is different from that on left side

- Assignment statement is FALSE when the types on right side and left side are different (type conversion cannot be performed).

Programming Script DM1000.T = DM2000.U 'Error

Operation description Assign right DM2000.U (constant: unsigned 16 bit) to DM1000.T (character string type) (conversion error)

- Even if the types are different, different types .U,.S,.D,.L,.F can also be converted (precision drops sometimes).

Programming Script DM1100.U = DM2100.D 'Script alarm grade “high” alarm

Operation description Assign right DM2100.D (unsigned 32 bit) to DM1100.U (unsigned 16 bit).
In such a case, low 16 bit is only assigned to DM2100.D.

"Setting method of warning level", page 2-53

● When it can not be assigned to left side

- In following script, the left "3" is value, and is not the device storing result and value. So it is incorrect as assignment statement.

Programming Script 3 = DM1000.U ''Error (can not be assigned to constant)

Operation description Assign right DM1000.U (unsigned 16 bit) to 3 (constant). (conversion error) (conversion error). changes error.

- During formula and function programming on the left side, incorrect as assignment statement, error appears.

Programming Script DM1100.U + 30 = DM1000.U 'Error (can not be assigned to formula)

Operation description Assign right DM1000.U (unsigned 16 bit) to left formula . (conversion error). changes error.

MEMO

Order of Prionity low

Operators is a data processing symbol, including addition (+), multiplication (∗), etc. used in working equation.

Operator List

■Operator type

The operators include the following 4 kinds: "arithmetic operator" comparison operator "character string connection operators" and "logical operator" etc.

Point **Operators shall be programed with single byte character.**

● Arithmetic operator

Arithmetic operator can return arithmetic operation result of 2 values.

Symbols	Processing content	Program example
^*1	Calculate the power	DM500 = DM100 ^ 2
∗	Calculate product of 2 values (multiplication)	DM502 = DM100 ∗ 3
/	Calculate quotient of 2 values (division)	DM504 = DM100 / 4
MOD*2	Divide 2 values, return remainder	DM506 = DM100 MOD 5
+	Calculate summation of 2 values (addition)	DM508 = DM100 + 6
—	Calculate difference of 2 values (subtraction)	DM510 = DM100 — 7

*1 Exponentiation can be used for .F and .DF suffixes with KV-8000/7000 Series CPU units, with KV-5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.

*2 1-byte space is required before and after symbol.

● String Connection Operator

Return connection result of 2 character strings.

Symbols	Processing content	Program example
&, +*3	Connect 2 character strings	DM600.T = "KEY" + "ENCE"

*3 “+”automatically identifies arithmetic operation and character string connection according to programming contents.

● Comparison operator

Return the evaluation result of size relation of 2 values in Boolean value (ON or OFF).

Symbols	Processing content	Program example
<	Less than	MR500 = DM100 < 10
<=	Less than or equal to (as follows)	MR501 = DM100 <= 20
>	Larger than	MR502 = DM100 > 30
>=	Greater than or equal to (above)	MR503 = DM100 >= 40
=	Equal to (equivalent)	MR504 = DM100 = 50
<>	Unequal to (unequivalent)	MR505 = DM100 <> 60

● Logic operator

In case bit relation operation for result of logical operation of 2 values and word relation operation Boolean value (ON or OFF), a word value is returned.

Symbols	Processing content	Program example
NOT*4	Calculat logical NOT of value	MR600 = NOT R000
AND*4	Calculate logical AND of 2 device (bit device relation, word device relation)	MR601 = R000 AND R001
OR*4	Calculate exclusive logical OR of 2 devices (bit device relation, word device relation)	MR602 = R000 OR R001
XOR*4	Calculate exclusive logical OR of 2 devices (bit device relation, word device relation)	MR603 = R000 XOR R001

*4 1-byte space is required before and after symbol.

Point

- **To operate device (operate element), please specify the relation of the bit device relation or the word device. Do not perform logical operation of bit device and word device, etc.**
- **When the types and sizes of operation device (operate element) are different, operate with larger type.**
  Example) DM1000.U and DM2000.D.... operate with 32 bit type

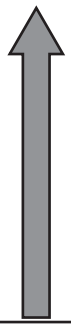
Operator priority

When multiple operators are used in 1 formula, each operation section of the formula is evaluated (calculated) according to the specified sequence. This sequence is called operator priority.

■ About order of priority

When different operators are used in a formula, evaluating is performed according to the sequence of arithmetic operator, comparison operator, and logical operator; besides, for arithmetic operator and logical operator, evaluating is performed according to the high to low sequence.

Order of priority high



Order of priority low

Arithmetic operator	Power	^
	Minus sign	—
	Multiplication and division	*, /
	Remainder operation	MOD
	Addition and subtraction	+, —
	String Concatenation	+
Comparison operator	Equal to, unequal to, Less than, more than, Below, above	=, <>, <, >, <=, >=
Logic operator	Logic Not	NOT
	Logic AND	AND
	Logic OR	OR
	Exclusive OR	XOR

● The priority between multiplication, division, addition, and subtraction

When multiplication and division (or addition and subtraction) exist in programming of 1 formula calculate according to the sequence (from left to right) of operators. Multiply divide is precedential in case of multiply divide and plus-minus method in programming.

Example) When DM1000 + DM2000 *30 - DM3000/10

Operating sequence 1: DM2000 * 30 .....result (1)
Operating sequence 2: DM3000/10 .....result (2)
Operating sequence 3: DM1000 + result (1) .....result (3)
Operating sequence 4: result (3) - result (2) .....final operation result

● Comparison operator priority

The priority of all comparison operators is same. So, when multiple comparison operators are specified in 1 formula, evaluating is performed according to the sequence(from left to right) of operators.

Example) When R000 = DM2 > 100 = R1 = FALSE assignment statement
R000 = (((DM2 > 100) = R001) = FALSE) Operation.

Compare sequence 1 : DM2 > 100 .....result (1)
Compare sequence 2 : result (1) = R001.....result (2)
Compare sequence 3 : result (2) = FALSE.....compare result
Operation 1 : compare result (ON during TRUE; OFF during FALSE)
assig to R000

● Logic operator priority

Device actual application example as follows.

Example) When R1000 = R000 OR R001 AND DM0 = 100 assignment statement
R1000 = (R000 OR (R001 AND (DM0 = 100))) Operation.
Operation sequence 1: DM0 = 100 comparison (ON or OFF) .....compare result (1)
Operating sequence 2: Compare result (1) and R001 logical AND .....result (2)
Operating sequence 3: Result (2) and R000 logical OR .....result (3)
Operating sequence 4: Assign result (3) to R1000

● Change priority

(if using parentheses ()), operation in “ ( ) ” is implemented before outer operation, so, priority can be changed, the part with working equation is firstly evaluated. The priority of operators is kept in "( )" .

Example) When DM5000 = DM3000 * (DM3001 + (DM3002 + DM3003 * 5) / 10)
Operating sequence 1: DM3003 * 5.....result (1)
Operating sequence 2: DM3002 +result (1).....result (2)
Operating sequence 3: Result (2) / 10 .....result (3)
Operating sequence 4: DM3001 + result (3) .....result (4)
Operating sequence 5: DM3000 * result (4).....Operating result
Operating sequence 6: Assign operation result to DM5000

Reference

Complicated programming operation and immethodical operation, is unrelated to the priority of operators, use "()" to clarify the programming operation step, improve maintenance (reading).

Differences between assignments statements and compare operators

Although the symbols (=) used by assignment statement and comparison operator are same, its meanings (application methods) are different.

- (statement) including statement of (=)..... assignment as assignment is TRUE

  Meaning: assign the result (value) on the right to the left

Example)

```
1 SELECT CASE DM2000          ' SELECT control statement
2 CASE 1
3     DM1000 = 100            ' Separately used as processing statement
4 CASE 2
5     DM3000 = DM1000 + 5     ' Separately used as processing statement
6 END SELECT
```

- As comparison operator (=). · · in statement including comparison operator, the statement is FALSE (process as a part of control statements such as conditional branch statements) Meaning: compare right side and left side, identify ON (TRUE) or OFF (FALSE)

Example)

```
7 IF DM1000 = 100 THEN        ' A part of control statement (IF statement)
8     BMOV(DM10,DM20,4)       ' Processing statement
9 END IF
```

* Although same “DM1000 = 100” is programed on the 3 rd line of assignment statement example and on the 7th line of comparison operator example, its meanings are different.

Control statement

Control statement is used to change the flowing direction of program, and, where applicable, enable processing flow branching or looping, change execution selection and the number of execution.

Control Statement List

Control statements include 2 types: "conditional branch statement" and "loop control statement".

■Conditional branch statement

Conditional branch statements enable processing flow branching according to the specified condition.

Control statement	Control statement	
IF statement	(1) Execute branch control statement according to conditional formula	

	IF Conditional formula A THEN
	Processing statement B
	END IF
		• If conditional formula A is TRUE, processing statement B will be executed. If conditional formula A is FALSE, any processing will not be performed.

	(2) Dual branch control statement shall be separately processed according to conditional formula	

	IF Conditional formula A THEN
	Processing statement B
	ELSE
	Processing statement C
	END IF
		• If conditional formula A is TRUE, processing statement B will be executed. If conditional formula A is FALSE, processing statement C will be executed.

	(3) Execute multiple branch control statements according to multiple conditional formula	

	IF Conditional formula A THEN
	Processing statement B
	ELSE IF conditional formula 2THEN
	Processing statement D
	.
	.
	.
	END IF
		• If conditional formula A is TRUE, processing statement B will be executed. conditional formula A is FALSE, conditional formula C is TRUE, processing statement D is executed.
		• If conditional formula A. C are FALSE, any processing will be not performed.

SELECT statement	According to conditional value, select corresponding processing, execute multiple branch control statement	
	SELECT CASE compare source CASE Conditional value A Processing statement B CASE Conditional value C Processing statement D . . . CASE ELSE Processing statement E END SELECT	• If compare source and conditional value A are same, processing statement B will be performed. If compare source and conditional value C are same, processing statement D will be performed. • If compare source is not conditional value A, C, etc., processing statement E will be performed.
MC statement	master control statement	
	MC Conditional formula A THEN Processing statement B MCR	• If conditional formula A is TRUE, processing statement B will be executed. If conditional formula A is FALSE, processing statement B will be executed under OFF condition.

■Loop control statement

Loop control statement is processed according to the specified conditional loop.

Control statement	Control statement	
FOR statement	The statement is a loop control statement for loop process before the conditional formula of the number specified is TRUE	
	FOR initial value TO final value STEP incremental value Processing statement A NEXT	• From initial value to final value, and also an incremental value , loop execution of processing statement.
WHILE statement	The variable number of loop control statement in loop process during conditional formula is TRUE	
	WHILE conditional formula Processing statement END WHILE	• Loop execution of processing statement during conditional formula is TRUE.
DO statement	The variable number of loop control statement in loop process before conditional formula is TRUE	
	DO Processing statement UNTIL conditional formula	• Loop execution of processing statement during conditional formula is FALSE.
BREAK statement	In loop control statement, unrelated to loop condition, exit from loop control	
	<loop processing statement> : BREAK : <loop processing statement>	• After BREAK execution, loop control including BREAK is ended irrespective condition.

IF ~ Statement (Conditional Branch (1))

IF statement is the most simple conditional branch statement to identify processing or not according to conditional formula.

IF ～ THEN ～ ELSE ～ END␣IF	
Basic format	**Format description**
IF␣ <Conditional formula>␣ THEN ↵ <Processing statement 1> ↵ END IF ↵	If conditional formula is TRUE, processing statement Processing statement 1 will be executed. IF statement end

* ␣Insert a single-byte space.
↵This is a line feed symbol.

Example of format	Explanation

IF R1000 = ON THEN
  DM1000 = DM1000 + 1
END IF
	'If R1000 is ON,
	'Constant 1 added to DM1000.

Tip

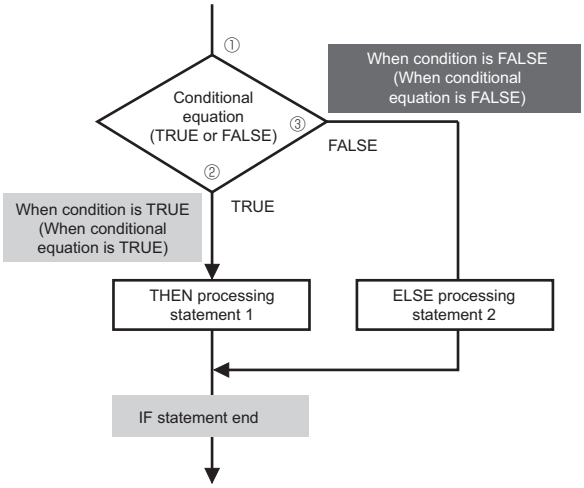
- If the result of conditional formula is bit (return Boolean value) formula, it can be used in any formula.
- Processing statement is a series of programming processing between THEN to END IF, capable of programming among lines if required.
- IF statement can be nested, so multiple-branch processing can be conducted by programming IF statement in processing statement and selecting AND condition.

Reference

Nesting of IF statements (writing of IF statements in IF statements) can be up to 200 times in the area script and up to 199 times in the box script.

"About control statements nesting", page 4-32

Processing flow



(1) Identify "TRUE" or "FALSE" via conditional formula.
(2) If conditional formula to "TRUE", processing statement following THEN will be executed, and END␣IF executed.
(3) if conditional formula is "FALSE", jump to END␣IF, any operation is not executed, processing after jump to END␣IF.

Programming example

Programming Script

```
IF (10 <= DM1000) AND (DM1000 <=_
    50) THEN
FM1000.T ="within the range"

END IF
```

'if DM 1000 is more than 10 or less than 50
'specifies that the character string is saved to FM 1000.T
'IF statement ends

Operation description

If DM1000 value is more than 10 and less than 50, character string "within range" is stored into FM1000.T.
If the value is out of the range, any operation will be not executed.

Reference

The part from (single quote) (or "REM") to the end of line(before line feed input) is processed as comment. So there is no any influence on actual operation.

"About Comments Function", page 2-69

IF ~ ELSE Statement (Conditional Branch (2))

The statement refers to a simple conditional branch statements dividing 1 conditional formula into 2 processing methods.

IF ～ THEN ～ ELSE ～ END␣IF	
Basic format	**Format description**
IF␣<Conditional formula>␣THEN ↵	If conditional formula is TRUE,
<Processing statement 1> ↵	processing statement 1 will be executed.
ELSE ↵	If conditional formula is FALSE,
<Processing statement 2> ↵	processing statement 2 will be executed
END␣IF ↵	IF statement end

* ␣Insert a single-byte space.
↵This is a line feed symbol.

Example of format	Explanation
IF R1000 = ON THEN	'If R1000 is ON,
DM2000 = 50	'The constant 50 is stored in the DM2000.
ELSE	'If R1000 is OFF,
DM2000 = 10	'The constant 10 is stored in the DM2000.
END IF	

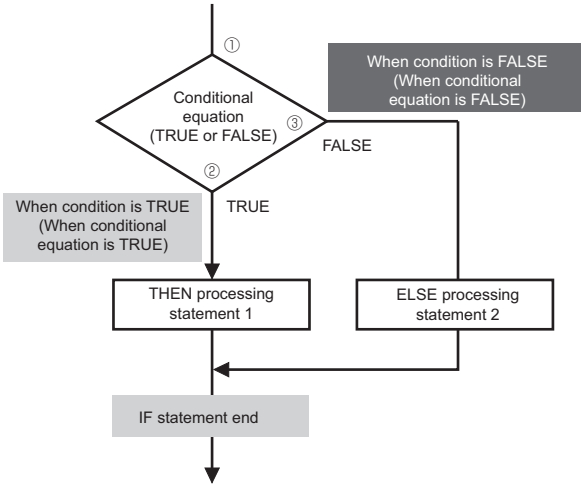
Tip

- If the result of conditional formula is bit (return Boolean value) formula, it can be used in any formula.
- Processing statement is a series of programming processing between THEN to END IF, capable of programming among lines if required.
- IF statement can be nested, so multiple-branch processing can be conducted by programming IF statement in processing statement and selecting AND condition.

Reference

IF statement nesting (program IF statement in IF statement), maximal upto 200 layers.
"About control statements nesting", page 4-32

Processing flow



(1) Identify "TRUE" or "FALSE" via conditional formula.
(2) If conditional formula is "TRUE", processing statement 1 below THEN will be executed, and jump to END IF(middle formula will be not executed).
(3) If conditional formula is "FALSE", the program between THEN ~ ELSE will be not executed, but processing statement 2 below ELSE will be executed.

Programming example

Programming Script

```
IF R1000 THEN                          'If R1000 is ON (the abbreviated form
                                       for R1000= ON)
   DM1000.F = COS(DM1100)              'COS function will be processed.
ELSE                                   'if R1000 is not ON (OFF),
   IF R0002 THEN                       'nested IF statement: if R0002 is ON,
      DM1000.F = SIN(DM1100)           'SIN function will be processed.
   ELSE                                'If R0002 is OFF,
      BMOV(DM5000,DM5500,10)           'BMOV function will be processed
   END IF                              'nested IF statement ends
END IF                                 'IF statement ends
```

Operation description

If contact R1000 is ON, COS (DM1100) result will be stored in DM1000.F.
If contact R1000 is not ON (OFF), and contact R0002 is ON, return value of SIN (DM1100) will be stored in DM1000.F.
If R1000 is not ON (OFF), and R0002 is also not ON (OFF), BMOV function will be used to transfer 10 word data of DM5000 to DM5500.

"SIN function", page 7-140
"COS function", page 7-142
"BMOV function", page 7-2

IF ~ ELSE␣IF Statement (Conditional Branch (3))

If the initial conditional formula is not met, it will be identified again according to the next conditional formula. The conditional branch statement may have multiple conditions.

IF ～ THEN ～ ELSE␣IF ～ END␣IF	
Basic format	**Format description**
IF␣ <Conditional formula 1>␣THEN ↵	If conditional formula 1 is TRUE
<Processing statement 1> ↵	processing statement 1 will be performed.
ELSE␣IF␣ <Conditional formula 2>␣THEN ↵	If conditional formula 1 is FALSE, conditional formula 2 is TRUE,
<Processing statement 2> ↵	processing statement 2 will be performed.
ELSE␣IF␣<Conditional formula 2>␣THEN ↵	If conditional formula 1.2 is FALSE, conditional formula 3 is TRUE, } ※2
⋮ ⋮	⋮
ELSE※1 ↵	If all conditional formula are FALSE,
<Processing statement n> *1 ↵	processing statement n will be executed.
END␣IF ↵	IF statement ends here.

* ␣Insert a single-byte space.
↵ This is a line feed symbol.

*1 "[ ]"can be omitted.

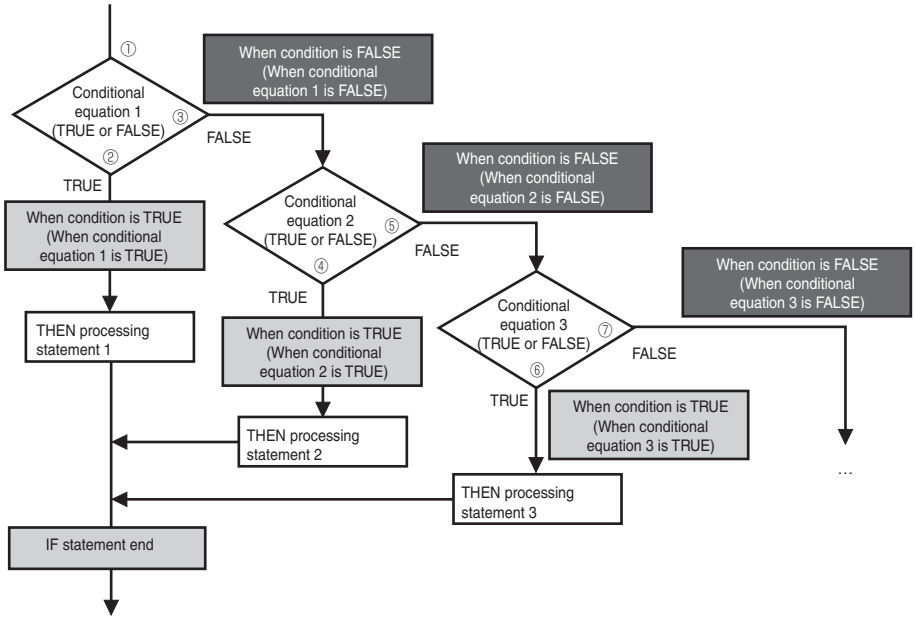
*2 You can set any counter maximal value. There is no upper limit.

Example of format	Explanation
IF R1000 THEN	'If R1000 is ON (the abbreviated form for R1000 = ON)
DM1000 = 1000	'1000 is saved into DM1000.
ELSE IF R1001 THEN	'If R1000 is not ON, R1001 is ON,
DM1000 = 1001	'1001 is saved into DM1000.
ELSE IF R1002 THEN	'If R1000, R1001 are not ON, R1002 is ON,
DM1000 = 1002	'1002 is saved into DM1000.
ELSE IF R1003 THEN	'If R1000, R1001, R1002 are not ON, R1003 is ON,
DM1000 = 1003	'1003 is saved into DM1000.
ELSE IF R1004 THEN	'If R1000, R1001, R1002, R1003 are not ON, R1004 is ON,
DM1000 = 1004	'1004 is saved into DM1000.
ELSE	'If R1000, R1001, R1002, R1003, R1004 are not ON,
DM1000 = 1005	'1005 is saved into DM1000.
END IF	

Tip

- If the result of conditional formula is bit (return Boolean value) formula, it can be used in any formula.
- Processing statement is a series of programming processing between control statements, capable of programming among lines if required.
- ELSE IF statement is different from IF statement, multiple lines are programmable, but it is not a nesting structure.

Processing flow



(1) Identify "TRUE" or "FALSE" via conditional formula 1.
(2) If conditional formula is "TRUE", processing statement 1 below THEN will be executed, and jump to END IF(middle formula will be not executed).
(3) If conditional formula is "FALSE", "TRUE" or "FALSE" will be identified with conditional formula 2.
(4) If conditional formula 2 is "TRUE", processing statement 2 will be executed, and jump to END IF (middle formula is not executed).
(5) If conditional formula 2 is "FALSE", "TRUE" or "FALSE" will be identified with conditional formula 3.
(6) If conditional formula 3 is "TRUE", processing statement will be executed, and jump to END IF (middle formula is not executed).
(7) If conditional formula 3 is "FALSE", "TRUE" or "FALSE" will be identified with conditional formula, and the like.

SELECT ␣ CASE ~ Statement (Multi-branch control)

The conditional branch statements can be processed according to the value stored in reference word device.

SELECT␣CASE ～	
Basic format	**Format description**
SELECT␣CASE␣Word device ↵	According to the value of word device, conditional branch
CASE␣Constant 1 *3 ↵	If constant 1 and word device value are same,
Processing statement 1 ↵	processing statement 1 will be performed.
CASE␣Constant 2 ↵	If constant 2 and word device value are same,
Processing statement 1 ↵	processing statement 2 will be performed.
CASE␣Constant 3 ↵	If constant 3 and word device value are same, } *2
⋮ ⋮	⋮
CASE␣ELSE*1 ↵	If constant and word device value are not same,
Processing statement n *1 ↵	the processing statement will be performed.
END␣SELECT ↵	SELECT statement ends here.

* ␣Insert a single-byte space.
↵This is a line feed symbol.

*1 "[ ]"can be omitted.

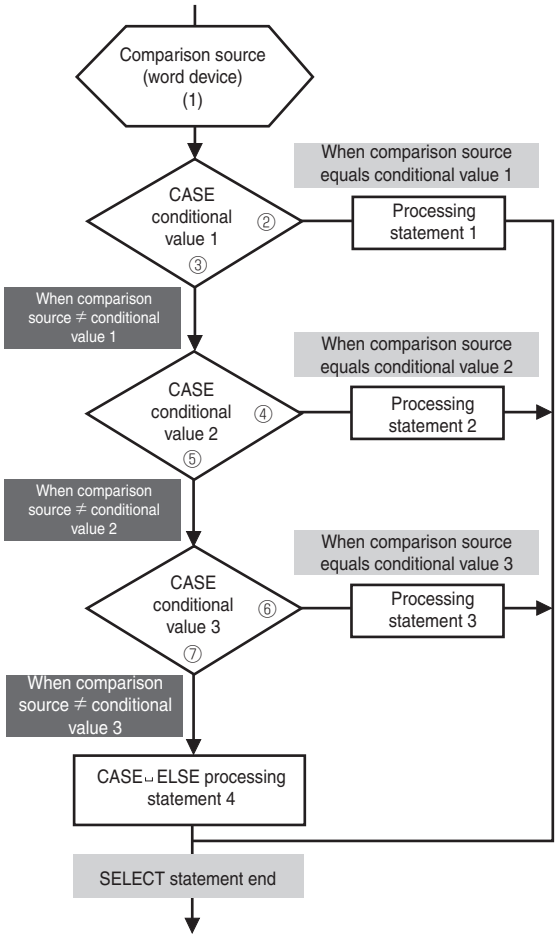
*2 Up to 200 CASE constants (ELSE) can be programed.

*3 A character string, device which handles a character string, or a bit type device cannot be specified for constant 1.

Example of format	Explanation
SELECT CASE DM1000	'With DM1000 store value as basis
CASE 1	'If DM1000 = 1,
DM2000 = 1000	'1000 is saved into DM2000.
CASE 2 TO 10	'If DM1000 = 2 to 10,
DM2000 = 2000	'2000 is saved into DM2000.
CASE 11,15,60	'If DM1000 = 11 or 15 or 60,
DM2000 = 3000	'3000 is saved into DM2000.
CASE IS > 100	'if DM1000 exceeds 100,*
DM2000 = 4000	'4000 is saved into DM2000.
CASE ELSE	'If DM1000 value is the value other than above,
DM2000 = 5000	'5000 is saved into DM2000.
END SELECT	

* The conditional formula using comparison operator can be written into the constant part. In this case, " IS" is used for comparison.
example) IS >= 100 (more than 100), IS < > 30 (values other than 30)

Processing flow



(1) The declaration acts as the identification reference value of conditional value.
(2) Compare conditional value 1 and compare source, if "conditional value 1 = compare source" is TRUE, processing statement 1 will be executed. After executing processing statement 1, jump to ENDSELECT.
(3) If "conditional value 1 = compare source" is FALSE, transfer to compare the next conditional value.
(4) Compare conditional value 2 and compare source, if "conditional value 2 = compare source" is TRUE, processing statement 2 will be executed. Execute processing statement 2, then, jump to END ␣ SELECT.
(5) If "conditional value 2 = compare source" is FALSE, transfer to compare the next conditional value.
(6) Compare conditional value 3 and compare source, if "conditional value 3 = compare source" is TRUE, processing statement 3 will be executed. Execute processing statement 3, then, jump to END ␣ SELECT.
(7) If "conditional value 3 = compare source" is FALSE, perform processing statement 4.

Tip

- The selected constant may be single constant, or multiple specifying and range specifying constants.

Example) CASE 1 .....Single specifying (only 1)

CASE 3,9,12,50 .....Multiple specifying (3, 9, 12 or 50)

CASE 8 TO 20.....Range specifying (above 8, below 20)

CASE IS>=100 .....Use comparison operator specifying range (above 100)

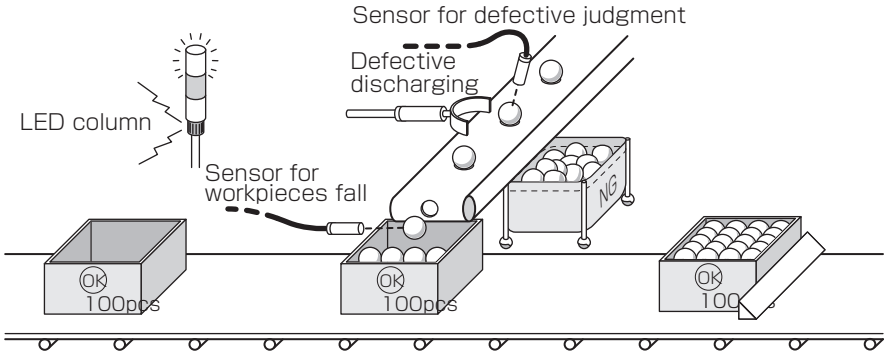
- CASE ELSE "Processing statement" script can be omitted.
  But CASE ELSE of " all CASE conditions are FALSE " can help solving the accident.

Reference script example

- Count the number of work pieces dropped out from belt conveyer. When work piece falls into the case, "1" is added to Good Counter (DM1000).
  After reaching 100 good articles, the case is changed. When reaching 200, the bell rings, and the case is changed. When 300, the alarm sounds, operation is shut off, and the Good Counter is reset.

  [reference script example]

```
IF LDP(R000) = ON THEN        'If workpiece drop sensor ( R000) is ON,
   DM1000 += 1                'Good counter (DM1000) plus _1
END IF
SELECT CASE DM1000            'Compare source is DM1000
CASE 100                      'If DM1000 value reaches 100,
  "change box"
CASE 200                      'If DM1000 value reaches 200,
  "bell ringing"
  "change box"
CASE 300                      'If DM1000 value reaches 300,
  "alarm ringing"
  "shutdown"
   DM1000 = 0                 'Reset DM1000 value.
CASE ELSE                     'If DM1000 value is the value other than 100, 200, 300,
  "display in operation"
  "warning blue lamp lit"
END SELECT                    'SELECT CASE statement ends
```



Point

If the selected constants are multiple, the processing statement written on the upper line is firstly executed; the processing statement written later will be not executed.

Programming Script

```
SELECT CASE DM1000          'Set selection reference to DM1000.
CASE 1                      'If DM1000 value is “1”,
  Processing statement 1
CASE 3 TO 20                'If DM1000 value is above 3, below 20,
  Processing statement 2
CASE IS >= 15               'If DM1000 value is more than 15
  Processing statement 3
CASE ELSE                   'If the value of DM1000 is something other
                            than the above,
  Processing statement 4
END SELECT
```

* if the value stored in DM1000 (compare source) is 17, the 2nd CASE and the 3rd CASE, will be matched. In this case, the 2nd CASE is given the priority for selection, and processing statement 2 will be executed. After processing statement is executed, END␣SELECT is jumped to, and processing statement 3 is not executed.

MC ~ MCR Statement (Master control)

The script is used to program Master Control.

For action of Master Control, see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual".

MC ～ THEN ～ MCR	
Basic format	**Format description**
MC␣<MC Execution condition > ␣THEN ↵	If MC execution condition is TRUE, master control processing will be conducted.
<Processing statement 1> ↵	Execute processing statement.
MCR ↵	If MC execution condition is FALSE, processing statement will be executed in OFF status.

* ␣Insert a single-byte space.
↵This is a line feed symbol.

Example of format	Explanation
MC R1000 THEN	'If R1000 is ON, Master Control processing will be conducted.
R500 = ON	'Execute after setting R0500 to ON.
MCR	'If R1000 is OFF, set R0500 to OFF before execution.

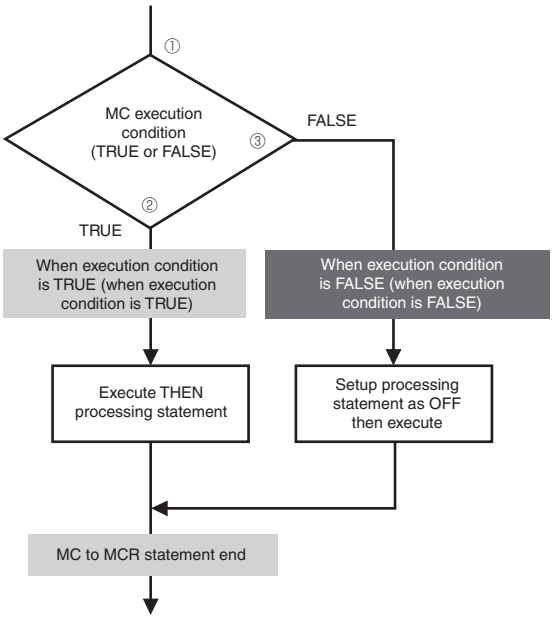
If MC execution condition is OFF, the function between MC to MCR will be executed in OFF status.

For details please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", MC, MCR command.

Tip

- If the result of the MC execution condition is a bit (return Boolean) expression, it can be used in any expression.
- Processing statement is a series of programming processing between MC to MCR, which can be used in programming multiple lines if required.

Processing flow



(1) Identify "TRUE" or "FALSE" via MC execution condition, determine MC to MCR processing method.
(2) If execution condition is "TRUE", processing statement below THEN will be executed.
(3) If conditional formula is "FALSE", will be executed after processing statement below THEN is set to OFF.

Point

Up to 16 layers are available for MC to MCR statement composition nesting (program MC to MCR statement in MC to MCR statement).

"About control statements nesting", page 4-32

FOR ~ NEXT Statement (specified loop control times)

According to loop condition, the control statement processing the number of statement executed is determined.

FOR ～ NEXT	
Basic format	
1	FOR ␣<Word device>=<Initial value >␣ TO ␣ <Final value >[␣STEP␣ <incremental value >] *1 ↵
2	<Processing statement> ↵
3	NEXT ↵
Format description	
1	· Determine initialization and termination value of word device limiting the number of loop controls. Determine how many word device values willbe added to every processing via STEP (the line is not added)
2	· The processing statement will be executed before word device exceeds final value.
3	· Plus incremental value to word device.

*1 Programming in [ ] can be omitted. If omitted, will be executed as STEP␣1.
␣Insert a single-byte space.
↵This is a line feed symbol.

Example of format	Explanation
FOR DM1000 = 0 TO 100 STEP 5	'DM1000 value loop s from 0 to more than 100.
DM2000 = DM2000 + 5	Plus 5 to DM2000 and store to DM2000
NEXT	'Add incremental value 5 to current value of DM1000, return FOR ~

Tip

Multiple elements are programed on the 1st line of format example. It shall be described separately.

Row 1: $\underbrace{\text{FOR␣}}_{④}\underbrace{\text{DM1000=0␣}}_{①}\underbrace{\text{TO␣100␣}}_{②}\underbrace{\text{STEP␣5}}_{③}$

(1)The specified word device can use any content. The initial value needs to be stored first. The initial value is only processed in the first time of executing FOR ~ NEXT statement.

(2) Specify the final value. FOR ~ NEXT identifies whether the value of word device exceeds final value, and branches the "execute processing statement (the final value is not exceeded)" and "terminating loop control (the final value is exceeded)" .

(3)Incremental value can be omitted. "STEP␣1" will be processed when omitted. The incremental processing of the word device specified is not executed in the first time. The execution sequence is before NEXT statement(the 3rd line of format example).

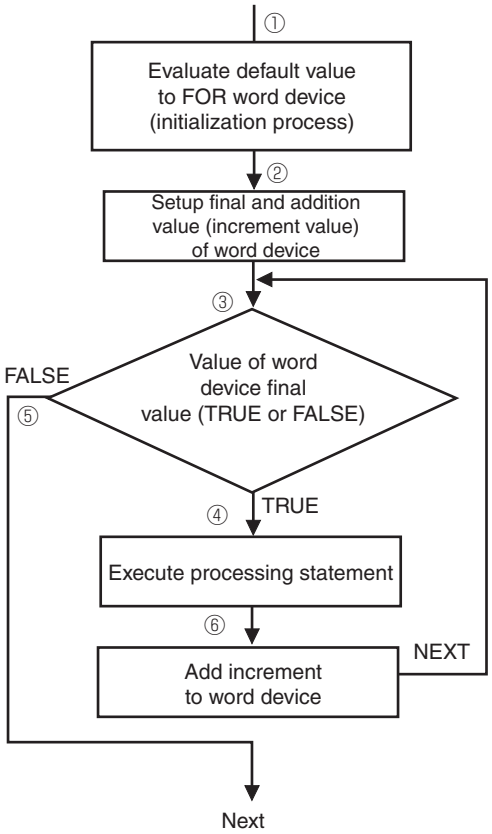
(4) Identify whether the value of word device specified meets the conditions.

Reference

According to constant symbol of STEP program, automatically identify "final value" and "word device final value".

例）FOR DM1100 = 100 TO 0 STEP － 5
Subtract 5 from 100, the value stored in DM1100 before 0.

Processing flow



(1) Assign the initial value to word device.
(2) Set final value indicating the end of loop process. Besides, set the increment value added on the value of word device.
(3) Identify whether the value of word device is less than final value.
(4) If (3) condition is "TRUE", processing statement will be executed.
(5) If " FALSE", loop process statement will be terminated.
(6) Plus incremental value to the value of word device, and return (3).

Point

If programming single precision floating point type real number on added value (incremental value), suffix .F shall be added on the specified word device.

Example) when setting the value to "0.5"

```
FOR DM2000.F = 0 TO 100 STEP 0.5      'DM2000 must be .F
        ⋮
```

Reference

FOR ~ NEXT NEXT statement will execute following processing.

- Specify word device +=incremental value

Precautions on Using FOR ~ NEXT statement

● The range that can be specified with STEP

In programming of FOR ~ NEXT statement, the incremental value specified by STEP is as follows.

The range that can be specified by STEP : -32767 to 32767 (-32768 cannot be specified)

If the range value is specified, error will occur during script conversion.

● When specifying single precision floating point type constant with STEP

The constant with decimal point can be specified with incremental value specified by STEP, but, shall be noted.

The constant with decimal point is automatically converted to single precision floating point type real number. In this case, some error will occur, error will also occur in loop index sometimes.

Programming example

● When FOR~NEXT statements are imperatively ended

Before meeting condition set by FOR ~ NEXT statement, when loop control statement is imperatively ended with other conditions, BREAK statement will be used.

Programming Script

```
FOR DM1000 = 0 TO 1000 STEP 10      'Before DM1000 value exceeds 1000
DM2000 += 9                         'DM2000 value plus 9 every time
IF DM2000 > 100 THEN                'DM2000 value exceeds 100
BREAK                               'Terminate the loop control
END IF                              'IF statement ends
NEXT                                'Add incremental value, return FOR ~
```

For details please refer to "BREAK Processing", page 4-31.

Point

A script should be programmed in the loop control statement to end loop control. If the loop processing is not stopped, this then becomes "infinite loop" (permanent loop processing) and scanning time out occurs during program execution (infinite loop is not detected as error during conversion).

"Loop control statement and processing time", page 4-35

Reference

Up to 100 layers are available for nesting of FOR ~ NEXT statement.

"About control statements nesting", page 4-32

WHILE Statement (pre-judge loop control)

When loop condition is "TRUE", execute control statement of loop process.

WHILE ～ END␣WHILE	
Basic format	**Format description**
WHILE␣<conditional formula> ↵	If conditional formula is TRUE, *1
<processing statement> ↵	executes processing statement.
END␣WHILE ↵	Return to the line start (WHILE).

* ␣Insert a single-byte space.
↵This is a line feed symbol.

*1 If the conditional formula is FALSE, loop control is then ended.

Example of format	Explanation
WHILE DM1000 < 100	'If DM1000 value is less than 100, *2
DM1000 += 5	'Add 5 to DM1000.
END WHILE	Return to the line start (WHILE).

*2 If DM1000 value is above 100, the processing statement will be not executed, will be executed from the next line of END WHILE.

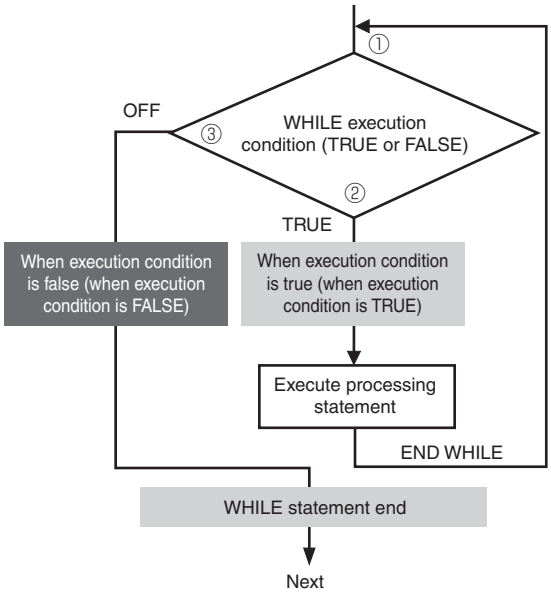
Tip

- If the result of conditional formula is bit (return Boolean value) formula, it can be used in any formula.
- Processing statement is a series of programming processing between WHILE ~ END WHILE, capable of programming among lines if required.

Reference

In loop control statement of WHILE statement/DO ~ UNTIL statement, maximal 199 layers are programmable for nesting of combination composition.
Combination and nesting via master control and IF statement can be also conducted.
"About control statements nesting", page 4-32

Processing flow



(1) Identify " TRUE " or "FALSE" with WHILE execution condition, determine processing method of WHILE ~ END WHILE.
(2) If execution condition is "TRUE", processing statement will be executed, and return (1) (WHILE statement start).
(3) If conditional formula is "FALSE" END WHILE statement will enter the next step.

Programming example

● When WHILE statement is imperatively terminated

In WHILE~END␣WHILE statement, before meeting loop termination condition, to imperatively terminate loop control statement with other conditions, use BREAK statement.

Programming Script

```
'From loop control statement, exit via BREAK
    statement
WHILE R1000 = OFF                    'During R1000 is OFF
                                     (condition is on TRUE),
  IF (DM2000 MOD 8) = 0 THEN         'If DM2000 becomes a multiple of 8,
                                     then
    ZRES(DM1000, DM1100)             'DM1000 to DM1100 is reset
  ELSE IF DM2000 > 100 THEN          'If DM2000 value exceeds 100
    BREAK                            'Terminate the loop control
  END IF                             'IF statement ends
  DM2000 += 1                        'Add 1 to DM2000.
END WHILE                            'Return WHILE ~.
```

For details please refer to 📖 "BREAK Processing", page 4-31.

Point

A script should be programmed in the loop control statement to end loop control. If the loop processing is not stopped, this then becomes "infinite loop" (permanent loop processing) and scanning time out occurs during program execution (infinite loop is not detected as error during conversion).
📖 "Loop control statement and processing time", page 4-35

DO ~ UNTIL Statement (Post Identification Loop Control)

When loop condition is "FALSE", execute control statement of loop process.

DO ～ UNTIL	
Basic format	**Format description**
DO ↵	Perform loop control
<Processing statement> ↵	Perform Processing statement
UNTIL␣<conditional formula> ↵	If conditional formula is FALSE, return to statement start (DO). *1

* ␣Insert a single-byte space.
↵This is a line feed symbol.

*1 If the conditional formula is FALSE, loop control is then ended.

Example of format	Explanation
DO	'process at the start of next step.
DM1000 += 5	'Add 5 to DM1000.
UNTIL DM1000 >= 100	'If DM1000 value is not above 100, return to sentence leading (DO).*2

*2 If DM1000 value is above 100, processing statement will be not executed, the execution will start from the next line of UNTIL.

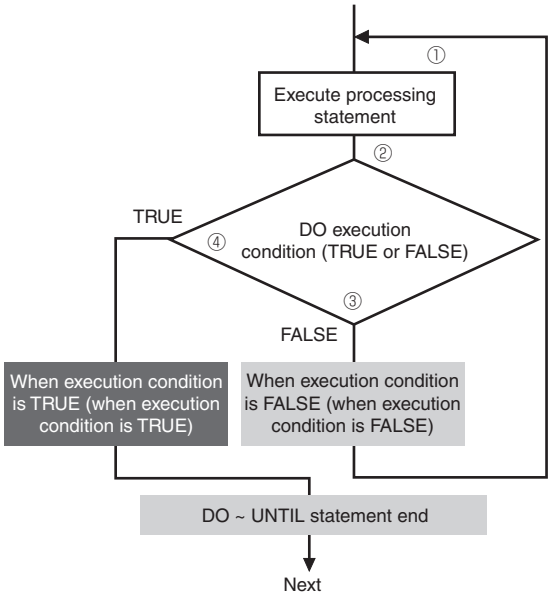
Tip

- If the result of conditional formula is bit (return Boolean value) formula, it can be used in any formula.
- Processing statement is a series of programming processing between DO ~ UNTIL, capable of programming among lines if required.
- The control statements that imply the same actions as those implied by DO ~ UNTIL statement can use WHILE statement.
- DO ~ UNTIL statement is firstly processed, then conditional formula identified. loop identification condition FALSE .
  For details please refer to "WHILE Statement (pre-judge loop control)", page 4-27.

Point

In loop control statement of WHILE statement/DO ~ UNTIL statement, maximal 199 layers are programmable for nesting of combination composition.
Combination and nesting via master control and IF statement can be also conducted.
"About control statements nesting", page 4-32

Processing flow



(1) Execute processing statement.

(2) Identify "TRUE" or "FALSE" via DO execution condition, determine DO to UNTIL processing method.

(3) If conditional formula is "FALSE", return to (1) (DO ~ UNTIL statement leading).

(4) If conditional formula is "TRUE", will enter next step.

About difference between WHILE statement between DO statement

The control statements that imply the same actions as those implied by DO ~ UNTIL statement can use WHILE statement.

The difference as follows

	WHILE statement	DO statement
Sequence different	1. Identification of conditional formula (TRUE or FALSE) 2. Execute loop process statement 3. Return to sentence leading	1. Execute loop process statement 2. Identification of conditional formula (TRUE or FALSE) 3. Return to sentence leading
Execute loop process statement	perform repeatedly when condition is"TRUE"	perform repeatedly when condition is"FALSE"

For details please refer to "WHILE Statement (pre-judge loop control)", page 4-27.

WHILE statement · · · · primary loop control statement
(primary processing statement will be not executed according to conditions)
DO statement · · · · post identification loop control statement
(1 processing statement must be performed)

Point

A script should be programmed in the loop control statement to end loop control. If the loop processing is not stopped, this then becomes "infinite loop" (permanent loop processing) and scanning time out occurs during program execution (infinite loop is not detected as error during conversion).

"Loop control statement and processing time", page 4-35

BREAK Processing

It is used in force termination of loop control (exit from loop).

BREAK	
Basic format	**Format description**
<loop control statement start> ↵	Program in loop process.* 1
BREAK ↵	Force termination of the loop process
"loop control statement end" ↵	loop control statement end

* ↵ This is a line feed symbol.

*1 BREAK must be programmed in loop process statement.

Example of format	Explanation
FOR DM1000 = 0 TO 50 STEP 2	'DM1000 from 0 to 50, loop.
DM2000 += 5	'Add 5 to DM2000.
IF DM2000 > 70 THEN	'If DM2000 value exceeds 70,
BREAK	'Imperatively terminate loop control (FOR ~ NEXT statement).
END IF	'IF statement end.
NEXT	'Plus 5 to DM1000, return to sentence leading of FOR statement.

* Except FOR ~ NEXT, loop process statement can be also used in WHILE statement, DO ~ UNTIL statement.

Point

BREAK must be written inside loop process statement.
If it is written outside the loop process, conversion error will occur.

Programming example

When BREAK statement is programmed in the nested loop control statement, the loop control statement with BREAK programming is only terminated.

Programming Script

```
FOR DM1000 = 1 TO 100 STEP 3                    'FOR statement (1)
  FOR DM2000 = 1 TO 100 STEP 3                  'FOR statement (2)
    FOR DM3000 = 1 TO 100 STEP 3                'FOR statement (3)
      DM4000 = DM1000 + DM2000 + DM3000         'Processing statement
      IF DM4000 > 100 THEN                      'BREAK Execution condition
        BREAK
      END IF
    NEXT                                        'When Break execute, FOR
                                                statement (3)
                                                'End.
  NEXT
NEXT
```

Operation description

FOR statement (3) contains control statement of BREAK statement, so, only FOR statement (3) is terminated during execution of BREAK.

"Precautions on Using Control Statement", page 4-32

Precautions on Using Control Statement

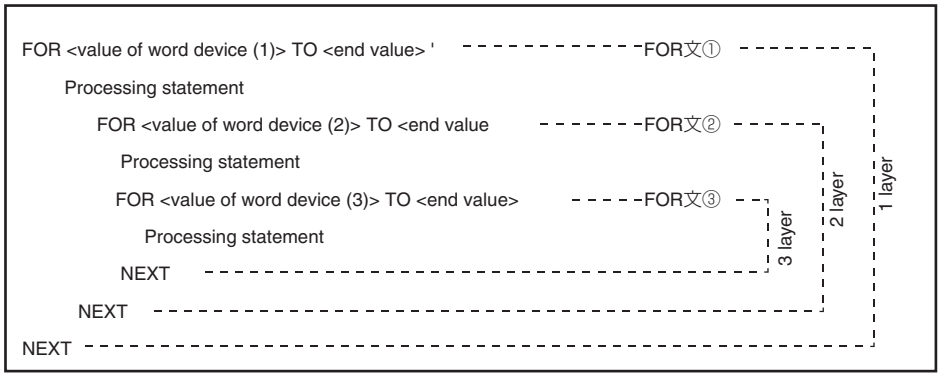
About control statements nesting

Control statement can be processed in conditional branch processing and loop process, so that other conditional branch and loop process can be performed.

■ Control statements nesting

Nesting means that a case is embedded into another case, other loop control statement is programmed in a loop control statement, thus forming a multi-layer control block. In the following example, FOR statement (2) is nested in FOR statement (1), and then 3 layers of FOR statement (3) are nested in it.

"nest programing example"



● Precautions on use of nesting

All conditional branch statements and loop control statements can be nested, but with the following limit.

- FOR statement is limited on separate layer, maximal100layers can be nested.
- WHILE statement, DO statement are limited on separate layer, maximal199layers can be nested.
- Conditional branch statements (IF statement, SELECT statement) with separate layer limit, maximal 200 layers can be nested.
- Master control with separate layer limit, maximal 16 layers can be nested.
- Nest is required in control block *(layer), control block cross programming is not allowed.

∗ The script from start to end of IF~END IF or FOR~NEXT control statements, etc. is called control block.

Point **In case of multi-nesting, the program will be difficult to be understood. Besides, scanning time will be also increased, so multi-nesting shall be avoided as possible.**

Point

Programming cannot be made under the following loop control statement or conditional branch or loop control statement and conditional branch statement cross.

[[FOR statement (loop control statement) and WHILE statement (loop control statement) cross]

```
FOR <value of word device (1)> TO <end value>                'Control statement (1)
    Processing statement (1)
    FOR <value of word device (2)> TO <end value>            'Control statement (2)
        Processing statement (2)
        WHILE <condition equation >                 'Control statement (3)
          Processing statement (3)
    NEXT
        END WHILE
NEXT
```

Control Block Cross

[[FOR statement (loop control statement) and IF statement (conditional branch statements) cross]

```
FOR <value of word device (1)> TO <end value>                'Control statement (1)
    Processing statement (1)
    FOR <value of word device (2)> TO <end value>            'Control statement (2)
        Processing statement (2)
        IF <conditional equation> THEN              'Conditional branch statement
            Processing statement (3)
    NEXT
        END IF
NEXT
```

Control Block Cross

■Actual nesting actions

Establish the following processing programs.

例

The 3-layer nesting using loop control statement (FOR statement, WHILE statement, do statement) is produced to calculate the number of actual processing.

Programming Script

```
DM1000 = 0                          'Clear the counter for the number of
FOR DM2000 = 1 TO 10                'processing.
                                    'DM2000 loop s from 1 to 10
  '--- FOR loop :10 times
  DM3000 = 0                        'WHILE loop counter is reset
                                    'Loop when DM3000 is
  WHILE DM3000 < 10                 'less than 10
    '--- WHILE loop :10 times
    DM3000 += 1                     'WHILE loop counter plus 1
    DM4000 = 0                      'DO loop counter is reset
    DO
      '--- DOloop : 10 times
      'The most deep nesting
      DM4000 += 1                   'DO loop counter plus 1
      DM1000 += 1                   'Plus 1 to the counter for the number of
    UNTIL DM4000 >= 10              'processing
                                    'Until DM4000 exceeds 10
  END WHILE                         'Loop
NEXT
```

Operation description

DM2000 performs 10 FOR loops. Among which, DM 3000 performs 10 WHILE loops, DM 4000 performs 10 DO loops, and (the bottom layer) counts the number of processing (DM0 1000).

When executing the program, 10 FOR loop s x 10 WHILE loop s x 10 DO loops take place, execute 1000 times of the most deep nesting processing.

Loop control statement and processing time

With the loop control statement (FOR ~ NEXT, WHILE, DO), loop process programming can be easily performed by KV script. Loop processing, however, should be performed within the allowed scanning time range.

■About processing time

Here the scanning time means the scanning time for a program which includes multiple loops.
Example) sum 1+2+3+……200,000, 1 to 200,000.

Programming Script

```
DM1000.D = 0
DM1002.D = 0
WHILE DM1000.D < 200000          '(1) to (2) processing shall be repeat
                                   'for 200,000 times.
  DM1000.D += 1                  '①
  DM1002.D = DM1002.D + DM1000.D '②
END WHILE
```

For details please refer to "WHILE Statement (pre-judge loop control)", page 4-27.

Processing time of above Sample Program as follows.

1 loop time	→	approx 1.65μs
200,000 loop time	→	approx 1.65μs x 200,000=about 330,000μs (330ms)

* Above processing time is only a reference example. Actual processing varies with the number of expansion unit connected and the application environment.

In this case, there is no problem for 1 loop processing whereas for 200,000 times of loop processing, the total processing time is within allowed scanning time (over 300ms for KV-8000/7500/7300/5500/5000/3000/1000).

If the program is used in KV script, easy programming can be enabled by using smaller programs to program a complicated procedure and loop process. However, attention should be paid to the scanning time.

Reference

When Ladder exceeds allowed scanning time (KV-8000/7500/7300/5500/5000/3000/1000 or KV Nano Series reaches 300ms), scanning time overflow may occur (error No.30)

When the processing time of loop control statement is long (complicated operation processing and macro processing), scanning timeout will still occur, even if the number of loops is small.
To avoid scanning timeout, changes should be made to the program in order to execute several scanning (or scanning every time) processes.

Example) Calculation 1+2+3+……200,000, sum 1～200,000.
(avoid scanning time out in programming)

Programming Script

```
IF R1004 = OFF THEN
  SET(R1004)                          'Operation in progress
   DM1000.D = 0                       'DM1000.D initialization
   DM1002.D = 0                       'DM1002.D initialization
END IF
IF R1004 AND DM1000.D < 200000 THEN
   DM1000.D  += 1                     'Counter
   DM1002.D = DM1002.D + DM1000.D     'Sum 1 to 200,000
END IF
```

Operation description

If R1004 is OFF, 0 is then stored in DM1000.D and DM1002.D (initialize) to set up R1004.
If R1004 is ON and DM1000.D is less than 200,000, 1 is then added to the current value of DM1000.D, and the value from 1 to 200,000 is added in turn, and stored to DM1002.D (operation is executed in each time scanning).

For details please refer to "IF ~ Statement (Conditional Branch (1))", page 4-12.

■ About infinite loop

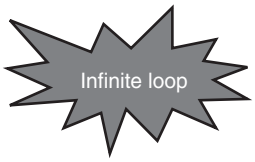
The situation whereby that loop process cannot ended is called infinite loop.

[inexecutable programming]

```
WHILE MR1000 = OFF
   DM1001 += 1
   DM1000 = DM1000 + DM1001
END WHILE
```

There is no programming for ending repeat control statement in progress statement. NO end since no programming to MR1000 become ON.

Add force end condition



Programming Script

```
WHILE MR1000 = OFF                    'If MR1000 is ON, loop process will be ended
   DM1001 += 1
   DM1000 = DM1000 + DM1001
   IF DM1000 > 1000 THEN
      BREAK
   END IF
END WHILE
```

Add end condition of repeat control statement.
Execute BREAK after DM1000 changed to 1000 above, end repeat control.

"BREAK Processing", page 4-31

Reference

You can program the BREAK statement for force termination of a loop control.
Alternatively, you can program MR1000 = ON to terminate the program. From the perspective of "improved program readability, it is recommended that the BREAK statement used in terminating a loop control.

■ About timer actions in a loop control statement

Timer cannot be used in force termination of loop control statement.
Timer, counter contact in loop control will not be changed, the following script becomes infinite loop.

[inexecutable Script Programming]

```
TMH(T001,1)                           '10ms timer
WHILE MR1000 = OFF                    'If MR1000 is ON, loop process will be ende
   DM1001 += 1
   DM1000 = DM1000 + DM1001
   IF T001.B THEN                     'BREAK is executed after the timer contact
      BREAK                           T001 is opened.
   END IF
END WHILE
```

Execute no processing statement because of without change to contact of timer.

"TMR function", page 5-22

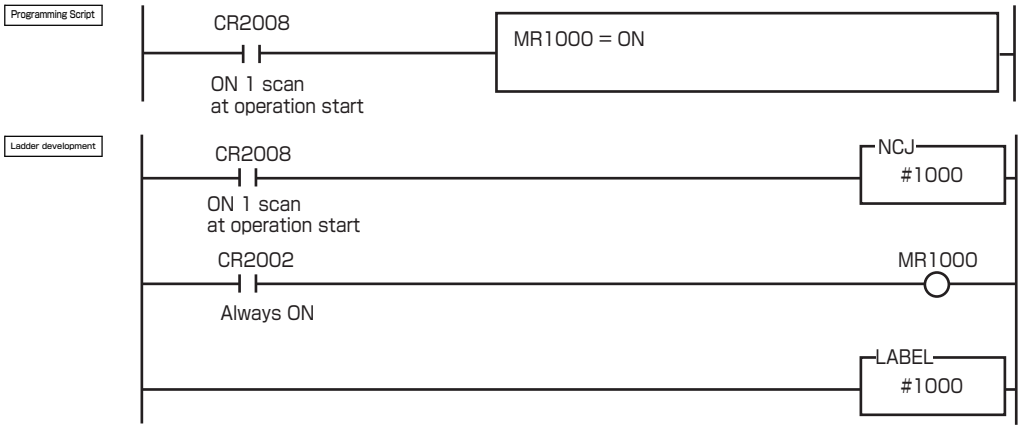
Precautions on Script Programming

Status of Hold Control Bit Device

In KV script, the program changing the status of the previous bit device can be conducted (assign ON / OFF).

Example)Turn MR1000 ON (assign ON).

[Box Script]



[Area Script]

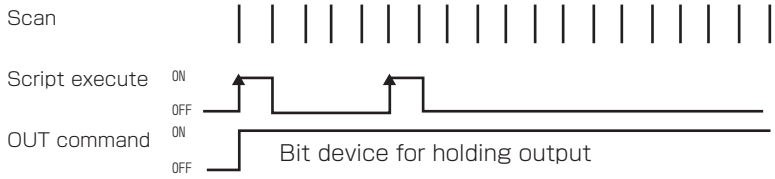
The MR1000=ON programed with KV script is converted to the OUT command taking normal ON (CR2002) as execution condition. So the script should be executed. After this, MR1000 is ON.

Reference

When you want to clear the value of the held bit data or word data, the following methods can be used.

- Reset by RES function (for bit data)
- Assign 0 with an assignment statement (for word data)
- Select the device from "Debug (D)" → "Clear all PLC devices (L)" in the menu and execute the clear operation.

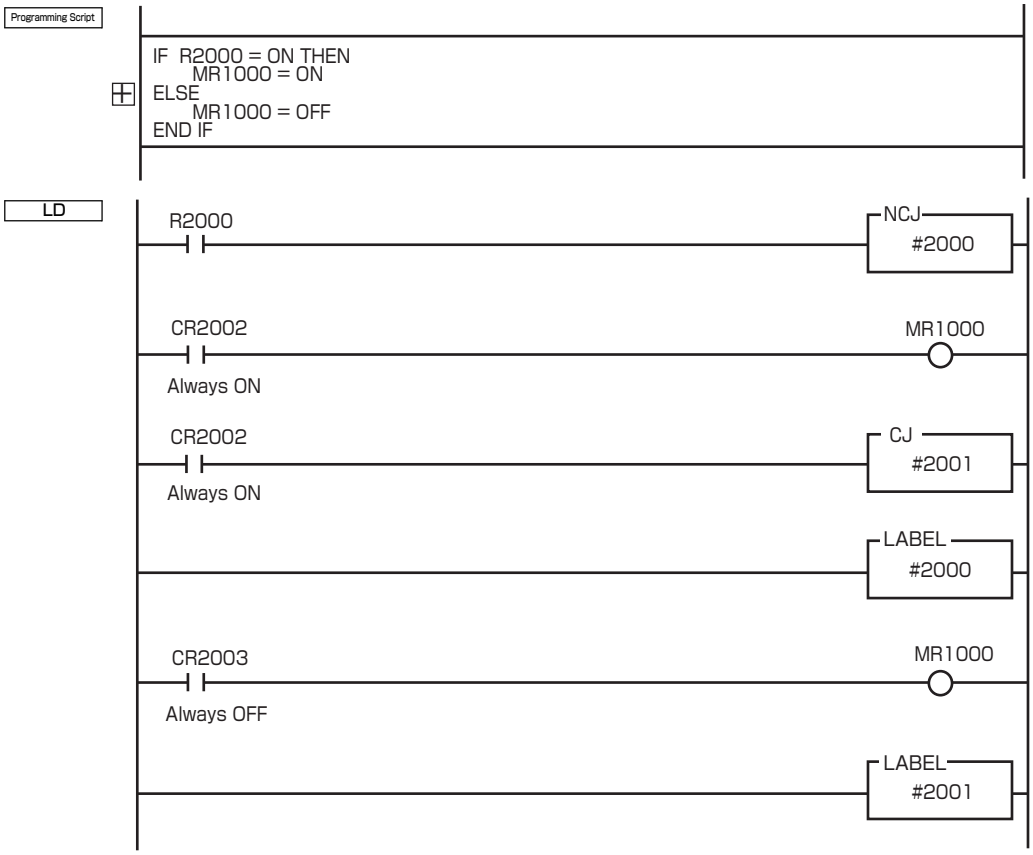
■ Execution timing



■ How to use

"MR1000 = ON" remains ON, so to turn it OFF, execute "MR1000 = OFF" or RES (MR1000).

Example)when R2000 (execute condition) is ON, MR1000 is turned to ON. When R2000 is not ON (OFF), MR1000 is turned to OFF.



Reference The status of the bit data can also be controlled by the SET / RES function.

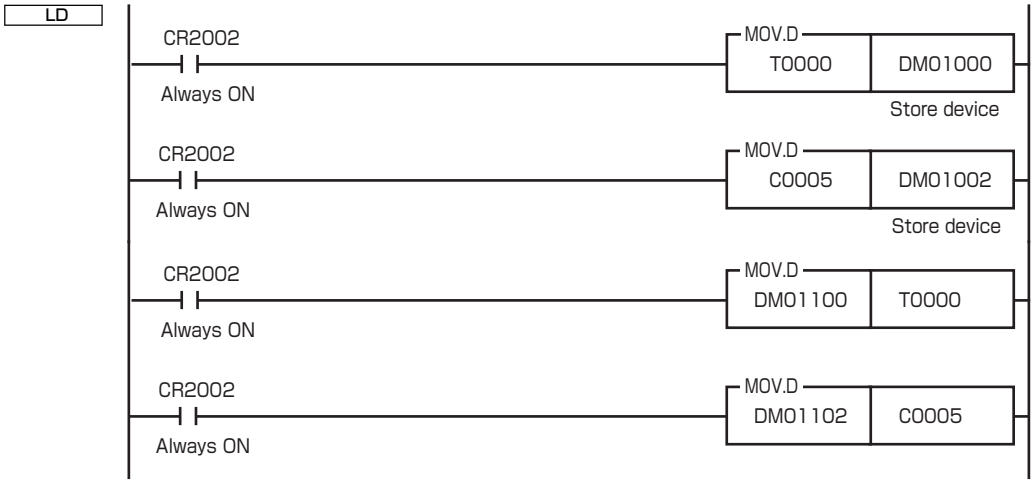
Precautions for using the timer / counter

Timer / counter processes the current value and contact status with the same device number.
The contact status can be processed as .B (bit type) like other bit data (.B suffix can also be omitted).
The current value can be processed as .D (32-bit unsigned) like other word data, but the .D suffix must be specified.

"Timer (contact) T", page 3-31
"Counter (contact) C", page 3-33

● Timer/ Counter current value reading

```
T0.D = DM1100.D     'Replace the current value of timer T0 with the value
                    'stored in DM1100.D.
C5.D = DM1102.D     'Replace the current value of counter C5 with the value
                    'stored in DM1102.D.
```



Point

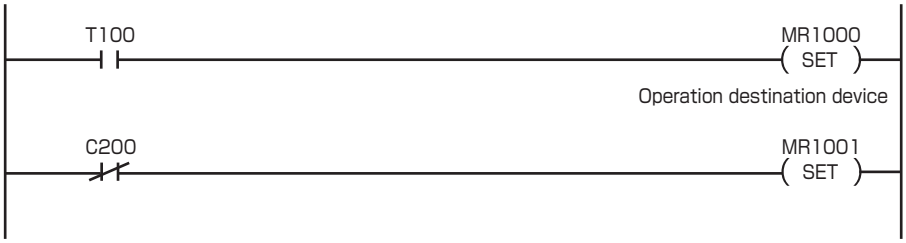
- **Current value of timer/counter is processed as 32 bit.**
- **When processing timer/counter as current value, shall specify suffix .D suffix, or describe the type.**
- **When the compatible model (CPU unit used) is KV-1000, the same device number cannot be used for the timer and counter.**
  Example) when T100(timer) has been used, C100 cannot be used (counter)

● Timer/ Counter contact

Programming Script

```
SET(T100.B, MR1000)        'When timer contact T 100 is ON, change
                           'MR1000 to ON (hold).
SET(C200.B = OFF, MR1001)  'When counter contact C 200 is OFF, MR1001
                           'changes to ON (hold).
```

LD



Reference The default type of timer / counter is bit type.

■ Precautions about the use of timer function

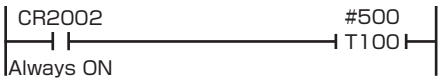
An execution condition is included in timer function which is also used as the reset instruction of counter. The execution condition of the timer function can omit programming, but it cannot be reset at this time, and continues to operate as a normal execution type function.

Programming Script

```
TMR(T100,500)
```

' Programming with execution condition omitted

Ladder development



Operation description Timer device T100 enables T100 contact ON 5 seconds after programmed start.
Only when RES (T100), or PLC power OFF or "RUN⇒PROG" conversion is converted, the contact can be cleared.)

■Precautions about use of timer function in Box Script and IF statement

Box Script and IF statement are converted to NCJ command to LABEL.
When execution condition of Box Script and IF statement is FALSE, jumping is performed between NCJ to LABEL. So care must be taken during use of timer function.

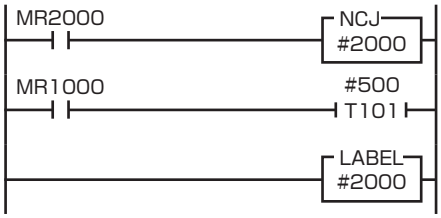
● Use timer function in IF statement of area script

The following points must be noted during use of area script, timer function in IF statement.

Programming Script

```
IF MR2000 = ON THEN                    ' Program in IF statement
    TMR (MR1000 = ON, T101, 500)
END IF
```

LD

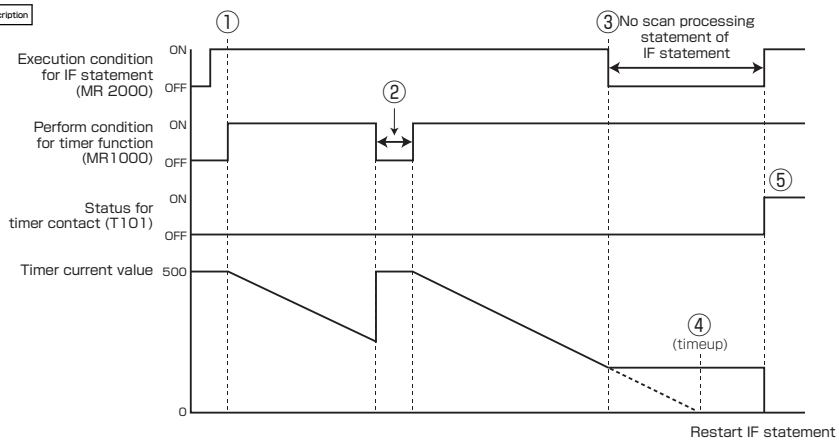


Notice

If condition execution of statement is FALSE, statement processed via N CJ command is jumped.In this case, keep the status of every contact(the status before N CJ jump), when execution condition of timer function is ON, timer is not reset.

The timer continues action before reset, processing statement jumps with N CJ command, so, even if timeout, timer contact changes to ON in PLC, will not output. In such status, when execution condition of IF statement is TRUE, processing statement is executed, so, any action without timer function, timer contact ON, output.

Operation description



(1) When execution condition of IF statement (MR2000) is ON, and when execution condition of timer (MR1000) is ON, timer T101 acts. timer contact T101 changes to ON 5 seconds after action starts.
(2) When execution condition of IF statement(MR2000) changes to ON, and execution condition of timer(MR1000) changes to OFF, the timer is reset.
(3) Timer action is unrelated to execution condition of timer before timeout, when execution condition of IF statement changes to OFF, processing statement(timer function) of IF statement is not scaned. In this case,, timer function seems in pause mode, but actually in continual action(due to no reset input).
(4) But after timeout, processing statement will be not scaned. So in stead of output, timer contact will be in standby status.
(5) During scan when execution condition of next IF statement is TRUE, timer contact changes to ON without timer action.

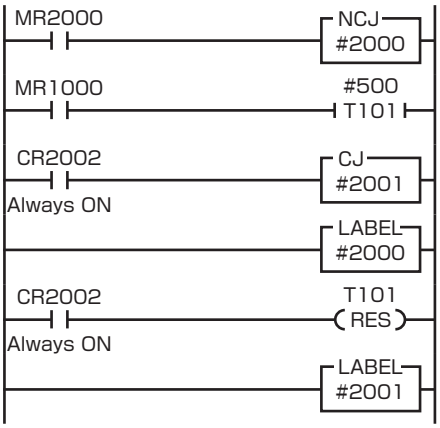
Precautions

When execution condition of IF statement is OFF, resetting timer function contained in processing statement can avoid this problem.
The change for programming script are as follows.*1

Programming Script

```
IF MR2000 = ON THEN
   TMR (MR1000 = ON, T101, 500)      ' Programming in IF statement
ELSE
   RES (T101)                        ' Reset timer device.
END IF
```

Ladder development



Operation description

When execution condition (MR2000) of IF statement is ON, and when execution condition of timer (MR1000) is ON, timer T101 acts.
Timer contact T101 changes to ON 5 seconds after action starts.

When execution condition (MR2000) of IF statement changes to OFF, reset timer device (T101).

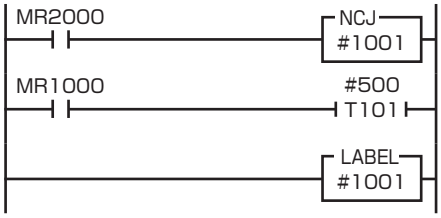
*1 hen programming area script including timer function between CJ-LABEL command, or programming in sub-program, timer function may fail to normal working, this fails to be avoided using above method.
For details, refer to the 📖 KV-8000 / 7000/5000/3000/1000 Series, KV Nano Series Instruction Reference Manual.

● Specify timer function using Box Script

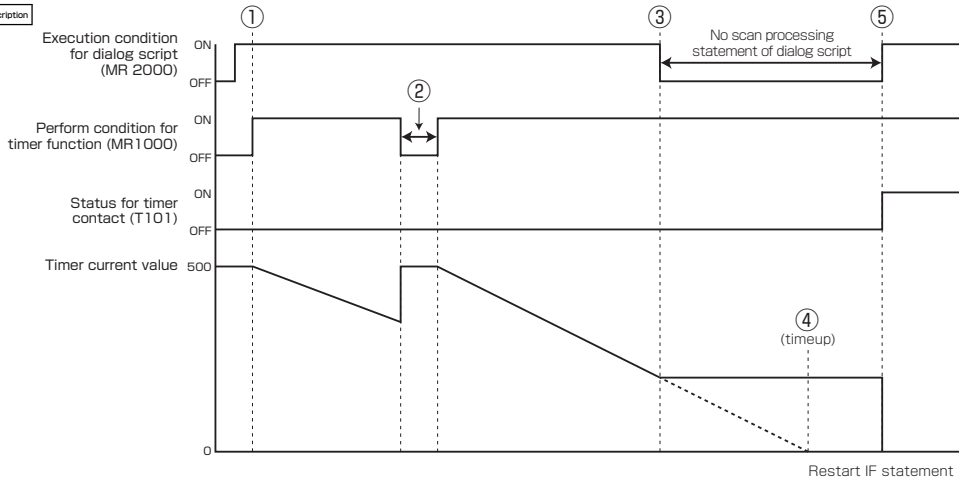
When using Box Script, care must be taken as same as timer function used in IF statement.

Programming Script

Ladder development



Operation description



(1) When execution condition of Box Script (MR2000) is ON, and when execution condition of timer (MR1000) is ON, timer T101 acts. timer contact T101 changes to ON 5 seconds after action starts.

(2) When execution condition of Box Script (MR2000) changes to ON, and execution condition of timer (MR1000) changes to OFF, and the timer is reset.

(3) Timer action is unrelated to execution condition of timer before timeout, when execution condition of Box Script changes to OFF, processing statement(timer function) of Box Script is not scaned. In this case, the timer function contines to work, although it seems that it is paused (cause no reset input).

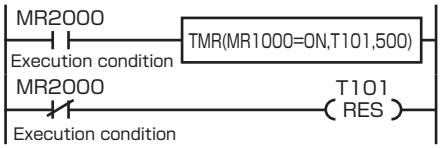
(4) But after setup timeout, processing statement will be not scaned, so, timer contact will be not output, and in standby status.

(5) During scan when execution condition of next Box Script is TRUE, timer contact changes to ON without timer action.

Precautions

When execution condition of Box Script is OFF, resetting timer function contained in Box Script can avoid this problem.
Script programming change as follows. *1

Programming Script



Operation description

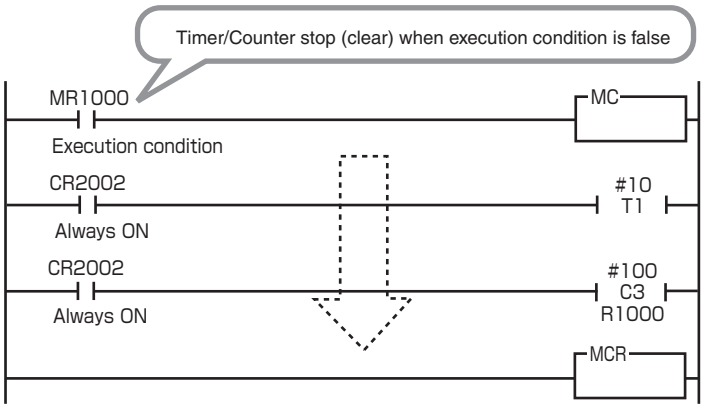
Reset timer device via contact B of execution condition of Box Script.

*1 When programming area script including timer function between CJ- LABEL, or programming in sub-routine, timer function may not work, and the above measures also do not work.
For details, refer to the 📖 KV-8000 / 7000/5000/3000/1000 Series, KV Nano Series Instruction Reference Manual.

● MC~MCR When using timer / counter function

```
        TMR(T1 , 10)
        CNT(C3 , 100 , R1000)
MCR
```

Ladder development



If condition is FALSE (MR1000 is OFF) OFF, the timer/counter is reset.

For details, see "MC ~ MCR Statement (Master control)", page 4-22.

MEMO

5

BASIC FUNCTIONS

Contact function 5-2
Bit contact function 5-8
Output function 5-12
Bit output function..... 5-14
Timer / Counter function..... 5-22
Direct I/O functions..... 5-36

LDP Rising edge differential of specified bit

Return value*1 = LDP (Rising edge detection target)

Load pulse

Argument/Return Value		Description	Available suffixes								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B*3*4	.T	#/$		
S	Rising edge detection target*2	Specifies the target to detect the rising edge.	—	—	—	—	—	—	.B	—	—	○	○
R	Return value	Specifies the target to operate.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work device/ work memory.

*2 When word data is specified as the rising edge detection target, the state of the LSB is detected.

*3 Direct I/O Relays (DR) can be specified when KV-8000/7500/7300/5500/5000/3000 or KV Nano Series is used.

*4 When the corresponding model is KV-8000 / 7500/7300/5500/5000/3000 and KV Nano series, the bit position of word data can be specified.

Description of Operation

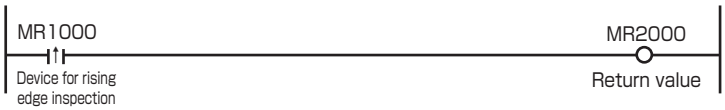
When the rising edge of S (change from OFF to ON) is detected, R is turned ON for one scan.

"About operation flag (CR2009 to CR2012)", page 1-18

For more information, please refer to "KV-8000 / 7000/5000/3000/1000 Series, KV Nano Series Instruction Reference Manual", "Contact Instruction".

● Example of format

MR2000 = **LDP** (MR1000)



Point **Attention is required when using this function in box scripts or IF statements.**

"Precautions When Using Box Script", page 2-13

Example program

● When using operation formulas

Programmed Script

MR2000 = **LDP** (DM1000 + DM1001 = 1000)

Operation Description

MR2000 is turned ON for one scan only when the total of DM1000 and DM1001 is 1000 or more (the operation result is FALSE ->TRUE).

● When combining with other functions

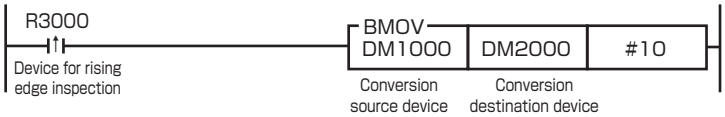
Programmed Script

BMOV (**LDP** (R3000), DM1000, DM2000, 10)

Operation Description

The content of ten words starting from DM1000 is block-moved to ten words starting from DM2000 only at the scan where the rising edge (OFF ->ON) of R3000 is detected.

Ladder conversion



Reference

Note that this function sometimes cannot be used depending on the argument of the function it is combined with.

"BMOV function", page 7-2

The KV script is converted to a ladder program before execution. When confirming the conversion content, please refer to "New Frame Script Step 5"(page 2-10).

LDF Falling edge differential of specified bit

Return value*1 = LDF (Falling edge detection target)

Load pulse fall

Argument/Return Value		Description	Available suffixes								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B*3*4	.T	#/$		
S	Falling edge detection target*2	Specifies the target to detect the falling edge.	—	—	—	—	—	—	.B	—	—	○	○
R	Return value	Specifies the target to operate.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work device/ work memory.

*2 When word data is specified as the falling edge detection target, the state of the LSB is detected.

*3 Direct I/O Relays (DR) can be specified when KV-8000/7500/7300/5500/5000/3000 or KV Nano Series is used.

*4 When the corresponding model is KV-8000 / 7500/7300/5500/5000/3000 and KV Nano series, the bit position of word data can be specified.

Description of Operation

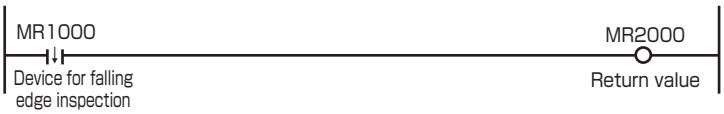
When the falling edge of S (change from ON to OFF) is detected, R is turned ON for one scan.

"About operation flag (CR2009 to CR2012)", page 1-18

For more information, please refer to "KV-8000 / 7000/5000/3000/1000 Series, KV Nano Series Instruction Reference Manual", "Contact Instruction".

● Example of format

MR2000 = **LDF** (MR1000)



Point **Attention is required when using this function in box scripts or IF statements.**

"Precautions When Using Box Script", page 2-13

Example program

● When using operation formulas

Programmed Script

MR2000 = **LDF** (DM1000 + DM1001 = 1000)

Operation Description

MR2000 is turned ON for one scan only when the total of DM1000 and DM1001 is 1000 or more (the operation result is TRUE → FALSE).

● When combining with other functions

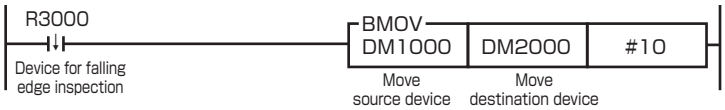
Programmed Script

BMOV (**LDF** (R3000), DM1000, DM2000, 10)

Operation Description

The content of ten words starting from DM1000 is block-moved to ten words starting from DM2000 only at the scan where the rising edge (ON→OFF) of R3000 is detected.

Ladder conversion



Reference

Note that this function sometimes cannot be used depending on the argument of the function it is combined with.

"BMOV function", page 7-2

The KV script is converted to a ladder program before execution. When confirming the conversion content, please refer to "New Frame Script Step 5"(page 2-10).

LDPB Rising edge differential bar of specified bit

Return value*1 = LDPB (Rising edge detection target)

Load pulse bar

Argument/Return Value		Description	Available suffixes								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B*3*4	.T	#/$		
S	Rising edge detection target*2	Specifies the target to detect the rising edge.	—	—	—	—	—	—	.B	—	—	○	○
R	Return value	Specifies the target to operate.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work relay.
*2 When word data is specified as the rising edge detection target, the state of the LSB is detected.
*3 Setable between direct I/O Relays (DR).
*4 The bit position of word data can be specified.

Point **The KV-1000 series does not support the LDPB function.**

Description of Operation

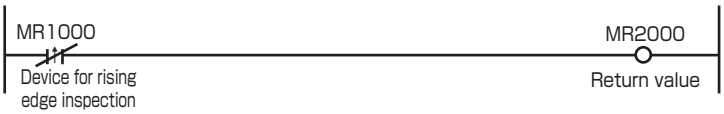
When the rising edge of S (change from OFF to ON) is detected, R is turned ON only within one scan.

"About operation flag (CR2009 to CR2012)", page 1-18

For more information, please refer to "KV-8000 / 7000/5000/3000/1000 Series, KV Nano Series Instruction Reference Manual", "Contact Instruction".

● Example of format

MR2000 = **LDPB** (MR1000)



Point **Attention is required when using this function in box scripts or IF statements.**
"Precautions When Using Box Script", page 2-13

LDFB Falling edge differential bar of specified bit

Return value*1 = LDFB (Falling edge detection target)

Load pulse bar

Argument/Return Value		Description	Available suffixes							Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.B*3*4	.T	#/$		
S	Falling edge detection target*2	Specifies the target to detect the falling edge.	—	—	—	—	—	.B	—	—	○	○
R	Return value	Specifies the target to operate.	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work relay.
*2 When word data is specified as the falling edge detection target, the state of the LSB is detected.
*3 Setable between direct I/O Relays (DR).
*4 The bit position of word data can be specified.

Point **The KV-1000 series does not support the LDFB function.**

Description of Operation

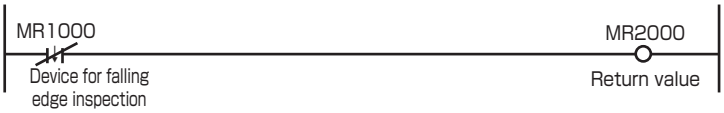
When the rising edge of S (change from OFF to ON) is detected, R is turned OFF for one scan.

"About operation flag (CR2009 to CR2012)", page 1-18

For more information, please refer to "KV-8000 / 7000/5000/3000/1000 Series, KV Nano Series Instruction Reference Manual", "Contact Instruction".

● Example of format

MR2000 = **LDFB** (MR1000)



Point **Attention is required when using this function in box scripts or IF statements.**

"Precautions When Using Box Script", page 2-13

BLD Specified bit information of word data

Return value*1 = BLD (Read target, bit position)

Bit load

Argument/Return Value		Description	Available suffixes								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B*3*4	.T	#/$		
S	Read target*2	Specifies the target to read the bit status.	.U	.U	.D	.D	—	—	—	—	—	○	○
n	Bit position	Specifies the bit position to determine the target ON/OFF. (16-bit binary data)	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	Specifies the target to operate.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work device/ work memory.

*2 When bit data is specified to S, the bit at the specified bit + n position is specified. Operation is normal even when relays straddle channels.

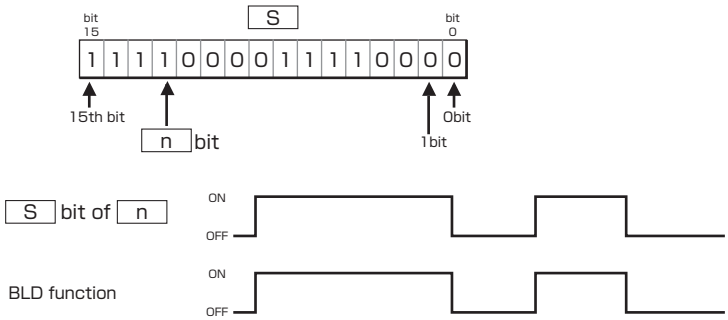
*3 The value of the bit position is specified as 0 to 15 when the suffix of the read target is .U, and 0 to 31 when it is .D.

*4 Direct I/O Relays (DR) can be specified when KV-8000/7500/7300/5500/5000/3000 or KV Nano Series is used.

*5 When the model is KV-8000 / 7500/7300/5500/5000/3000 and KV Nano series, the bit position of word data can be specified.

Description of Operation

The ON/OFF status of the bit position specified by n of S is transferred to R.



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000 / 7000/5000/3000/1000 Series, KV Nano Series Instruction Reference Manual", "Position Contact Instruction".

● Example of format

R1000 = **BLD** (DM1000, 10)

Point

When a value other than a constant is written in [n], if the operation result exceeds the range corresponding to the [S] suffix (0 to 15 for .U, 0 to 31 for .D), the result is as follows.

When the suffix of [S] is .U: The values of the lower 4 bits of [n] are used.

When the suffix of [S] is .D: The values of the lower 5 bits of [n] are used.

Example program

● When specifying bit data

Programmed Script

R1000 = **BLD** (MR2009.U, 10)

Operation Description

R1000 is turned ON when bit 10 (MR2103) of MR2009.U is ON.

● When combining with other functions

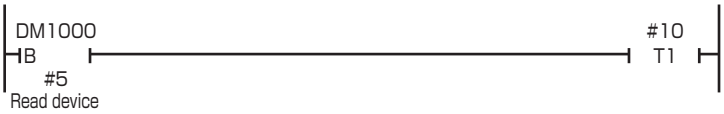
Programmed Script

TMR (**BLD** (DM1000, 5), T001, 10)

Operation Description

Timer (T1) is started when bit 5 of DM1000 is ON.

Ladder conversion



Reference

Note that this function sometimes cannot be used depending on the argument of the function it is combined with.

"TMR function", page 5-22

The KV script is converted to a ladder program before execution. When confirming the conversion content, please refer to "New Frame Script Step 5"(page 2-10).

BLDB Specified bit information bar of word data

Return value*1 = BLDB (Read target, bit position)

Bit load bar

Argument/Return Value		Description	Available suffixes								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B*3*4	.T	#/$		
S	Read target*2	Specifies the target to read the bit status.	.U	.U	.D	.D	—	—	—	—	—	○	○
n	Bit position	Specifies the bit position to determine the target ON/OFF. (16-bit binary data)	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	Specifies the target to operate.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored to a work device / work memory.

*2 When bit data is specified to S, the bit at the specified bit + n position is specified. Operation is normal even when relays straddle channels.

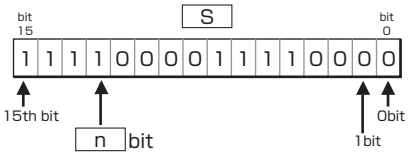
*3 The value of the bit position is specified as 0 to 15 when the suffix of the read target is .U, and 0 to 31 when it is .D.

*4 Direct I/O Relays (DR) can be specified when KV-8000/7500/7300/5500/5000/3000 or KV Nano Series is used.

*5 When the model is KV-8000 / 7500/7300/5500/5000/3000 and KV Nano series, the bit position of word data can be specified.

Description of Operation

The ON/OFF state of Bit n of S is inverted and transferred to R.



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000 / 7000/5000/3000/1000 Series, KV Nano Series Command Reference Manual", "Position Contact Command".

● Example of format

R1000 = **BLDB** (DM1000, 10)

Point

When a value other than a constant is written in [n], if the operation result exceeds the range corresponding to the [S] suffix (0 to 15 for .U, 0 to 31 for .D), the result is as follows.
When the suffix of [S] is .U: The values of the lower 4 bits of [n] are used.
When the suffix of [S] is .D: The values of the lower 5 bits of [n] are used.

Example program

● When the read object is specified as bit data

Programmed Script

R1000 = **BLDB** (MR2009.U, 10)

Operation Description

R1000 is turned ON when bit 10 (MR2103) of MR2009.U is OFF.

● When combining with other functions

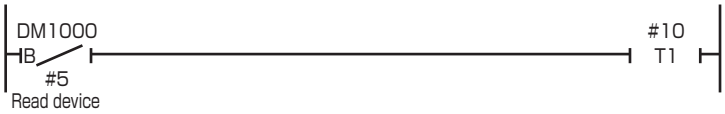
Programmed Script

TMR (**BLDB** (DM1000, 5), T1, 10)

Operation Description

Timer (T1) is started when bit 5 of DM1000 is OFF.

Ladder conversion



Reference

Note that this function sometimes cannot be used depending on the argument of the function it is combined with.

"TMR function", page 5-22

The KV script is converted to a ladder program before execution. When confirming the conversion content, please refer to "New Frame Script Step 5"(page 2-10).

SET Output ON of specified bit

Setting

SET ([execution condition,]*1 Destination device)

Argument/Return Value		Description	Applicable Suffix								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B*2*3	.T	#/$		
D	Object	Specifies the target to be held after ON.	—	—	—	—	—	—	.B	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 Direct I/O Relays (DR) can be specified when PCL KV-8000/7500/7300/5500/5000/3000 or KV Nano Series is used.

*3 Devices for bit position of word device can be specified when KV-8000/7500/7300/5500/5000/3000 or KV Nano Series is used.

Description of Operation

When the execution condition is TRUE, D is turned ON. Once the destination device is turned ON, it is held in the ON state even if the execution condition becomes FALSE. (The ON target is turned OFF with the RES function)

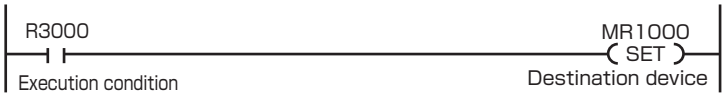
"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see KV-5500/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Output Instructions."

"RES function", page 5-13

● Example of format

SET (R3000, MR1000)



Sample Program

● When specifying a word device to the destination device

Programmed Script

SET (MR1000, DM2000.B)

Operation Description

When MR1001 is ON, the LSB of DM2000 is set to OFF.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

RES Output OFF of specified bit

Reset

RES ([Execution condition,] *1Destination device)

Argument/Return Value		Description	Applicable Suffix								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B*2*3	.T	#/$		
D	Object	Specifies the target to be turned OFF.	—	—	—	—	—	—	.B	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 Direct I/O Relays (DR) can be specified when PCL KV-8000/7500/7300/5500/5000/3000 or KV Nano Series is used.

*3 Devices for bit position of word device can be specified when KV-8000/7500/7300/5500/5000/3000 or KV Nano Series is used.

Description of Operation

When the execution condition is TRUE, D is turned OFF. (Targets turned ON with the SET function are turned OFF with the RES function)

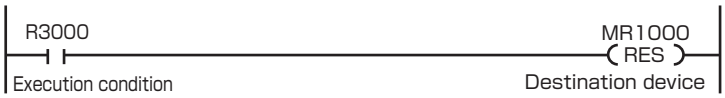
"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see KV-5500/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Output Instructions."

"SET function", page 5-12

● Example of format

RES (R3000, MR1000)



Sample Program

● When specifying a word device to the destination device

Programmed Script **RES** (MR1001, DM2000.B)

Operation Description When MR1001 is ON, the LSB of DM2000 is set to OFF.

● When resetting the timer

Programmed Script **RES** (MR1002, T100)

Operation Description When MR1002 is ON, the current value of T100 is changed to the same value as the setting value.

Reference The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

BOUT Specified bit output of word device

Bit output

BOUT ([Execution condition,][*1] Destination device, bit position)

Argument/Return Value		Description	Applicable Suffix								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Object[*2]	Specifies the target to output the input status.	.U	.U	.D	.D	—	—	—	—	—	○	—
n	bit position[*3]	Specifies the bit position to output.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

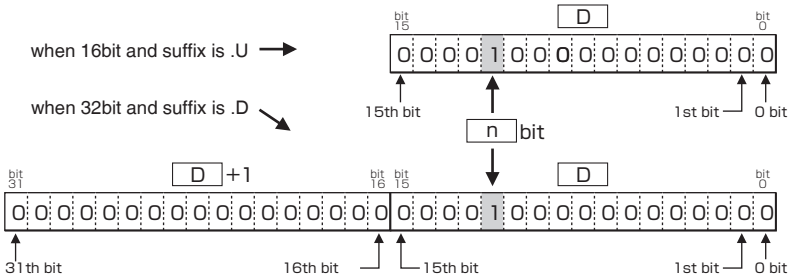
*1 "[ ]" can be omitted. (When the execution condition has been omitted, CR2002 (always ON) is specified.)

*2 When a bit device has been specified by D, the relay of the device No. to be calculated by the value of the [device No.+ n] is specified. Operation is normal even when relays straddle channels.

*3 For the value of the bit position, specify 0 to 15 when the suffix of the destination device is .U, and 0 to 31 when the suffix is .D.

Description of Operation

When the execution condition is TRUE, Bit n of D is turned ON, and when the execution condition is FALSE, it is turned OFF.



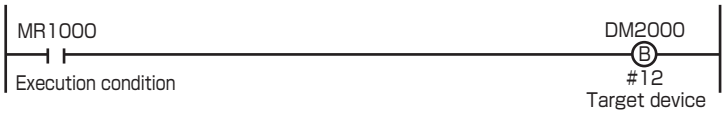
"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Bit Output Instructions."

● Example of format

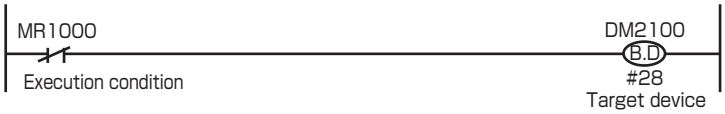
- D : 16-bit instruction

BOUT (MR1000, DM2000, 12)



- D : 32-bit instruction

BOUT (MR1000 = OFF, DM2100.D, 28)



Point

When programming a device or operation formula in [n], the operation result is as follows. When the operation result exceeds suffix range (0 to 15 for .U suffix and 0 to 31 for .D suffix) of [D]:
When the suffix of [D] is .U: The values of the lower 4 bits of [n] are used.
When the suffix of [D] is .D: The values of the lower 5 bits of [n] are used.

Sample Program

● When specifying a bit device to the destination device

Programmed Script

BOUT (DM1000 <= 100, MR2002.U, 14)

Operation Description

When DM1000 is 100 or less, bit 14 (MR2100) of MR2002.U is turned ON, and when DM1000 is greater than 100, bit 14 is turned OFF.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10).

BOUT
BOUB
BSET
BRES

BOUB Specified bit inverted output of word device

Bit out bar

BOUB ([Execution condition,] Destination device, bit position)*1

Argument/Return Value		Description	Applicable Suffix								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Object*2	Specifies the target with the input state inverted and output.	.U	.U	.D	.D	—	—	—	—	—	○	—
n	bit position*3	Specifies the bit position to output.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

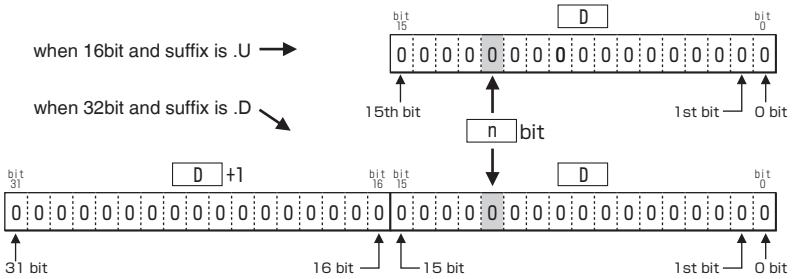
*1 "[ ]" can be omitted. (When the execution condition has been omitted, CR2002 (always ON) is specified.)

*2 When a bit device has been specified by D, the relay of the device No. to be calculated by the value of the [device No.+ n] is specified. Operation is normal even when relays straddle channels.

*3 For the value of the bit position, specify 0 to 15 when the suffix of the destination device is .U, and 0 to 31 when the suffix is .D.

Description of Operation

When the execution condition is TRUE, Bit n of D is turned OFF, and when the execution condition is FALSE, it is turned ON.



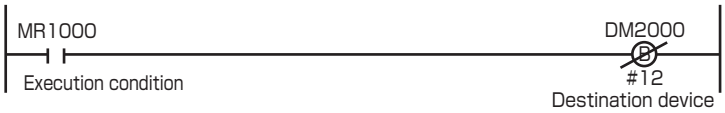
"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Bit Output Instructions."

● Example of format

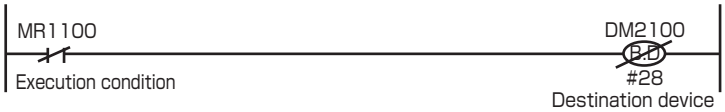
- D : 16-bit instruction

BOUB (MR1000, DM2000, 12)



- D : 32-bit instruction

BOUB (MR1100 = OFF, DM2100.D, 28)



Point

When programming a device or operation formula in [n], the operation result is as follows. When the operation result exceeds suffix range (0 to 15 for .U suffix and 0 to 31for .D suffix) of [D]:
When the suffix of [D] is .U: The values of the lower 4 bits of [n] are used.
When the suffix of [D] is .D: The values of the lower 5 bits of [n] are used.

Sample Program

● When specifying a bit device to the destination device

Programmed Script

BOUB (DM1000 <= 100, MR2002.U, 14)

Operation Description

When DM1000 is 100 or less, bit 14 (MR2100) of MR2002.U is turned OFF, and when DM1000 is greater than 100, bit 14 is turned ON.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10).

BSET Specified bit output ON of word device

BSET ([Execution condition,][*1] Destination device, bit position)

Set

Argument/Return Value		Description	Applicable Suffix								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Object[*2]	Specifies the target to be held after ON.	.U	.U	.D	.D	—	—	—	—	—	○	—
n	bit position[*3]	Specifies the bit position to be turned ON and held.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

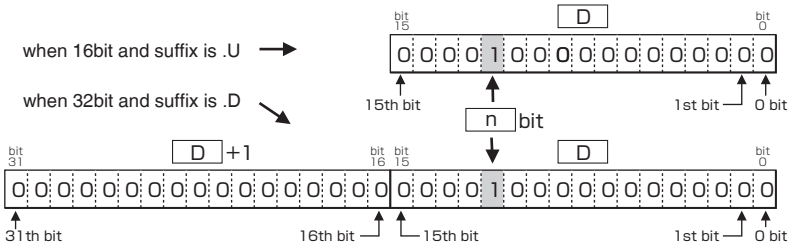
*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When a bit device has been specified by D, the relay of the device No. to be calculated by the value of the [device No.+ n] is specified. Operation is normal even when relays straddle channels.

*3 For the value of the bit position, specify 0 to 15 when the suffix of the destination device is .U, and 0 to 31 when the suffix is .D.

Description of Operation

When the execution condition is TRUE, Bit n of D is turned ON. Once the destination device is turned ON, it is held in the ON state even if the execution condition becomes FALSE.
(The ON target can be turned OFF with the BRES function)



"About operation flag (CR2009 to CR2012)", page 1-18

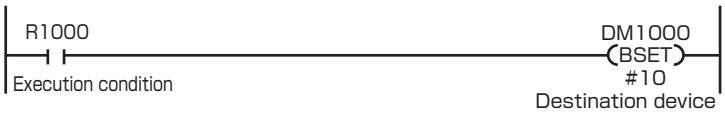
For details, please see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Bit Output Instructions."

"BRES function", page 5-20

● Example of format

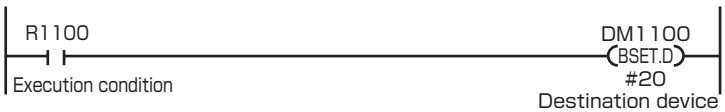
- D : 16-bit instruction

BSET (R1000, DM1000, 10)



- D : 32-bit instruction

BSET (R1100 = OFF, DM1100.D, 20)



Point

When programming a device or operation formula in [n], the operation result is as follows. When the operation result exceeds suffix range (0 to 15 for .U suffix and 0 to 31for .D suffix) of [D]:
When the suffix of [D] is .U: The values of the lower 4 bits of [n] are used.
When the suffix of [D] is .D: The values of the lower 5 bits of [n] are used.

Sample Program

● When specifying a bit device to the destination device

Programmed Script

BSET (DM1000 = 100, MR2008.U, 10)

Operation Description

When DM1000 is 100, bit 10 (MR2102) of MR2008.U is turned ON and is held in this state.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10).

BRES Specified bit output OFF of word device

Reset

BRES ([Execution condition,]*1 Destination device, bit position)

Argument/Return Value	Description	Applicable Suffix								Const	Device / Variable	For mula
		.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D Object*2	Specifies the target to be turned OFF.	.U	.U	.D	.D	—	—	—	—	—	○	—
n bit position*3	Bit position Specifies the bit position to turn OFF.	.U	.U	.U	.U	—	—	—	—	○	○	○
R Return value	None	—	—	—	—	—	—	—	—	—	—	—

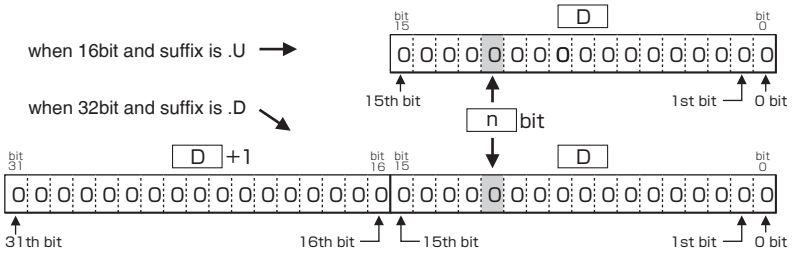
*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When a bit device has been specified by D, the relay of the device No. to be calculated by the value of the [device No.+ n] is specified. Operation is normal even when relays straddle channels.

*3 For the value of the bit position, specify 0 to 15 when the suffix of the destination device is .U, and 0 to 31 when the suffix is .D.

Description of Operation

When the execution condition is (TRUE), Bit n of D is turned OFF.



"About operation flag (CR2009 to CR2012)", page 1-18

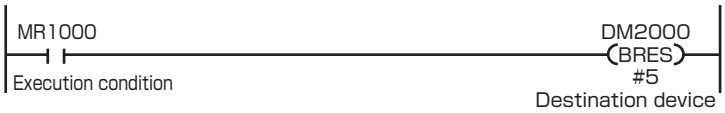
For details, please see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Bit Output Instructions."

"BSET function", page 5-18

● Example of format

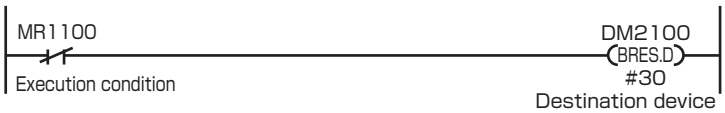
- D : 16-bit instruction

BRES (MR1000, DM2000, 5)



- D : 32-bit instruction

BRES (MR1100 = OFF, DM2100.D, 30)



Point

When programming a device or operation formula in [n], the operation result is as follows. When the operation result exceeds suffix range (0 to 15 for .U suffix and 0 to 31for .D suffix) of [D]:
When the suffix of [D] is .U: The values of the lower 4 bits of [n] are used.
When the suffix of [D] is .D: The values of the lower 5 bits of [n] are used.

Sample Program

● When specifying a bit device to the destination device

Programmed Script

BRES (DM1000 >= 100, MR2008.U, 10)

Operation Description

When DM1000 is 100 or more, bit 10 (MR2102) of MR2008.U is turned OFF.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10).

TMR 100 ms 32-bit down timer

Timer
TMR ([Timer execution condition,][*1] timer device, timer setting value)

Argument/Return Value		Description	Applicable Suffix								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Timer	Specifies the timer. (T0 to T3999[*3])	—	—	—	—	—	—	—	—	—	—	—
n	timer setting value[*2]	Specifies the setting value of the timer in 100 ms units. (0 to 4294967295)	.D	.D	.D	.D	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

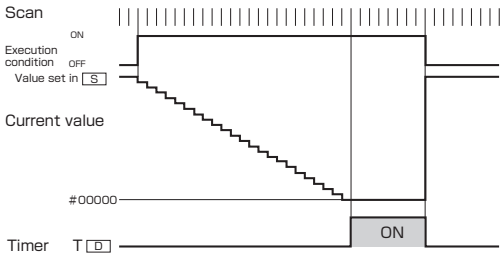
*1 • "[ ]" can be omitted. The execution condition can also be used as a reset. (When the execution condition is OFF, the TMR function is reset.)
• When the execution condition is omitted, use the RES function separately to reset the timer.

*2 When a bit device has been specified to the timer setting value, continuous 32 bits are occupied, and when a word device has been specified, continuous two words are occupied.

*3 For the KV Nano Series, device numbers T0 to T511 can be specified.

Description of Operation

When the execution condition is TRUE, the timer specified by D is ON. S becomes the current value, and is decremented by 1 every 100 ms. When the current value becomes 0, the specified timer (contact) turns ON. When the execution condition is FALSE, the specified timer is turned OFF and the current value is reset.

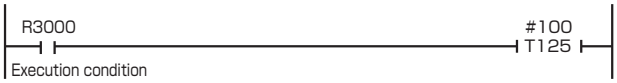


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Timer/Counter Instructions."

● Example of format

TMR (R3000, T125, 100)



Reference The error of the TMR instruction is "within ± (100 ms + 1 scan time)."

Point

- **When using KV-8000/7500/7300/5500/5000/3000 or KV Nano Series, the device No.identical with other TMR, TMH, TMS, TMU function (instruction), UDT instructions cannot be used.**
- **When using KV-1000, the device No. identical with other TMR, TMH, TMS, CNT function (instruction), OUTC, UDC, and UDT instructions can be used.**
- **Operation formulas can also be specified to setting values. Attention is required in the following instances as the range in which operation formulas can be handled is - 2147483648 to +2147483647.**
- **When the result of the operation formula is a minus value, the value obtained by adding 4294967295 to the result of the operation formula is set as the setting value.**
- **Attention is required when using this function in box scripts and programming this function in sub-routines.**

"Precautions for using the timer / counter", page 4-40

Reading and changing timer current values

Timer devices handle current values by specifying .D (unsigned 32-bit) as their suffix.

● Reading the current value

Specify the timer on the right side and the storage location for the current timer value read on the left side using an assignment statement.

Programmed Script DM1000.D = T125.D

Operation Description The current value of T125.D is rewritten to the value currently stored in DM1100.D.

● Rewriting the current value

Specify the storage location for the rewritten value on the right side and the timer on the left side using an assignment statement.

Programmed Script T125.D = DM1100.D

Operation Description The current value of T125.D is rewritten to the value currently stored in DM1100.D.

Point

- **When the current value is greater than the setting value when the current value is rewritten, the current value is changed to the same value as the setting value.**
- **As unsigned 32-bit data is used for the counter setting value and current value of the TMR function, unsigned 32-bit (.D) must be used for the devices to store the current value and setting value and the data to change.**
- **A target (32-bit unsigned) should be specified for the set value to change the set value of the timer or read the set value.**

KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Timer/Counter Instructions"

Reading Counter Contacts

Contacts are handled by specifying .B (bit type) as the suffix of the timer device. (.B can be omitted)

Programmed Script

MR1000 = T125.B

Operation Description

MR1000 is substituted with the contact state of T125.B.

Point

Timer contacts cannot be substituted with other bit information.
To forcibly reset a timer, reset the timer contact by the RES function.
After the timer suffix is omitted, the timer is processed as bit type.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10).

MEMO

TMH 10 ms 32-bit down timer

High-speed timer

TMH ([Timer execution condition,][*1] timer device, timer setting value)

Argument/Return Value		Description	Applicable Suffix .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	For mula
D	Timer	Specifies the timer. (T0 to T3999[*3])	—	—	—	—	—	—	—	—	—	—	—
n	timer setting value[*2]	Specifies the setting value of the timer in 10ms units. (0 to 4294967295)	.D	.D	.D	.D	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

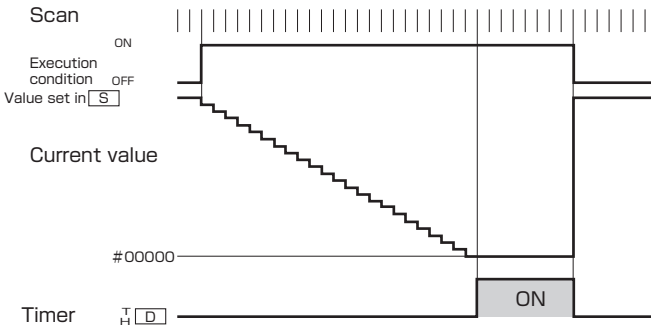
*1 • "[ ]" can be omitted. The execution condition can also be used as a reset. (When the execution condition is OFF, the TMH function is reset.)
• When the execution condition is omitted, use the RES function separately to reset the timer.

*2 When a bit device has been specified to the timer setting value, continuous 32 bits are occupied, and when a word device has been specified, continuous two words are occupied.

*3 For the KV Nano Series, device numbers T0 to T511 can be specified.

Description of Operation

When the execution condition is TRUE, the timer specified by n is ON. S becomes the current value, and is decremented by 1 every 10 ms. When the current value becomes 0, the specified timer (contact) turns ON. When the execution condition is FALSE, the specified timer is turned OFF and the current value is reset.

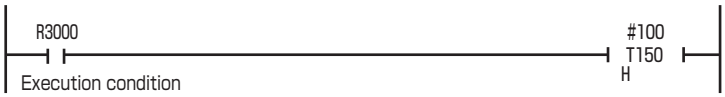


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Timer/Counter Instructions."

● Example of format

TMH (R3000, T150, 100)



Reference The error of the TMH instruction is "within ± (10ms + 1 scan time)."

Point

- **When using KV-8000/7500/7300/5500/5000/3000 or KV Nano Series, the device No.identical with other TMR, TMH, TMS, TMU function (instruction), UDT instructions cannot be used.**
- **When using KV-1000, the device No. identical with other TMR, TMH, TMS, CNT function (instruction), OUTC, UDC, and UDT instructions can be used.**
- **Operation formulas can also be specified to setting values. Attention is required in the following instances as the range in which operation formulas can be handled is - 2147483648 to +2147483647.**
  - When the result of the operation formula is a minus value, the value obtained by adding 4294967295 to the result of the operation formula is set as the setting value.
- **Attention is required when using this function in box scripts and programming this function in sub-routines.**
  "Precautions for using the timer / counter", page 4-40

"TMR function", page 5-22

Sample Program

● When using operation formulas for the setting value

Programmed Script

TMH (R1000, T150, DM1000 + 100)

Operation Description

High-speed timer T150 is activated for the duration that the execution condition R1000 is TRUE. When the execution condition is FALSE, the timer is stopped and the current value is reset.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10).

TMS 1 ms 32-bit down timer

High-speed 1ms timer

TMS ([Timer execution condition,][*1] timer device, timer setting value)

Argument/Return Value		Description	Applicable Suffix								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Timer	Specifies the timer. (T0 to T3999[*3])	—	—	—	—	—	—	—	—	—	—	—
n	timer setting value[*2]	Specifies the setting value of the timer in 1ms units. (0 to 4294967295)	.D	.D	.D	.D	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

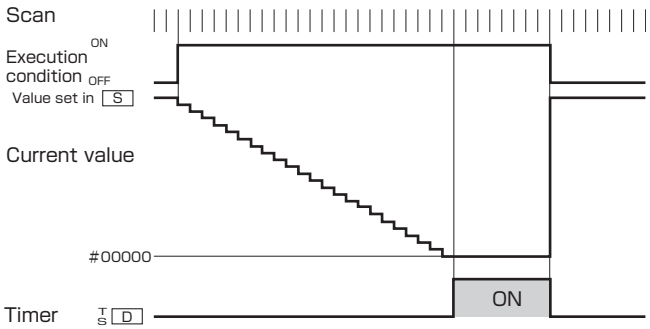
*1 • "[ ]" can be omitted. The execution condition can also be used as a reset. (When the execution condition is OFF, the TMS function is reset.)
• When the execution condition is omitted, use the RES function separately to reset the timer.

*2 When a bit device has been specified to the timer setting value, continuous 32 bits are occupied, and when a word device has been specified, continuous two words are occupied.

*3 For the KV Nano Series, device numbers T0 to T511 can be specified.

Description of Operation

When the execution condition is TRUE, the timer specified by D is ON. S becomes the current value, and is decremented by 1 every 1ms. When the current value becomes 0, the specified timer (contact) turns ON. When the execution condition is FALSE, the specified timer is turned OFF and the current value is reset.

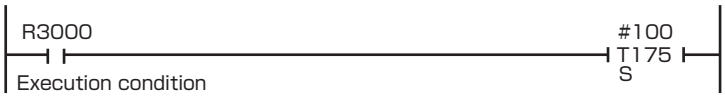


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Timer/Counter Instructions."

● Example of format

TMS (R3000, T175, 100)



Reference The error of the TMS instruction is "within ± (1ms + 1 scan time)."

Point

- **When using KV-8000/7500/7300/5500/5000/3000 or KV Nano Series, the device No. identical with other TMR, TMH, TMS, TMU function (instruction), UDT instructions cannot be used.**
- **When using KV-1000, the device No. identical with other TMR, TMH, TMS, CNT function (instruction), OUTC, UDC, and UDT instructions can be used.**
- **Operation formulas can also be specified to setting values. Attention is required in the following instances as the range in which operation formulas can be handled is - 2147483648 to +2147483647.**
  - When the result of the operation formula is a minus value, the value obtained by adding 4294967295 to the result of the operation formula is set as the setting value.
- **Attention is required when using this function in box scripts and programming this function in sub-routines.**

"Precautions for using the timer / counter", page 4-40

"TMR function", page 5-22

TMU 10µs 32-bit down timer

High-speed 10µs timer

TMU ([Timer execution condition,][*1] timer device, timer setting value)

Argument/Return Value		Description	Applicable Suffix .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	For mula
D	Timer	Specifies the timer. (T0 to T3999[*3])	—	—	—	—	—	—	—	—	—	—	—
n	timer setting value[*2]	Specifies the timer setting value in 1 µs. (0 to 4294967295)	.D	.D	.D	.D	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 • "[ ]" can be omitted. The execution condition can also be used as a reset. (When the execution condition is OFF, the TMU function is reset.)
• When the execution condition is omitted, use the RES function separately to reset the timer.

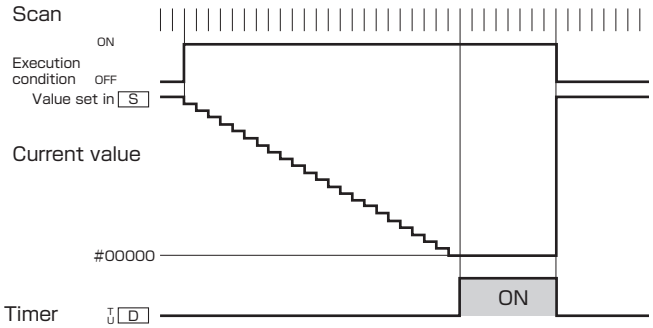
*2 When a bit device has been specified to the timer setting value, continuous 32 bits are occupied, and when a word device has been specified, continuous two words are occupied.

*3 For the KV Nano Series, device numbers T0 to T511 can be specified.

Point **The TMU function can only be used with KV-8000/7000 Series CPU units, with KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.**
"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

When the execution condition is TRUE, the timer specified by D is ON. S becomes the current value, and is decremented by 1 every 10µs. When the current value becomes 0, the specified timer (contact) turns ON. When the execution condition is FALSE, the specified timer is turned OFF and the current value is reset.

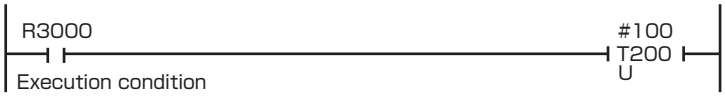


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Timer/Counter Instructions."

● Example of format

TMU (R3000, T200, 100)



Reference The error of the TMU instruction is "within ± (10 µ s + 1 scan time)."

Point

- **You cannot use the same device number as other TMR, TMH, TMS, TMU functions (instructions) and UDT instructions.**
- **Operation formulas can also be specified to setting values. Attention is required in the following instances as the range in which operation formulas can be handled is - 2147483648 to +2147483647.**
  - **When the result of the operation formula is a minus value, the value obtained by adding 4294967295 to the result of the operation formula is set as the setting value.**
- **Attention is required when using this function in box scripts and programming this function in sub-routines.**

"Precautions for using the timer / counter", page 4-40

"TMR function", page 5-22

CNT 32 bit up counter

Counter

CNT ([Counter execution condition,][*1] counter device, counter setting value, input device)

Argument/Return Value		Description	Applicable Suffix								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Counter	Specifies the counter. (C0 to C3999)[*3]	—	—	—	—	—	—	—	—	—	—	—
n	Counter Set value[*2]	Specifies the setting value of the counter. (0 to 4294967295)	.D	.D	.D	.D	—	—	—	—	○	○	○
S	Input target	Specifies the target to be used for count input.	—	—	—	—	—	—	.B	—	—	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

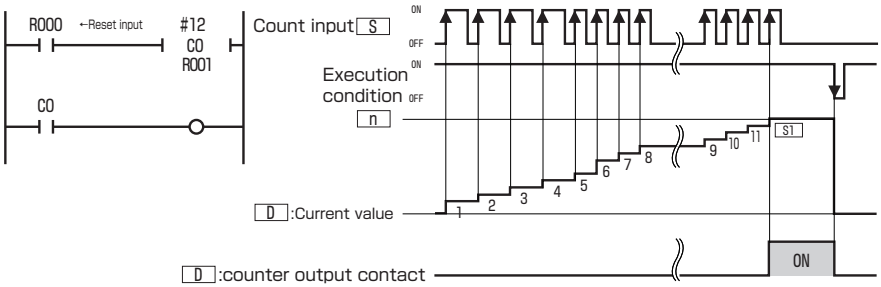
*1 • "[ ]" can be omitted. The execution condition can also be used as a reset. (When the execution condition is OFF, the CNT function is reset.)
• When the execution condition is omitted, use the RES function separately to reset the timer.

*2 When a bit device has been specified to the timer setting value, continuous 32 bits are occupied, and when a word device has been specified, continuous two words are occupied.

*3 For the KV Nano Series, device numbers C0 to C255 can be specified.

Description of Operation

When the counter execution condition is satisfied, the current value of the specified counter is added (+1) at the rising edge of the count input specified by S. When the current value reaches the set value specified by n, D turns ON. When the execution condition is not satisfied, the specified counter is turned OFF and the current value is reset.

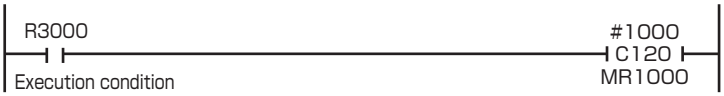


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Timer/Counter Instructions."

● Example of format

CNT (R3000, C120, 1000, MR1000)



Point

- **The device No. identical with instruction of OUTC and UDC cannot be used.**
- **When using KV-1000, the device No. identical with other TMR, TMH, TMS, CNT function (instruction), OUTC, UDC, and UDT instructions can be used.**
- **Operation formulas can also be specified to setting values. Attention is required in the following instances as the range in which operation formulas can be handled is - 2147483648 to +2147483647.**
  - **When the result of the operation formula is a minus value, the value obtained by adding 4247483648 to the result of the operation formula is set as the setting value.**

Reading and changing the current counter value

Counter devices handle current values by specifying .D (unsigned 32-bit) as their suffix.

● Reading the current value

Specify the counter on the right side and the storage location for the current counter value read on the left side using an assignment statement.

Programmed Script DM1000.D = C100.D

Operation Description The current value of C100.D is stored to DM1000.D.

● Rewriting the current value

Specify the storage location for the current value of the counter read on the right side and the counter on the left side using an assignment statement.

Programmed Script C100.D = DM1100.D

Operation Description The current value of C100.D is rewritten to the value currently stored to DM1100.D.

Point

- **When the current value is greater than the setting value when the current value is rewritten, the current value is changed to the same value as the setting value.**
- **As unsigned 32-bit data is used for the counter setting value and current value of the CNT function, unsigned 32-bit (.D) must be used for the devices to store the current value and setting value and the data to change.**
- **A target (32-bit unsigned) should be specified for the set value to change the set value of the counter or read the set value.**
  KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Timer/Counter Instructions"

Reading Counter Contacts

Contacts are handled by specifying .B (bit type) as the suffix of the counter device.(.B can be omitted) (.B can be omitted)

Programmed Script

MR1000 = C100.B

Operation Description

MR1000 is substituted with the contact state of C100.B.

Point

- **Counter contacts cannot be substituted with other bit information.**
  To forcibly reset a counter, reset the counter contact by the RES function.
- **The counter is processed as bit type when its suffix is omitted.**

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10).

MEMO

RFSX Input refresh

Refresh X

RFSX ([Execution condition,][*1] Leading device for changing input, number of relays)

Argument/Return Value		Description	Applicable Suffix								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Input refresh start	Specifies the start of the input status to be refreshed.	—	—	—	—	—	—	.B	—	—	○	—
n	Number of relay	Specifies the number of relay points with input status to be refreshed. (1 to 256)[*2]	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 " [ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When device is specified, continuous 16 bits are processed.
When relays other than the leading relay (e.g. R002, R1012) of a channel are specified, 16 bits are processed straddling the next channel.
The number of relays can be specified within the range of 1 to 256.

Description of Operation

In general, this function can be used for executing END instruction in ladder, or when changing batch changing input by anything in 1 scan period.

When the execution condition is TRUE, status of n inputs of bit devices starting from D will be changed.

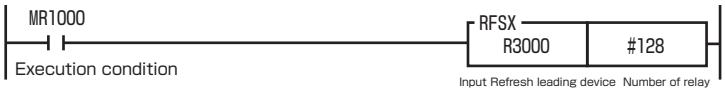
"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Direct I/O Instructions".

Point **For the KV Nano Series, this function cannot be used with relays assigned to expansion units.**

● Example of format

RFSX (MR1000, R3000, 128)



RFSY Output refresh

Refresh Y

RFSY ([Execution condition,][*1] Leading device for changing output, number of relays)

Argument/Return Value		Description	Applicable Suffix								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Output refresh start	Specifies the start of the output status to be refreshed.	—	—	—	—	—	—	.B	—	—	○	—
n	Number of relay	Specifies the number of relay points with output status to be refreshed. (1 to 256)[*2]	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 " [ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When device is specified, continuous 16 bits are processed.
When relays other than the leading relay (e.g. R002, R1012) of a channel are specified, 16 bits are processed straddling the next channel.
The number of relays can be specified within the range of 1 to 256.

Description of Operation

In general, this function can be used for executing END instruction in ladder, or when changing batch-changing output by anything in 1 scan period.

When the execution condition is TRUE, status of [n] inputs of bit devices starting from [D] will be changed.

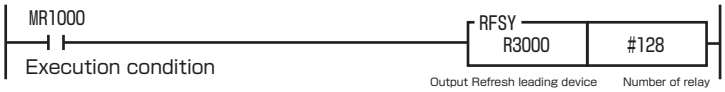
"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Direct I/O Instructions".

Point **For the KV Nano Series, this function cannot be used with relays assigned to expansion units.**

● Example of format

RFSY (MR1000, R3000, 128)



MEMO

6

APPLICATION FUNCTION

Flow Functions 6-2
Macro function 6-6
Function block function 6-10
Module function 6-18
Index register function 6-20
Specifying functions indirectly 6-26
File register function 6-36
Unit program function 6-42
Buffer memory function 6-52
Unit type judgment function 6-58
Array element number acquisition function 6-60
Word size get function 6-62

CALL Start specified sub-routine

Call

CALL ([Execution condition,][*1] Sub-routine No.)

Argument/Return Value		Description	Available suffixes								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
n	Sub-routine No.	Specifies the sub-routine No. to start up.（0 to 99）	—	—	—	—	—	—	—	—	*2 ○	—	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 Specify sub-routine No. within the range of 0 to 99.Sub-routine No. cannot be specified by $ (HEX).

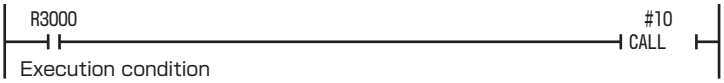
Description of Operation

When the execution condition is TRUE, the sub-routine specified by n is ON.

For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Flow Instructions."

● Example of format

CALL (R3000, 10)

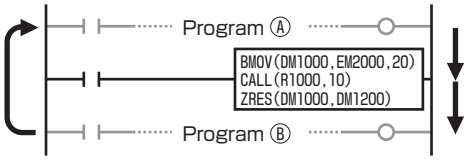


Reference The CALL function can be programmed down to a nesting level of 16.

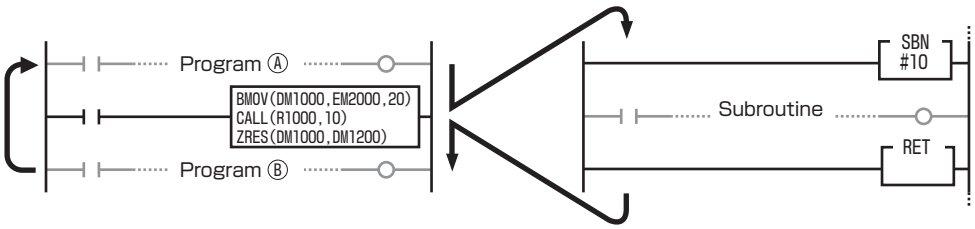
Point

- **Only constants can be specified as the sub-routine No. Devices or formulas cannot be used.**
- **Sub-routine No. constants cannot be specified by $ (HEX).**
- **The SBN and RET instructions are not available by script functions. Program them in the ladder.**

- When the CALL instruction is not executed (R1000=OFF) .



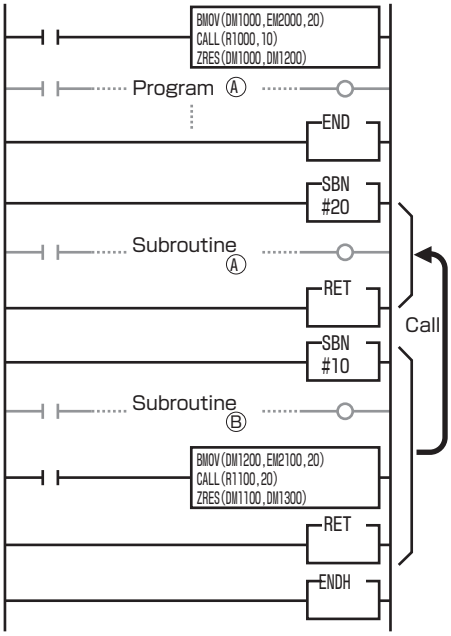
- When the CALL instruction is executed (R1000=ON)



Hint

■ About the sub-routine nesting structure

- Sub-routine functions can be programmed down to a nesting level of 16.
- To call a different sub-routine from inside a subroutine, the sub-routine to be called must already be defined (programmed in the above rung).



ECALL Module sub-routine instruction

Ecall
ECALL ([Execution condition,][*1] "Module name (string)")

Argument/Return Value		Description	Available suffixes								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Module name (Text string)	Specify module name with text string.									*2 ○		
n	Sub-routine No.	Specify the sub-routine No. to call.	—	—	—	—	—	—	—	—	*2 ○	—	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 Enter the macro name set on KV STUDIO enclosed by " ".

*3 Specify sub-routine No. within the range of 0 to 99.Sub-routine No. cannot be specified by $ (HEX).

Point **The KV-1000 series does not support the ECALL function.**

Description of Operation

When the execution condition is TRUE, Sub-routine [n] specified by Module [S] is executed.
All types of module (every scan, initialization, stand up, fixed period) can be specified except macro.
"About operation flag (CR2009 to CR2012)", page 1-18
For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Flow Instructions."

● Example of format

ECALL (MR1000, "Module 1", 1)

Point
- **Sub-routine No. constants cannot be specified by $ (HEX).**
- **The SBN and RET instructions are not available by script functions. Program them in the ladder.**
- **For the KV Nano Series, fixed period modules cannot be specified.**

MEMO

MCALL Start sub-routine type macro

Macro call

MCALL ([Execution condition,][*1] "Macro name (Text string) " [,Macro argument 1, Macro argument 2, Macro argument 3……, Macro argument 10])

Argument/Return Value		Description	Available suffixes								Const	Device/ Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Macro name (Text string)	Specifies the macro name by a text string.	—	—	—	—	—	—	—	—	*3○	—	—
S2	Macro argument*2*4	Specifies the macro argument. (The number of arguments differs according to the macro).	—	—	—	—	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 The number and attribute of macro argument devices differ according to the argument setting made to the macro to execute.
"KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "MCALL Instruction"

*3 Enter the macro name set on KV STUDIO enclosed by " ".

*4 When an ID No. is used, the format becomes "(macro name)_(ID No.)". For example) BatchSend_2
Please see "Macro" in the User's Manual of the CPU unit used for details.

Description of Operation

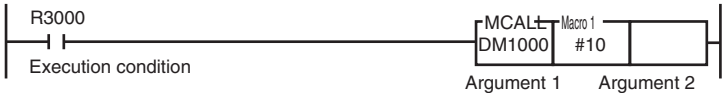
When the execution condition is TRUE, the sub-routine type macro specified by S1 is executed. The argument specified by S2 is handed over to the macro. With macros, arguments are handled as device No (P0 onwards) or constants (V0 onwards) according to the argument settings.

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Macro Instructions."

● Example of format

MCALL (R3000, "Marco1", DM1000, 10)



Point

- **Text strings other than sub-routine type macros enclosed by " " cannot be programmed as macro names.**
- **When the MCALL function is handled in box scripts or in IF statements, attention is required as OFF processing when stopped may operate differently to when executed with an instruction.**
  Please see "Macro" in the User's Manual of the CPU unit used for details.

MSTRT Start self-hold type macros

Macro start

MSTRT ([Execution condition,]*1 "Macro name (Text string) " [,Macro argument 1, Macro argument 2, Macro argument 3……, Macro argument 10] *2)

Argument/Return Value		Description	Available suffixes								Const	Device/ Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Macro name (Text string)	Specifies the macro name by a text string.	—	—	—	—	—	—	—	—	*4 ○	—	—
S2	Macro argument*3*5	Specifies the macro argument. (The number of arguments differs according to the macro).	—	—	—	—	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 The number of arguments differs according to the macro to execute.

*3 The number and attribute of macro argument devices differ according to the argument setting made to the macro to execute.
"KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "MSTRT Instructions"

*4 Enter the macro name set on KV STUDIO enclosed by " ".

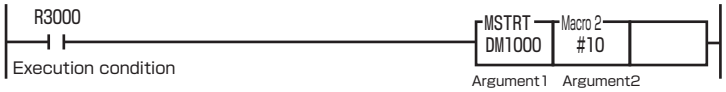
*5 When an ID No. is used, the format becomes "(macro name)_(ID No.)". For example) BatchSend_2
Please see "Macro" in the User's Manual of the CPU unit used for details.

Description of Operation

When the execution condition is TRUE, the self-hold type macro specified by S1 is executed.
The argument specified by S2 is handed over to the macro. With macros, arguments are handled as device No (P0 onwards) or constants (V0 onwards) according to the argument settings.
"About operation flag (CR2009 to CR2012)", page 1-18
For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Macro Instructions."

● Example of format

MSTRT (R3000, "Macro2", DM1000, 10)



Point

- **Text strings other than self-hold type macros enclosed by " " cannot be programmed as macro names.**
- **When the MSTRT function is handled in box scripts or in IF statements, attention is required as OFF processing when stopped may operate differently to when executed with an instruction.**
  Please see "Macro" in the User's Manual of the CPU unit used for details.

MEND End self-hold type macros

Macro end

MEND ([Execution condition][*1])

Argument/Return Value		Description	Available suffixes								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

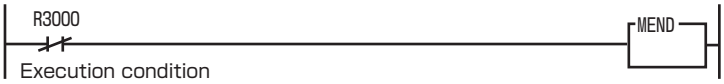
Description of Operation

When the execution condition is TRUE, self-hold type macros with a MEND function written to them stop.

For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Macro Instructions."

● Example of format

MEND (R3000 = OFF)



Reference "MSTRT function", page 6-7

Point

- **The MEND function can be used only when the content of a self-hold type macro is programmed by a script.**
- **At least one MEND function (instruction) must be programmed in self-hold type macros.**

MEMO

FB Start function block

Function block

FB ([Execution condition,] *1 "Function block name (Text string)"[,Argument 1, Argument 2, Argument 3……, Argument 64]*2)

Argument/Return Value		Description	Available suffixes								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Function block name (Text string)	Specifies the function block name in the text string.	—	—	—	—	—	—	—	—	*4 ○	—	—
S2	variable*3*5	Specifies the arguments. (The number of arguments differs according to the function block)	—	—	—	—	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 The number of arguments differs according to the function block to execute.

*3 The number and attributes of argument devices differ according to the argument setting configured in the function block to execute.
"KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Function Block Instructions"

*4 Enter the function block name set in KV STUDIO enclosed by " ".
Refer to "Function Block" in "KV-8000 Series User's Manual" and "KV-7000 Series User's Manual" for details.

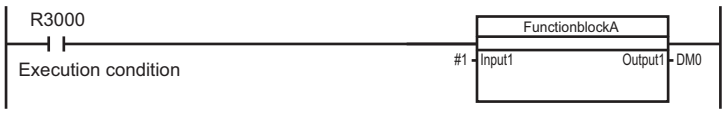
- FB functions can be used only by KV STUDIO ver.9.0 or later.
- The FB function can be used with KV-8000 series CPU units and KV-7000 Series CPU units that have CPU function version 2.0 or later only.
"KV-7500/7300 CPU Function Version", page 3

Description of Operation

When the execution condition is true (TRUE), the function block specified by S1 is activated.
Deliver the variable specified by S2 to the function block. The function block will process arguments according to the data type in the argument settings.
"About operation flag (CR2009 to CR2012)", page 1-18
For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Function Block Instructions"

● Example of format

FB (R3000, " Functionblock A", 1, DM0)



Point

- **Text strings other than function block names enclosed by " " cannot be written in function block name.**
- **When an FB function with the "OFF processing while execution is stopped" setting enabled is handled in box scripts or in IF statements, attention is required as OFF processing when stopped may operate differently to when executed with an instruction. Refer to "Function Block" in "KV-8000 Series User's Manual" and "KV-7000 Series User's Manual" for details.**

FBCALL Function block call

Function block call

FBCALL ([Execution condition,][*1] "Function name (Text string)", [,Argument 1, Argument 2, Argument 3……, Argument 64][*2])

Argument/Return Value		Description	Available suffixes								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Function block name (Text string)	Specifies the function block name in the text string.	—	—	—	—	—	—	—	—	*4 ○	—	—
S2	Function name (Text string)	Specifies the instance with a string.	—	—	—	—	—	—	—	—	*4 ○	—	—
S3	Independent variable*3	Specifies the arguments.	—	—	—	—	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 The number of arguments differs according to the function block to execute.

*3 It varies depending on the number of arguments, attributes, or the argument settings set in the function block to be executed.
"KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Function Block Instructions"

*4 Enter the function block name set in KV STUDIO enclosed by " ".

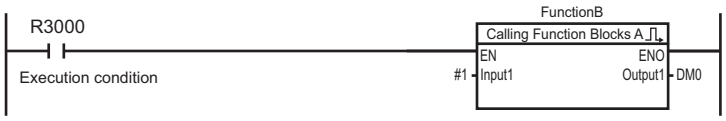
- The FBCALL function can be used only with KV STUDIO Ver.11.0 or later.
- The FBCALL function can be used only with KV-8000 with CPU function version 2.0 or later.
"KV-8000 CPU Function Version", page 2

Description of Operation

When the execution condition is true (TRUE), the instance specified by S2 of the function block specified by S1 is executed. When the execution condition is turned OFF, the operation of the instance stops. Deliver the variable specified by S3 to the function block. The function block will process arguments according to the data type in the argument settings.
For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Function Block Instructions"

● Example of format

FBCALL (R3000, "Function blockA", ExamplesB, #1, DM0)



Reference

Writing with named arguments eliminates the need to change the order in which the arguments are specified even if their order changes in the function block argument settings. Named arguments are written as follows.

FBCALL(R3000, "Function Block A", Examples B, Input1:=#1, OutPut1=>DM10, ENO=>MR000)

IN and IN-OUT arguments are specified with :=, while OUT arguments are specified with =>. ENO can be omitted if not used.

Unit arguments cannot be written in named arguments. The function block with a unit argument set should be written using a ladder program.

Point

- **Text strings other than function block names enclosed by " " cannot be written in function block name.**
- **The instance need not be surrounded by "".**
- **When an FB function with the "OFF processing while execution is stopped" setting enabled is handled in box scripts or in IF statements, attention is required as OFF processing when stopped may operate differently to when executed with an instruction.**
- **Please refer to "Function Blocks" in "KV-8000 Series User Manual"**

FBSTRT Function block start

Function block start

FBSTRT ([Execution condition,][*1] ""Function block name (Text string)", [,Argument 1, Argument 2, Argument 3……, Argument 64][*2])

Argument/Return Value		Description	Available suffixes								Const	Device/ Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Function block name (Text string)	Specifies the function block name in the text string.	—	—	—	—	—	—	—	—	*4 ○	—	—
S2	Function name (Text string)	Specifies the instance with a string.	—	—	—	—	—	—	—	—	○	—	—
S3	Independent variable*3	Specifies the arguments.	—	—	—	—	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 The number of arguments differs according to the function block to execute.

*3 It varies depending on the number of arguments, attributes, or the argument settings set in the function block to be executed.
"KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Function Block Instructions"

*4 Enter the function block name set in KV STUDIO enclosed by " ".

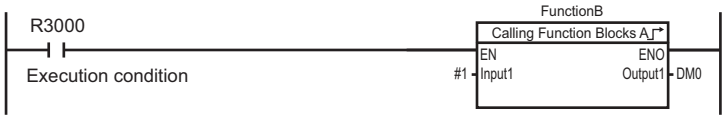
- The FBSTRT function can be used only with KV STUDIO Ver.11.0 or later.
- The FBSTRT function can be used only with KV-8000 with CPU function version 2.0 or later.
"KV-8000 CPU Function Version", page 2

Description of Operation

When the execution condition is true (TRUE), the instance specified by S2 of the function block specified by S1 is executed. The operation of the instance continues even if the execution condition is turned OFF. When the FEND instruction is executed in the function block, the operation stops.
Deliver the variable specified by S3 to the function block. The function block will process arguments according to the data type in the argument settings.
For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Function Block Instructions"

● Example of format

FBSTRT (R3000, "Function block A", ExamplesB, #1, DM0)



Reference

Writing with named arguments eliminates the need to change the order in which the arguments are specified even if their order changes in the function block argument settings. Named arguments are written as follows.

FBSTRT(R3000, "Function Block A", Examples B, Input1:=#1, OutPut1=>DM10, ENO=>MR000)

IN and IN-OUT arguments are specified with :=, while OUT arguments are specified with =>. ENO can be omitted if not used.

Unit arguments cannot be written in named arguments. The function block with a unit argument set should be written using a ladder program.

Point

- **Text strings other than function block names enclosed by " " cannot be written in function block name.**
- **The instance need not be surrounded by "".**
- **When an FB function with the "OFF processing while execution is stopped" setting enabled is handled in box scripts or in IF statements, attention is required as OFF processing when stopped may operate differently to when executed with an instruction.**
- **Please refer to "Function Blocks" in "KV-8000 Series User Manual"**

FEND End function block

Function block end

FEND ([Execution condition,]*1)

Argument/Return Value		Description	Available suffixes								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

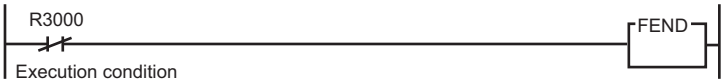
- FEND functions can be used only by KV STUDIO ver.9.0 or later.
- The FEND function can be used with the KV-8000 series CPU units and KV-7000 Series CPU units that have CPU function version 2.0 or later only.
  "KV-7500/7300 CPU Function Version", page 3

Description of Operation

When the execution condition is TRUE, function blocks with an FEND function written to them stop.
For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Function Block Instructions"

● Example of format

FEND (R3000 = OFF)



Reference "FBSTRT function", page 6-13

Point **FEND functions can only be used in function blocks.**

FUN Start function

Function

FUN ([Execution condition,]*1 "Function name (Text string)" [,Argument 1, Argument 2, Argument 3……, Argument 64])

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Function name (Text string)	Specifies the function name in the text string.	—	—	—	—	—	—	—	—	*3 ○	—	—
S2	Independent variable*2	Specifies the arguments. (The number of arguments differs according to the function)	—	—	—	—	—	—	—	—	○	○	○
R	Return value	The FUN function has the data format set when the argument is set.* 4	○	○	○	○	○	○	○	○	—	○	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 The number and attributes of argument devices differ according to the argument setting configured in the function to execute.
"KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "FUN instruction"

*3 Enter the function name set on KV STUDIO enclosed by " ".
Refer to "Function Block" in "KV-8000 Series User's Manual" and "KV-7000 Series User's Manual" for details.

*4 Only KV-8000 with CPU function version 2.0 or later has a return value. The value to be stored in the return value is specified by the program in the function.

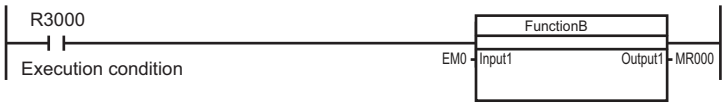
- FUN functions can be used only by KV STUDIO ver.9.0 or later.
- The FUN function can be used with the KV-8000 series CPU units and KV-7000 Series CPU units that have CPU function version 2.0 or later only.
"KV-7500/7300 CPU Function Version", page 3

Description of Operation

When the execution condition is true (TRUE), the function specified by S1 is executed.
Deliver the variable specified by S2 to the function. The function will process arguments according to the data type in the argument settings.
"About operation flag (CR2009 to CR2012)", page 1-18
For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Function Block Instructions"

● Example of format

FUN (R3000, "FeaturesB", EM0, MR000)



Reference

Refer to After describing by naming the independent variable, it is no longer necessary to change the designated order of the independent variable even when the order of the independent variable changes during the function's independent variable setting. Named arguments are written as follows.

FUN(R3000, "Function B", Input1:=EM0, OutPut1=>MR000, ENO=>MR100)

IN and IN-OUT arguments are specified with :=, while OUT arguments are specified with =>. ENO can be omitted if not used.

Unit arguments cannot be written in named arguments. The function block with a unit argument set should be written using a ladder program.

Point

- **The function name cannot be programmed to use anything other than the function name surrounded by " ".**
- **What can be described by naming arguments is limited to KV STUDIO Ver.11.0 and later, and the model is KV-8000 with CPU function version 2.0 and later.**

MDSTRT Module execution start

Module start

MDSTRT ([Execution condition,]*1 "Module name (Text string)")

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Module name (Text string)	Specify module name with text string.	—	—	—	—	—	—	—	—	*2 ○	—	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 Enter the macro name set on KV STUDIO enclosed by " ".

Point **MDSTRT function can be used only by KV STUDIO ver.4 or later.**

Description of Operation

When the execution condition is TRUE, Module S is executed.

The module in ON status operates from next scan period.

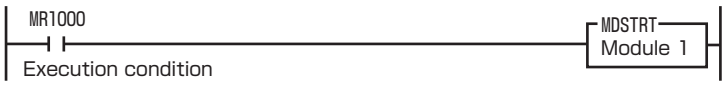
No operation is executed when the module is ON.

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Module Instructions."

● Example of format

MDSTRT (MR1000, "Module 1")



Point

- **As the default status at the start of the module execution, the differential execution type instruction turns ON when the rising edge is detected, and turns OFF when the falling edge is detected. The differential sign cannot be detected immediately in initial execution stage.**
- **When MDSTRT and MDSTOP function instruction are required to be executed for many times for the same module in the same scan period, the final module instruction is executed.**
- **Execution of initialization module cannot be stopped or started.**

MDSTOP Module execution stop

Module
MDSTOP ([Execution condition,] *1"Module name (Text string)")

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Module name (Text string)	Specify module name with text string.	—	—	—	—	—	—	—	—	*2 ○	—	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 Enter the macro name set on KV STUDIO enclosed by " ".

Point **MDSTOP function can be used only by KV STUDIO ver.4 or later.**

Description of Operation

When the execution condition is TRUE, Module S is executed.

No operation is executed when the module is ON.

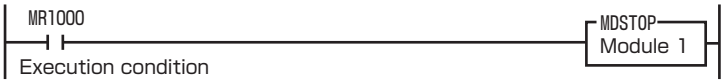
The instruction can stop its own module.

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Module Instructions."

● Example of format

MDSTOP (MR1000, "Module 1")



Point

- **When MDSTRT and MDSTOP function instruction are required to be executed for many times for the same module in the same scan period, the final module instruction is executed.**
- **Do not stop all modules. Under this circumstance, please restore by using any method. (Only after the program is stopped.)**
  - **Cut off the power supply of CPU units once, and power on again.**
  - **Toggle selector switch "RUN-PROG" of CPU units to "PROG" mode.**
  - **In KV STUDIO, turn RUN mode to OFF, then ON again.**
- **Execution of initialization module cannot be stopped or started.**

ZPUSH Batch purge of index register

Zpush

ZPUSH ([Execution condition,]*1 Purge destination device)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Purge destination device*2	Specifies the leading device of the purge destination data block.	*3 .S	*3 .S	*4 .L	*4 .L	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 T,C,CR,CTC,CTH,Z cannot be used.

*3 When the type of device for store destination is specified in .S:
- Continuous 13 words are occupied. When nesting functions, "number of register purges x 12 words + 1" words are occupied.
- When KV-8000/7500/7300/5500/5000/3000 or KV Nano series is used, only lower 16 bits are processed.

*4 When the type of device for store destination is specified in .L:
- Continuous 25 words are occupied. When nesting functions, "number of register purges x 24 words + 1" words are occupied.
- For KV-1000, it is processed as .S.

Point **For Z (index register), when KV-8000/7500/7300/5500/5000/3000 or KV Nano Series is used, it is processed as .L (signed 32-bit); for KV-1000, it is processed as .S (signed 16-bit).**

Description of Operation

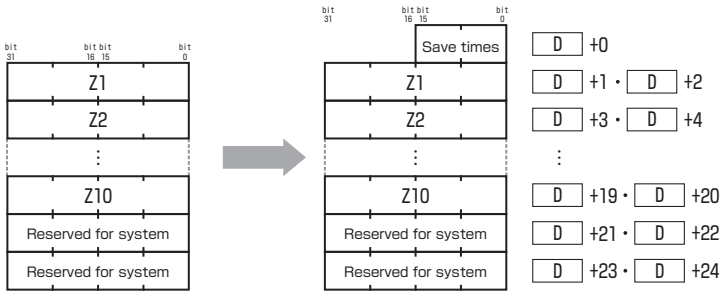
When the execution condition is TRUE, the values of Z1 to Z10 index registers are stored in the data block specified by D. After the function is executed, the value of [D+0]is incremented by 1.

Point
- **Before the execution of initial ZPUSH function,[D+0] is set to 0. This is not necessary from the second "ZPUSH function" onwards.**
- **The "ZPUSH function" and "ZPOP function" are used as a pair.**
- **Do not write to the range of D occupied by the "ZPUSH function" in other functions and instructions.**
- **When the data block range exceeds the device range, the function is not executed.**

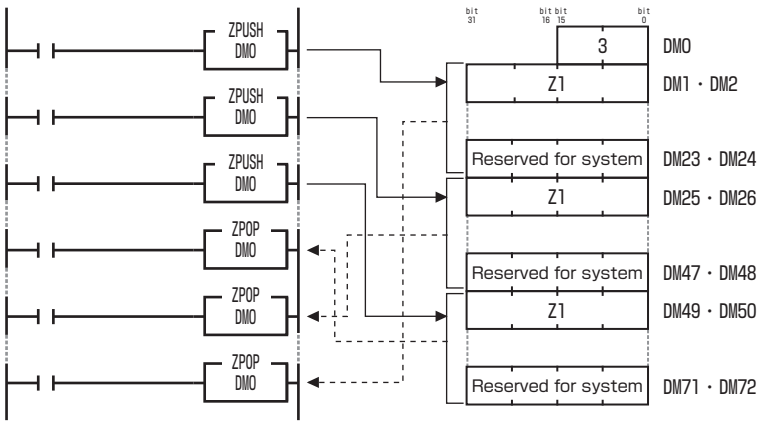
"About operation flag (CR2009 to CR2012)", page 1-18

■When using KV-8000/7500/7300/5500/5000/3000 or KV Nano series

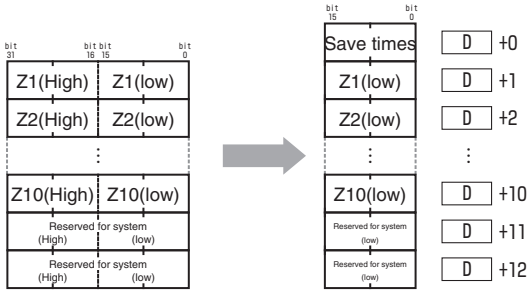
- D : 32-bit purge destination device



Reference "The "ZPUSH function" and ZPOP function" are programmed nested. When nesting, specify the same start.

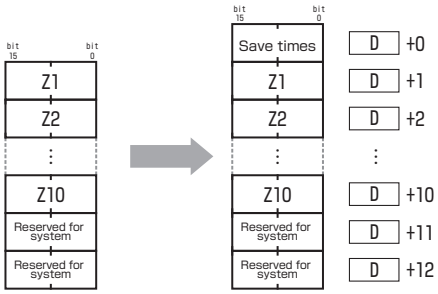


- D : 16-bit purge destination device



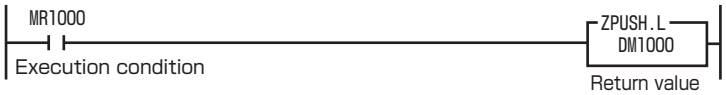
Point **For KV-8000/7500/7300/5500/5000/3000 or KV Nano Series, Z (index register) is processed as .L (signed 32-bit). When 16-bit type suffix (.S, .U) is specified in D, only 16 bits are stored in , and higher 16 bits are discarded.**
The stored value cannot be restored even if the ZPOP function (instruction) is used.
When the values used in program exceed .S (-32768 to +32767), please do not specify .S in D.

■When using KV-1000



● Example of format

ZPUSH (MR3000, DM1000.L)



For details, please see 📖"KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "3-8 Index Register Instructions."

MEMO

ZPOP Batch restore of index registers

ZPop

ZPOP ([Execution condition,]*1 Purge destination device)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Purge destination device*2	Specifies the leading device of the purge destination data block.	*3 .S	*3 .S	*4 .L	*4 .L	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 T,C,CR,CTC,CTH,Z cannot be used.

*3 When the type of device for store destination is specified in .S:
- Continuous 13 words are occupied. When nesting functions, "number of register purges x 12 words + 1" words are occupied.
- When KV-8000/7500/7300/5500/5000/3000 or KV Nano Series is used, only lower 16 bits are processed.

*4 When the type of device for store destination is specified in .L:
- Continuous 25 words are occupied. When nesting functions, "number of register purges x 24 words + 1" words are occupied.
- For KV-1000, it is processed as .S.

Point **For Z (index register), when KV-8000/7500/7300/5500/5000/3000 or KV Nano series is used, it is processed as .L (signed 32-bit); for KV-1000, it is processed as .S (signed 16- bit).**

Description of Operation

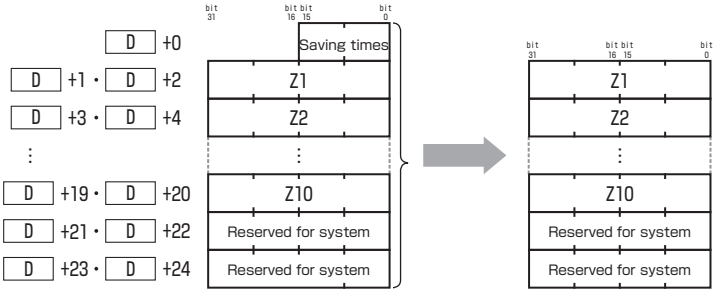
When the execution condition is TRUE, the values of Z1 to Z10 index registers are stored in the data block (index register stored in ZPUSH function) specified by D. After the function is executed, the value of [D+0]is incremented by -1.

Point
- **The "ZPUSH function" and "ZPOP function" are used as a pair.**
- **The function is not executed when either of the following conditions is established:**
  - **[D+0] is "0".**
  - **When the data block range exceeds the device range**

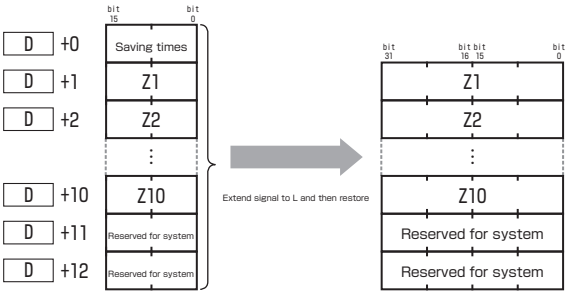
"About operation flag (CR2009 to CR2012)", page 1-18

■When using KV-8000/7500/7300/5500/5000/3000 or KV Nano series

- D : 32-bit purge destination device



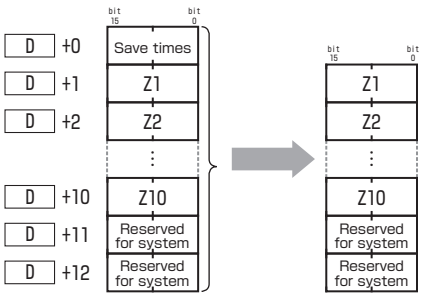
- D : 16-bit purge destination device



Point

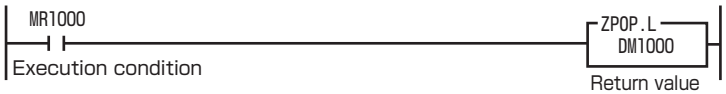
For KV-8000/7500/7300/5500/5000/3000 or KV Nano Series, Z (index register) is processed as .L (signed 32-bit). When 16-bit type suffix (.S, .U) is specified in D, only 16 bits are stored in , and higher 16 bits are discarded.
The stored value cannot be restored even if the ZPOP function (instruction) is used.
When the values used in program exceed .S (-32768 to +32767), please do not specify .S in D.

■When using KV-1000



● Example of format

ZPOP (MR1000, DM1000.L)



For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "3-8 Index Register Instructions."

ADRSET Address storage of indirectly specified target

Set address

ADRSET ([Execution condition,][*1] indirect specification device, destination store address)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Indirectly specified target	Specifies the target that is indirectly specified.	.U	.S	.D	.D	.F	*4 .DF	.B	.T	—	○	—
D	Destination store device*2 *3	Specifies the address storage location.	—	—	.D	.D	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

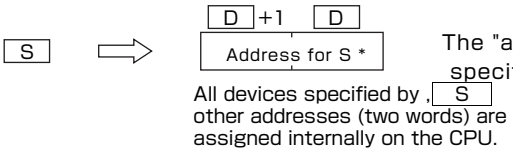
*2 Continuous two words are used.

*3 Unable to specify CM.

*4
- The .DF suffix can only be used when the compatible model is a KV-8000/7000 Series CPU unit, KV-5500/5000/3000 Series CPU unit that has a CPU function version 2.0 or higher and a KV Nano Series base unit.
- Z cannot be specified for the target of .DF suffix.

Description of Operation

When the execution condition is TRUE, the address of S is stored in continuous two words starting from D. After the addresses are stored, the result will be identical with that indirectly specified by when the function is programmed by indirect specification (* destination Store device).



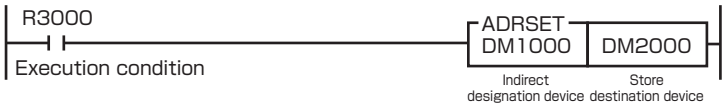
The "address" of S , rather than the values specified by S , is stored in [D・D+1].

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Indirect Specified Instructions."

● Example of format

ADRSET (R3000, DM1000, DM2000.D)



Point

- **The address storage location always occupies 2 continuous words starting with the target number.**
- **Before indirect specification (* destination Store device) is used, be sure to set the indirect specification device to the destination Store device by the ADRSET function.**
- **Be sure to use the ADRSET, ADRINC,ADRDEC,ADRADD and ADRSUB functions to generate and manipulate addresses.**

Sample Program

● When specifying a bit device to the indirect specification device

Programmed Script

ADRSET (R3000, MR1000, DM2000.D)

Operation Description

When R3000 is ON, the address of bit device MR1000 is stored in DM2000.D.
(one point lookup)

● When an index register is used

Programmed Script

ADRSET (DM1000:Z1, DM2000.D:Z1)

Operation Description

The address of the DM (1000+Z1) is stored in DM (2000+Z1) .D at every scan (no execution condition).

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10).

Hint

When the address of a bit device is used for indirect specification, the address is processed as a word device if the suffix is omitted. To process the address as a bit device, modify it by the .B suffix.

Programmed Script

ADRSET (R1000, DM1000.D)

MR2000 = LDP (*DM1000.B)

Operation Description

The address of R1000 is stored in DM1000.D.
*When the rising edge of *DM1000.B(R1000) is detected, MR2000 is turned ON for one scan.

Reference

"LDP function", page 5-2

ADRINC Add 1 to stored address

Increment address

ADRINC ([Execution condition,]*1 Address Store device)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Address Store device*2 *3	Specifies the storage location for the indirectly specified address.	—	—	.D	.D	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 Continuous two words are used.

*3 Unable to specify CM.

Description of Operation

When the execution condition is TRUE, the indirect specification address currently stored in D becomes the next address. When a word device address is currently stored, this indicates the address of the next word device, or when a bit device address is currently stored, this indicates the address of the next bit device.

Adds 1 to the address currently stored in D

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Indirect Specified Instructions."

● Example of format

ADRINC (R3000, DM2000.D)

Point

- **The address storage target always occupies 2 continuous words.**
- **Before using the ADRINC function, be sure to set the indirectly specified address to the address storage target using the ADRSET function.**
- **Be sure to use the ADRSET, ADRINC, ADRDEC, ADRADD, and ADRSUB functions to generate and manipulate addresses.**
- **The function is not executed when either of the following conditions is established:**
  - **D is inappropriate**
  - **The address is inappropriate after the increment.**

Sample Program

● When an index register is used

Programmed Script

ADRINC (R3000, DM2000.D:Z1)

Operation Description

When the R3000 is ON, the indirect specification address currently stored in DM(2000+Z1). D is the next address.*

*Before execution, the address must be stored in DM2000.D by the ADRSET function (instruction).

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10).

Hint

When the currently stored address is a bit device, the ADRINC function indicates the next bit device. When the address straddles channels at this time, the address becomes the address of the next channel.

Programmed Script

ADRSET (R3015, DM1000.D)

ADRINC (*DM1000.B = ON, DM1000.D)

SET (*DM1000.B)

Operation Description

The address of bit device R3015 is stored in DM1000.

*When DM1000.B(R3015) is ON, the address stored in *DM1000 is incremented by 1, and *DM1000.B(R3100) is set.

Reference

"ADRSET function", page 6-26

"SET function", page 5-12

ADRDEC Decrement stored address by 1

Decrement address

ADRDEC ([Execution condition,]*1 Address Store device)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Address Store device*2 *3	Specifies the storage location for the indirectly specified address.	—	—	.D	.D	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 Continuous two words are used.

*3 Unable to specify CM.

Description of Operation

When the execution condition is TRUE, the indirect specification address currently stored in D becomes the next address. When a word device address is currently stored, this indicates the address of the previous word device, or when a bit device address is currently stored, this indicates the address of the previous bit device.

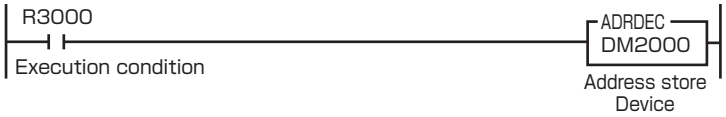
The address currently stored in D is subtracted by 1.

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Indirect Specified Instructions."

● Example of format

ADRDEC (R3000, DM2000.D)



Point

- **The address storage target always occupies 2 continuous words.**
- **Before the ADRDEC function is used, be sure to set the indirect specification address to the address Store device by the ADRDEC function.**
- **Be sure to use the ADRSET, ADRINC, ADRDEC, ADRADD, and ADRSUB functions to generate and manipulate addresses.**
- **The function is not executed when either of the following conditions is established:**
  - **D is inappropriate**
  - **The address is inappropriate after the decrement.**

Sample Program

● When an index register is used

Programmed Script

ADRDEC (R3000, DM2000.D:Z1)

Operation Description

When the R3000 is ON, the indirect specification address currently stored in DM(2000+Z1). D is made into the previous address.*

*Before execution, the address must be stored in DM2000.D by the ADRSET function (instruction).

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10).

Hint

When the currently stored address is a bit device, the ADRDEC function indicates the previous bit device. When the address straddles channels at this time, the address becomes the address of the previous channel.

Programmed Script

ADRSET (R3100, DM1000.D)

ADRDEC (*DM1000.B, DM1000.D)

SET (*DM1000.B)

Operation Description

The address of bit device R3100 is stored in DM1000.D.

*When DM1000.B(R3100) is ON, the address stored in DM1000.D is incremented by 1, and *DM1000.B(R3015) is set.

Reference

"ADRSET function", page 6-26

"SET function", page 5-12

ADRADD Add stored address

Add address

ADRADD ([Execution condition,]*1 Address add value, address store device)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Address add value	Specifies the address add data or the device to which the data is currently stored.	.S	.S	.L	.L	—	—	—	—	○	○	○
D	Address store device*2	Specifies the storage location for the indirectly specified address.	—	—	.D	.D	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

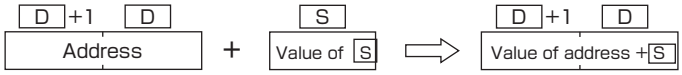
*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 Unable to specify CM.

Description of Operation

When the execution condition is TRUE, the indirect specification address currently stored in D is made into the address by adding S . When the address of the word device is currently stored, this indicates only address of the word device of address add value is added, or when the address of the bit device is currently stored, this indicates only bit address of the bit device is added.

S is subtracted from the address in D .

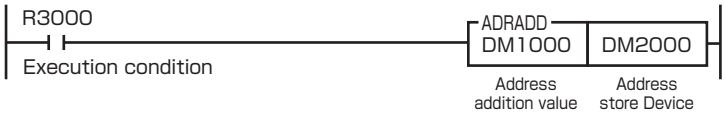


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Indirect Specified Instructions."

● Example of format

ADRADD (R3000, DM1000.S, DM2000.D)



Point

- **The address storage target always occupies 2 continuous words.**
- **Be sure to set the indirect designated address to the address storage object through the ADRSET function before use.**
- **Be sure to use the ADRSET, ADRINC, ADRDEC, ADRADD, and ADRSUB functions to generate and manipulate addresses.**
- **The function is not executed when either of the following conditions is established:**
  - **D is inappropriate**
  - **The address is inappropriate after the increment.**

Sample Program

When a device address is used for indirect specification, the indirect specification device (*DM1000, etc.) is processed as a 16-bit device. To handle the device as a 32-bit device, add the .D(.L) suffix.
The ADRADD function is handy when handling 32-bit data as it indicates the device No. obtained by adding the value specified by the address add value.

Programmed Script

```
ADRSET ( DM2000, DM1000.D )
*DM1000.D += 1
ADRADD ( 2, DM1000.D )
*DM1000.D += 1
```

Operation Description

The address of DM2000 is stored in DM1000.D
"1" is added to the stored address as a 32-bit device.
"2" is added to the address stored to DM1000.D, and "1" is added to the next 32-bit address (DM2002.D).

Reference

"ADRSET function", page 6-26

ADRSUB Subtract stored address

Subtract address

ADRSUB ([Execution condition,]*1 Address subtract value, address store device)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Address subtraction value	Specifies the address subtract data or the device to which the data is currently stored.	.S	.S	.L	.L	—	—	—	—	○	○	○
D	Address store device*2	Specifies the storage location for the indirectly specified address.	—	—	.D	.D	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

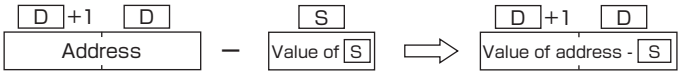
*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 Unable to specify CM.

Description of Operation

When the execution condition is TRUE, the indirect specification address currently stored in D is made into the address obtained by subtracting S. When the address of the word device is currently stored, this indicates the address of the word device that is obtained by subtracting the address subtract value, or when the address of the bit device is currently stored, this indicates the bit device obtained by subtracting the address subtract value.

S is subtracted from the address in D.

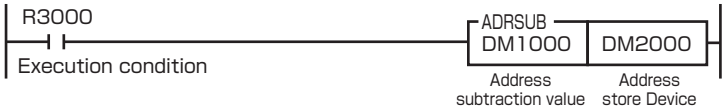


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Indirect Specified Instructions."

● Example of format

ADRSUB (R3000, DM1000.S, DM2000.D)



Point

- **The address storage target always occupies 2 continuous words.**
- **Before using, please be sure to set the indirect designated address to the address storage object through the ADRSUB function.**
- **Be sure to use the ADRSET, ADRINC, ADRDEC, ADRADD, and ADRSUB functions to generate and manipulate addresses.**
- **The function is not executed when either of the following conditions is established:**
  - **D is inappropriate**
  - **The address is inappropriate after the decrement.**

Sample Program

When a device address is used for indirect specification, the indirect specification device (*DM1000, etc.) is processed as a 16-bit device. To handle the device as a 32-bit device, add the .D(.L) suffix.
The ADRSUB function is handy when handling 32-bit data as it indicates the device No. obtained by adding the value specified by the address add value.

Programmed Script

```
ADRSET ( DM2000, DM1000.D )
*DM1000.D += 1
ADRSUB ( 2, DM1000.D )
*DM1000.D += 1
```

Operation Description

The address of DM2000 is stored in DM1000.D
"1" is added to the stored address as a 32-bit device.
"2" is subtracted from the address stored in DM1000.D, and "1" is added to the previous 32-bit address (DM1998.D).

Reference

"ADRSET function", page 6-26

FRSET Switch file register mode

Switch mode

FRSET ([Execution condition,][*1] Bank No.)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
n	File No.	Specifies current bank No. after switching. (0 to 15[*2])	.U	.U	.U	.U	—	—	—	—	[*2] ○	—	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 The scope of the bank number is 0 to 15 for KV-8000/7500/7300 and 0 to 3 for KV-5500/5000/3000.

Description of Operation

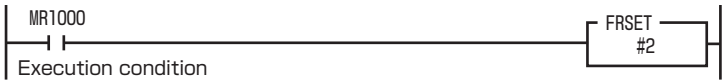
When the execution condition is TRUE, switch the current file register No. to n.

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "File Register Instructions."

● Example of format

FRSET (MR1000, 2)



MEMO

FRSTM Batch-store file register

Return value*1 =FRSTM (Drive number*1 , Bank No., File No., result store device)

Batch-store file register

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
n1	Drive number*2	Set drive number to be saved.*2 (0 to 1)	.U	.U	.U	.U	—	—	—	—	*2○	○	○
n2	File No.	Specifies the bank number of the file register for batch saving. (0 to 15*3)	.U	.U	.U	.U	—	—	—	—	*3○	○	○
S	File No.	Specifies the leading No. of bit device for notifying the store result. (0 to 999)	.U	.U	.U	.U	—	—	—	—	*4○	○	○
D	Save result storage target	Specifies the bit device for notifying the store result.	—	—	—	—	—	—	.B	—	—	○	—
R	Return value	Turns ON when the function is successfully started up, and turns OFF when startup fails.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work relay.
*2 For KV-8000/7500/7300 only, set drive number (0: memory card, 1:CPU memory). In case of KV-5500/5000/3000, this cannot be set.
*3 Set bank number to scope of 0-15 for KV-8000/7500/7300, 0 to 3 for KV-5500/5000/3000.
*4 Please specify the number of data within the range of 0 to 999.
*5 T,C,CTC,CTH,Z cannot be specified.
*6 When a bit device has been specified to the store result Store device, continuous two bits are used as the notification bit. When a word device has been specified, each of the LSBs of continuous two words is used as the notification bit device.

Description of Operation

When the execution condition is TRUE, the data in file register of the n2 bank number is batch-stored in memory card in BIN bits on the drive number n1 as the file name specified by S file number.
The write complete notification and abnormality end warning is stored in D.
When the FRSTM function is started normally, is ON, otherwise OFF.

Reference

- The file name stored in memory card is "\FR\FR (file NO. 3-digit) .FRU" ((file extension is FRB in case of KV-5500/5000/3000.)
  (Example) When setting 10 to S, create the file \FR\FR010.FRU on the memory card or CPU memory.
- Meaning of notification bit of D
  (Example) When MR2000 has been specified

Notification bit is		ON (1, TRUE)	OFF (0, FALSE)
MR2000	End notification	Completed	Not completed
MR2001	Error notification	Normal end	Normal end

"About operation flag (CR2009 to CR2012)", page 1-18
For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "File Register Instructions."

● **Example of format**

MR1000 = FRSTM (1, 10, MR2000)

FRLDM Batch- read file register

Batch- read file register

Return value [*1]=FRLDM (Drive number[*1] , Bank No., File No., result store device)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
n1	Drive number[*2]	Set the drive number for batch reading.[*2] (0 to 1)	.U	.U	.U	.U	—	—	—	—	[*2]○	○	○
n2	File No.	Specifies the bank number of the file register for batch reading. (0 to 15[*3])	.U	.U	.U	.U	—	—	—	—	[*3]○	○	○
S	File No.	Specify file No. of memory card to be read. (0 to 999)	.U	.U	.U	.U	—	—	—	—	[*4]○	○	○
D	Read result storage target	Specifies the leading No. of device for notifying read result	—	—	—	—	—	—	.B	—	—	○	—
R	Return value	Turns ON when the function is successfully started up, and turns OFF when startup fails.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work relay.
*2 For KV-8000/7500/7300 only, set drive number (0: memory card, 1:CPU memory). In case of KV-5500/5000/3000, this cannot be set.
*3 The scope of the bank number is 0 to 15 for KV-8000/7500/7300 and 0 to 3 for KV-5500/5000/3000.
*4 Please specify the number of data within the range of 0 to 999.
*5 T, C, CTC, CTH, Z cannot be specified.
*6 When a bit device has been specified to the read result store device, continuous two bits are used as the notification bit. When a word device has been specified, each of the LSBs of continuous two words is used as the notification bit device.

Description of Operation

When execution conditions are satisfied (TRUE), the data saved in drive number n1 and specified by S file number (\FR\FR(file number consisting of 3 digits).FRB) is read, and saved in the n2 bank number file register.
The write complete notification and abnormality end warning is stored in D.
When the FRSTM function is started normally, the return value is ON, otherwise OFF.

Reference

- The file name read in memory card is "\FR\FR (file NO. 3-digit) .FRU" ((file extension is FRB in case of KV-5500/5000/3000.) ((file extension is FRB in case of KV-5500/5000/3000.)
  (Example) When setting 10 to S, create the file \FR\FR010.FRU on the memory card or CPU memory.
- Meaning of notification bit of D
  (Example) When MR2000 has been specified

Notification bit is		ON(1、TRUE)	OFF(1、FALSE)
MR2000	End notification	Completed	Not completed
MR2001	Error notification	Normal end	Normal end

"About operation flag (CR2009 to CR2012)", page 1-18
For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "File Register Instructions."

● **Example of format**

MR1000 = FRLDM (1, 10, MR2000)

UPSTRT Unit program start

Unit program start

UPSTRT (Execution condition, Unit No., Request details, Execution start slot number, Execution result)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Execution condition	Specifies the bit device to be used for the execution condition.	—	—	—	—	—	—	.B	—	—	○	○
n1	Unit No.*1	Specifies the unit number or storage location. (1 to 48)	.U	.U	.U	.U	—	—	—	—	○	○	○
S2	Request details*2	Specifies the start of the unit program details that stores the execution request.	.U	.U	.U	.U	—	—	—	—	—	○	—
D1	Execution start Slot number	Specifies the storage location for the slot number for starting.	.U	.U	.U	.U	—	—	—	—	—	○	—
D2	Execution result*3	Specifies the start to store the result.	—	—	—	—	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 $ (hexadecimal specification) cannot be used.
*2 4 continuous words are occupied.
*3 2 continuous words are occupied.

Point

- **The UPSTRT function can be used only by KV STUDIO ver.9.0 or later.**
- **The UPSTRT function can be used with the KV-8000 series CPU units and KV-7000 Series CPU units that have CPU function version 2.0 or later only.**
  "KV-7500/7300 CPU Function Version", page 3

Description of Operation

At the rising edge of the execution condition S1, the unit program specified by S2 in the n1 unit starts, following the parameters specified by S2+2.
If the 0 bit of S2+3 is OFF, an assignable empty slot is automatically used.
If the 0 bit of S2+3 is ON, the slot specified by S2+1 is used.
When the function is complete, the used slot number is stored in D1, and the execution results are stored in D2 and D2+1.

S2 ~ : Request details

Request details			Details
S2	Program number		Specifies the unit program number to start.
S2 +1	Slot number		If the 0 bit of S2 +3 is ON, specifies the slot number to use.
S2 +2	Parameter		Specifies the parameters of the unit program to start. For flow: specifies the first block number to execute.
S2 +3	0 bit	Slot specification method	ON : The slot specified by S2 +1 is used. OFF : An assignable empty slot is automatically used.

D2 ~ : Execution result device

Execution result			Details
D2	0 bit	Completion bit	Turns OFF when executing the function, turns ON when complete. Turns ON regardless of whether the function ends normally or abnormally.
	1 bit	Execution failed bit	If execution of the function fails, turns ON simultaneously with the completion bit. Turns OFF when executing the function.
D2 +1	Completion code		The completion code is stored when the function is complete. 0: Started normally. 1: The specified unit program does not exist. 2: There are no empty slots. 3: Operation enabled is OFF. 4: A unit program error has occurred. 5: The specified slot does not exist. 6: The specified slot is in use. 7: The unit program is already being executed. 8: The specified block does not exist. 9: The unit program has already stopped being executed. 10: The specified unit program is not flow. 100: An error has occurred on the unit. 1000: The specified unit program does not exist. 1001: The specified unit is illegal.

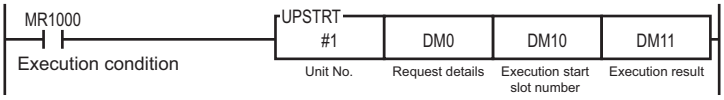
If the unit program start function is executed at the rising edge of the execution condition, the process will continue even if the input conditions are turned OFF at the next scan.

The completion bit and execution failure bit of the result storage location turn ON when completed, and turn OFF when the function is executed.

For details, see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Unit Program Instructions."

● Example of format

UPSTRT(MR1000, #1, DM0, DM10, DM11)



UPSTOP Unit program forced stop

Unit program forced stop

UPSTOP (Execution condition, Unit No., Slot number, Execution result)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Execution condition	Specifies the bit device to be used for the execution condition.	—	—	—	—	—	—	.B	—	—	○	○
n1	Unit No.*1	Specifies the unit number or storage location. (1 to 48)	.U	.U	.U	.U	—	—	—	—	○	○	○
n2	Slot number	Specifies the storage location for the slot number for forced end.	.U	.U	.U	.U	—	—	—	—	○	○	○
D1	Execution result*2	Specifies the start to store the result.	.U	.U	.U	.U	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 $ (hexadecimal specification) cannot be used.

*2 2 continuous words are occupied.

Point

- **The UPSTOP function can be used only by KV STUDIO ver.9.0 or later.**
- **The UPSTOP function can be used with the KV-8000 series CPU units and KV-7000 Series CPU units that have CPU function version 2.0 or later only.**
  "KV-7500/7300 CPU Function Version", page 3

Description of Operation

At the rising edge of the execution condition S, the unit program being executed in the slot specified by n2 in the n1 unit is stopped forcefully, and the slot becomes empty.
When the function is complete, the execution results are stored in D1 and D1+1.

Reference "UPSTRT function", page 6-42

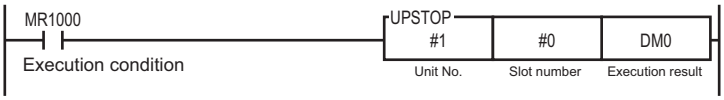
For details, see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Unit Program Instructions."

D1 ~ : Execution result storage destination device

Execution result			Details
D1	0 bit	Completion bit	Turns OFF when executing the function, turns ON when complete. Turns ON regardless of whether the function ends normally or abnormally.
	1 bit	Execution failed bit	If execution of the function fails, turns ON simultaneously with the completion bit. Turns OFF when executing the function.
D1 + 1	Completion code		The completion code is stored when the function is complete. 0 : Started normally. 1 : The specified unit program does not exist. 2 : There are no empty slots. 3 : Operation enabled is OFF. 4 : A unit program error has occurred. 5 : The specified slot does not exist. 6 : The specified slot is in use. 7 : The unit program is already being executed. 8 : The specified block does not exist. 9 : The unit program has already stopped being executed. 10 : The specified unit program is not flow. 100 : An error has occurred on the unit. 1000 : The specified unit program does not exist. 1001 : The specified unit is illegal.

● Example of format

UPSTOP(MR1000, #1, #0, DM0)



UFSUS Suspend flow

Suspend flow

UFSUS (Execution condition, Unit No., Slot number, Request details, Execution result)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Execution condition	Specifies the bit device to be used for the execution condition.	—	—	—	—	—	—	.B	—	—	○	○
n1	Unit No.*1	Specifies the unit number or storage location. (1 to 48)	.U	.U	.U	.U	—	—	—	—	○	○	○
n2	Slot number	Specifies the storage location for the slot number for temporary stop.	.U	.U	.U	.U	—	—	—	—	○	○	○
n3	Request details	Specifies the target that stores the temporary stop method.	.U	.U	.U	.U	—	—	—	—	○	○	—
D1	Execution result*2	Specifies the start to store the result.	.U	.U	.U	.U	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 $ (hexadecimal specification) cannot be used.

*2 2 continuous words are occupied.

Point

- **The UFSUS function can be used only by KV STUDIO ver.9.0 or later.**
- **The UFSUS function can be used with the KV-8000 series CPU units and KV-7000 Series CPU units that have CPU function version 2.0 or later only.**
  "KV-7500/7300 CPU Function Version", page 3

Description of Operation

At the rising edge of the execution condition S, the flow being executed in the slot specified by n2 in the n1 unit is suspended, following the method specified in n3.

When the function is complete, the execution results are stored in D1 and D1+1.

D1 ~ : Execution result storage destination device

Execution result			Details
D1	0 bit	Completion bit	Turns OFF when executing the function, turns ON when complete. Turns ON regardless of whether the function ends normally or abnormally.
	1 bit	Execution failed bit	If execution of the function fails, turns ON simultaneously with the completion bit. Turns OFF when executing the function.
D1 + 1	Completion code		The completion code is stored when the function is complete. 0 : Started normally. 1 : The specified unit program does not exist. 2 : There are no empty slots. 3 : Operation enabled is OFF. 4 : A unit program error has occurred. 5 : The specified slot does not exist. 6 : The specified slot is in use. 7 : The unit program is already being executed. 8 : The specified block does not exist. 9 : The unit program has already stopped being executed. 10 : The specified unit program is not flow. 100 : An error has occurred on the unit. 1000 : The specified unit program does not exist. 1001 : The specified unit is illegal.

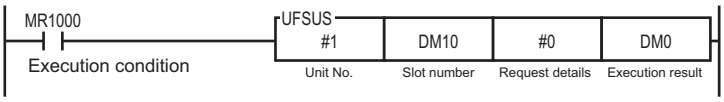
n3 : Request details

Request details			Details
n3	0 bit	Suspension method	ON: Blocks being executed are forcefully ended, and flow is suspended. OFF: Flow is suspended when the blocks being executed are completed.

For details, see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Unit Program Instructions."

● Example of format

UFSUS(MR1000, #1, DM10, #0, DM0)



UFRSM Resume flow

Resume flow

UFRSM (Execution condition, Unit No., Slot number, Execution result)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Execution condition	Specifies the bit device to be used for the execution condition.	—	—	—	—	—	—	.B	—	—	○	○
n1	Unit No.*1	Specifies the unit number or storage location. (1 to 48)	.U	.U	.U	.U	—	—	—	—	○	○	○
n2	Slot number	Specifies the storage location for the slot number for restarting.	.U	.U	.U	.U	—	—	—	—	○	○	○
D1	Execution result*2	Specifies the start to store the result.	.U	.U	.U	.U	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 $ (hexadecimal specification) cannot be used.

*2 2 continuous words are occupied.

Point

- **The UFSUS function can be used only by KV STUDIO ver.9.0 or later.**
- **The UFSUS function can be used with the KV-8000 series CPU units and KV-7000 Series CPU units that have CPU function version 2.0 or later only.**
  "KV-7500/7300 CPU Function Version", page 3

Description of Operation

At the rising edge of the execution condition S, the suspended flow in the slot specified by n2 in the n1 unit is resumed.

When the instruction is complete, the execution results are stored in D1 and D1+1.

Reference "UFSUS function", page 6-46

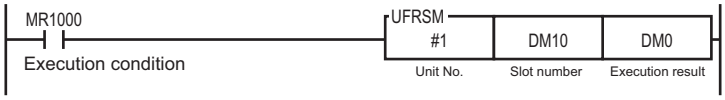
For details, see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Unit Program Instructions."

D1 ~ : Execution result storage destination device

Execution result			Details
D1	0 bit	Completion bit	Turns OFF when executing the function, turns ON when complete. Turns ON regardless of whether the function ends normally or abnormally.
	1 bit	Execution failed bit	If execution of the function fails, turns ON simultaneously with the completion bit. Turns OFF when executing the function.
D1 + 1	Completion code		The completion code is stored when the function is complete. 0 : Started normally. 1 : The specified unit program does not exist. 2 : There are no empty slots. 3 : Operation enabled is OFF. 4 : A unit program error has occurred. 5 : The specified slot does not exist. 6 : The specified slot is in use. 7 : The unit program is already being executed. 8 : The specified block does not exist. 9 : The unit program has already stopped being executed. 10 : The specified unit program is not flow. 100 : An error has occurred on the unit. 1000 : The specified unit program does not exist. 1001 : The specified unit is illegal.

● Example of format

UFRSM(MR1000, #1, DM10, DM0)



UMALLOC Allocate buffer memory

Allocate buffer memory

UMALLOC (Execution condition, Unit No., Allocated size, Execution result, Allocation result storage destination)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Execution condition	Specifies the bit device to be used for the execution condition.	—	—	—	—	—	—	.B	—	—	○	○
n1	Unit No.*1	Specifies the unit number or storage location. (1 to 48)	.U	.U	.U	.U	—	—	—	—	○	○	○
n2	Allocated size	Specifies the storage location for the number of secured words.	.U	.U	.U	.U	—	—	—	—	○	○	○
D1	Execution result	Specifies the start to store the result.	.U	.U	.U	.U	—	—	—	—	○	○	—
D2	Allocation result storage destination	Specifies the storage location for the start address of the secured area.	.U	.U	.U	.U	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 $ (hexadecimal specification) cannot be used.

Point

- **The UMALLOC function can be used only by KV STUDIO ver.9.0 or later.**
- **The UMALLOC function can be used with the KV-8000 series CPU units and KV-7000 Series CPU units that have CPU function version 2.0 or later only.**
  "KV-7500/7300 CPU Function Version", page 3

Description of Operation

At the rising edge of the execution condition S, an area of the number of words specified by n2 from the buffer memory of the n1 unit is allocated. The leading address of the allocated area is stored in D2 and the execution results are stored in D1.
If allocation fails, #0 is stored in D2.

Reference

- The area is automatically allocated.
- Values can be freely read from/written to the allocated area.
- When the unit's operation enabled relay turns OFF, the allocated area is deallocated.

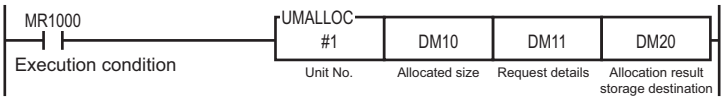
D1 : Execution result

Execution result			Details
D1	0 bit	Completion bit	Turns OFF when executing the function, turns ON when complete. Turns ON regardless of whether the function ends normally or abnormally.
	1 bit	Execution failed bit	If memory cannot be allocated, turns ON simultaneously with the completion bit. Turns OFF when executing the function.

For details, see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Unit Program Instructions."

● **Example of format**

UMALLOC(MR1000, #1, DM10, DM11, DM20)



UPSTRT
UPSTOP
UFSUS
UFRSM
UMALLOC

UREAD Read buffer

Read buffer

UREAD ([Execution condition,][*1] Unit No., leading buffer address, Leading read data store device, number of read data)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
n1	Unit No.[*2]	Specifies expansion special unit/CPU Unit of buffer for data to be read. (0 to 48)	.U	.U	.U	.U	—	—	—	—	*3 ○	○	○
n2	Specify leading buffer device for data to be read[*3]	Specify leading device of buffer memory for data to be read. (0 to 59999[*3])	.U	.U	.U	.U	—	—	—	—	*3 ○	○	○
D	Leading read data store device[*5]	Specify leading device for read data to be stored.	.U	.S	.D	.L	.F	—	—	.T	—	○	—
n3	Number of read data[*4]	Specify number of data read from buffer. (1 to 60000[*4])	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	○	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 Please specify a unit number within the range of 0 to 48 (0 to 15 for the Nano Series). $(HEX) cannot be specified.

*3 Please specify the buffer memory start address as 0 to 59999 for KV-8000/7500/7300 and 0 to 32767 all others.

*4 Please specify the read data number as 1 to 60000 for KV-8000/7500/7300 and 1 to 32768 all others.

*5 CTC,CTH,Z cannot be specified.

Description of Operation

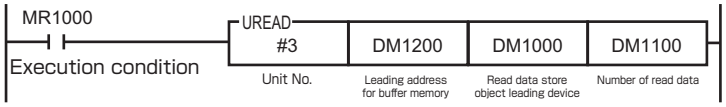
When execution condition is TRUE, read buffer of n1 connected with CPU unit, address of n2 and n3 data (16-bit word data), and stored them in D.

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Buffer Memory Instructions."

● Example of format

UREAD (MR1000, 3, DM1200, DM1000, DM1100)



MEMO

UWRIT Write buffer

Write buffer

UWRIT ([Execution condition,]*1 Unit No., leading buffer address Leading write data device, number of write data)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
n1	Unit No.*2	Specifies expansion special unit/CPU Unit of buffer for data to be written. (0 to 48)	.U	.U	.U	.U	—	—	—	—	*3 ○	○	○
n2	Specify leading buffer device for data to be read*3	Specify leading device of buffer memory for data to be read. (0 to 59999*3)	.U	.U	.U	.U	—	—	—	—	*3 ○	○	○
D	Leading write data device*5	Specify leading buffer device for data to be written.	.U	.S	.D	.L	.F	—	—	.T	○	○	*6 ○
n3	Number of write data*4	Read number of data within specified range. (1 to 60000*4)	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 Please specify a unit number within the range of 0 to 48 (0 to 15 for the Nano Series). cannot be specified by $ (HEX).

*3 Please specify the buffer memory start address as 0 to 59999 for KV-8000/7500/7300 and 0 to 32767 all others.

*4 Please specify the read data number as 1 to 60000 for KV-8000/7500/7300 and 1 to 32768 all others.

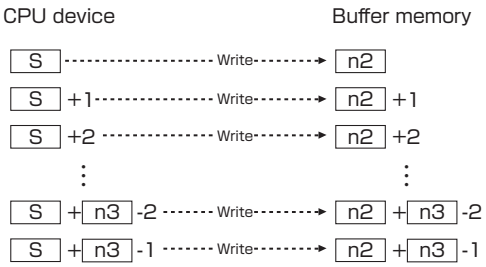
*5 CTC,CTH,Z cannot be specified.

*6 The text string operation can only be specified. Operation other than text string operation (e.g. DM0+DM2) cannot be programmed.

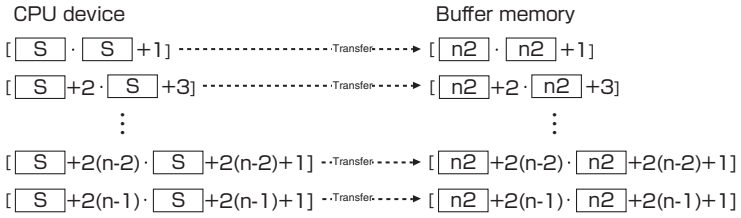
Description of Operation

When execution condition is TRUE, n3 data of device stored in the address of n2 starting from S are written into the buffer of n1 connected with CPU unit.

- S : When move source device is 16-bit (.U/.S)



- [S] : When move source device is 32-bit (.D/.L/.F)



Reference

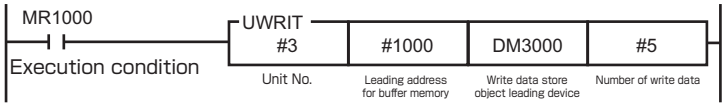
- When the constant is specified in [S], UWRIT function executes the operation identical with UFILL function.
  "UFILL function", page 6-56

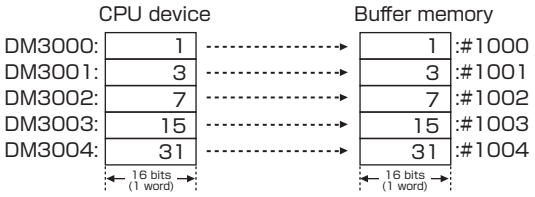
"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Buffer Memory Instructions."

● Example of format

UWRIT (MR1000, 3, 1000, DM3000.U, 5)





UFILL

Insert same value in data string of buffer

Buffer file

UFILL ([Execution condition,][*1] Unit No., Leading buffer address, Leading write data store device, number of write data)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
n1	Unit No.[*2]	Specifies expansion special unit/CPU Unit of buffer for data to be written. (0 to 48)	.U	.U	.U	.U	—	—	—	—	*3 ○	○	○
n2	Specify leading buffer device for data to be read[*3]	Specify leading device of buffer memory for data to be read. (0 to 59999[*3])	.U	.U	.U	.U	—	—	—	—	*3 ○	○	○
D	Leading write data store device[*5]	Specifies the device for the data to be written or stored.	.U	.S	.D	.L	.F	—	—	.T	○	○	○
n3	Number of write data[*4]	Read number of data within specified range. (1 to 60000[*4])	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 Please specify a unit number within the range of 0 to 48 (0 to 15 for the Nano Series). cannot be specified by $ (HEX).

*3 Please specify the buffer memory start address as 0 to 59999 for KV-8000/7500/7300 and 0 to 32767 all others.

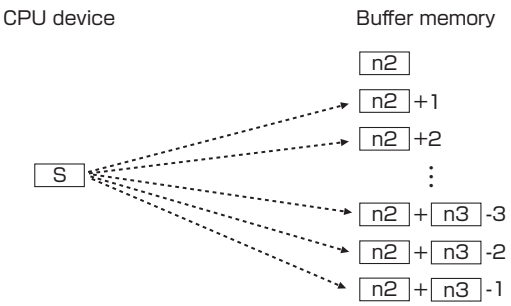
*4 Please specify the read data number as 1 to 60000 for KV-8000/7500/7300 and 1 to 32768 all others.

*5 CTC,CTH,Z cannot be specified.

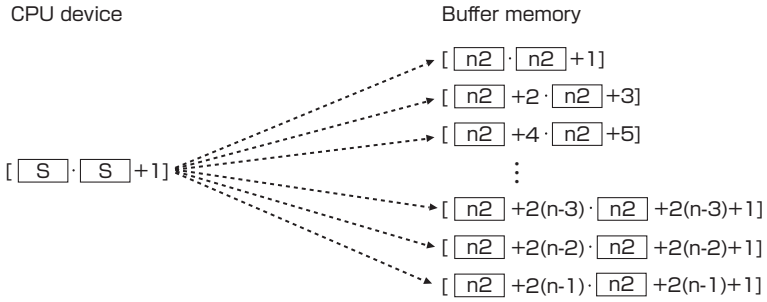
Description of Operation

When execution condition is TRUE, data stored in the address of n2 in S are batch-written into the buffer of n1 connected with CPU unit in three batches.

- S : When move source device is 16-bit (.U/.S)



- [S] : When move source device is 32-bit (.D/.L/.F)

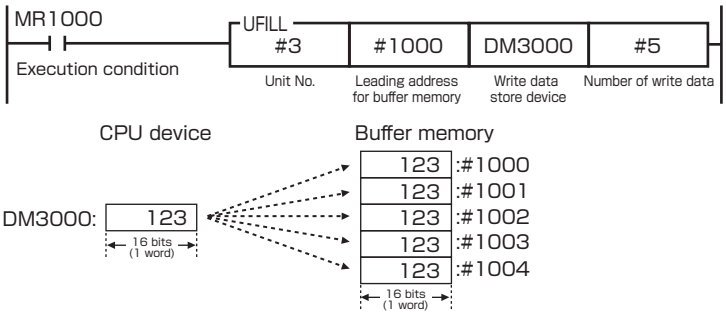


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Buffer Memory Instructions."

● Example of format

UFILL (MR1000, 3, 1000, DM3000.U, 5)



IS_□□□ Unit type judgment

Unit type judgment

IS_□□□ ([Execution condition], Unit No., Execution result)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
n 1	Unit No.*1	Specifies the unit number or storage location. (1 to 48)	.U	.U	.U	.U	—	—	—	—	○	○	○
D	Execution result	Specifies the storage location for the result notification.	.B	.B	.B	.B	—	—	—	—	—	○	—

*1 $ (hexadecimal specification) cannot be specified.

*2 When a word device is specified, the LSB is used as the notification bit device. Nothing except the LSB is changed.

Point

- **The unit type judgment function can be used in KV STUDIO Ver. 9.1 or later only.**
- **The unit type judgment function can be used with the KV-8000 series CPU units and KV-7000 Series CPU units that have CPU function version 2.1 or later only.**
  "KV-7500/7300 CPU Function Version", page 3

Description of Operation

The unit type of the n 1 unit is checked at the rising edge of the execution condition, and D turns ON if it matches the unit type corresponding to the instruction. D turns OFF if there is no match or if it is not connected.

Instruction and unit type

Replace "IS_" in the instruction with "KV-" to find the corresponding unit type.

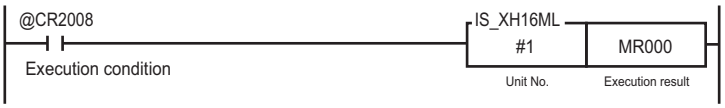
Example)

(Instruction) (Unit type)

IS_XH16ML → KV-XH16ML

● Example of format

IS_XH16ML(@CR2008, #1,MR000.U)



MEMO

ACNT Get the number of array elements

Get the number of array elements

Return value = ACNT (Object array, number of dimensions)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Object array	Array type variable for obtaining the number of elements.*1	—	—	—	—	—	—	—	—	—	○	—
n	Number of dimensions	Specifies the number of dimensions.	.U	—	—	—	—	—	—	—	○	○	○
R	Return value	The ACNT function has a .D format.*2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	—	—

*1 Variables other than array type cannot be specified. Also, variable-length arrays cannot be specified.

*2 When the type other than .D is specified by return value, type conversion will be automatically conducted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

Description of Operation

The number of elements in the [n] dimensions of [S] is counted, and the result is stored in [[R]/[R]+1] as 32-bit BIN data.

Example

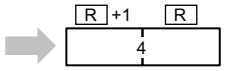
When [S]=Array_A (2-dimensional array of 5×4) and [n]=2

Array_A

Number of elements 4

Number of elements 5

[0,0]	[0,1]	[0,2]	[0,3]
[1,0]	[1,1]	[1,2]	[1,3]
[2,0]	[2,1]	[2,2]	[2,3]
[3,0]	[3,1]	[3,2]	[3,3]
[4,0]	[4,1]	[4,2]	[4,3]



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Get the number of array elements".

Point

The ACNT instruction obtains the number of elements instead of the word size. For example, when UDINT (2-word unsigned integer) is specified as 4 dimensions, 4 is stored in [[R]/[R]+1].

● Example of format

duSizeOfDimension = ACNT(Array_A, 2)

MEMO

WSIZE Get word size

Return value*1 = WSIZE (target variable)

Get word size

Argument/Return Value	Description	Recognized Type								Const	Device/ Variable	For mula
		.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S Target variable	Specifies the variable to get the word size.* 2	—	—	—	—	—	—	—	—	—	○	—
R Return value	The WSIZE function has the .D type.* 3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	—	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.

*2 The following variables cannot be specified.

- ·TIMER type variable
- ·COUNTER type variable
- ·Function block type variable
- ·Variable-length array
- ·Variable-length string

Other arrays can be specified.

*3 When the type other than .D is specified by return value, type conversion will be automatically conducted during conversion.

*4 The ".DF" suffix cannot be specified for device Z.

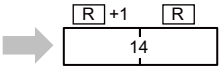
Description of Operation

The word size of S is stored in [R/R+1] as 32-bit binary data.

Example

S = When the following structure type variables are specified

Member name	Data type
member1	UINT
member2	STRING[20]
member3	REAL



→The number of words in STRING[20] is 11 after (20+1)/2 is rounded up, so 1 + 11 + 2 = 14.

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Get word size".

Point

- **The WSIZE instruction obtains the word size. For example, when an array type variable or structure type variable with 4 UDINT type data (2-word unsigned integer) is specified, 8 is stored in [R/R+1] as 2 words × 4 = 8 words. It does not store the number of elements or the number of members.**
- **[R/R+1] stores the word size including the offset. For details on the offset, refer to the User's Manual of the CPU unit used.**
- **[R/R+1] stores the result of word size rounded up. For example, 3 is stored for BOOL[33].**

● Example of format

```
duSizeOfTag = WSIZE(tagAxis_A)
```

7

ARITHMETIC OPERATION FUNCTIONS

Data move function 7-2
Arithmetic Operation Functions 7-14
Logic Operation Functions 7-22
Data Shift Functions 7-38
Data Control Functions 7-62
Data Conversion Functions 7-80
Floating Point Functions 7-114
Text Processing Functions 7-154

BMOV Batch data move of word block

Block move

BMOV ([execution condition,][*1] Move source, move destination, number of move data)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Move source*2*3	Specifies the start of the move source data block.	.U	.U	.U	.U	—	—	—	—	—	○	—
D	Move destination*2*3	Specifies the start of the move destination data block.	Handled as same type as S.					—	—	—	—	○	—
n	Number of move data	Specifies the storage location for the number of data to be moved or the number of move data.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

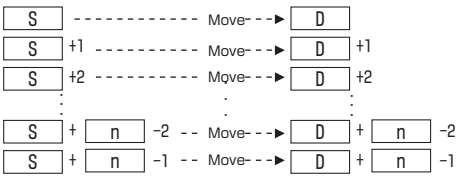
*2 When device T or C is specified, the current value becomes the move source.

*3 When bit data is specified, 16/32 continuous bits are processed. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

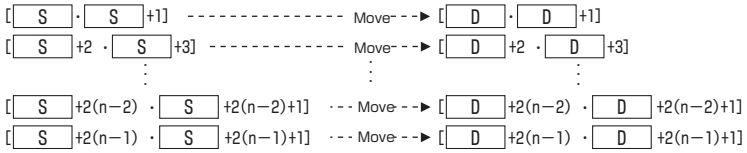
Description of Operation

When the execution condition is met (TRUE), the contents of the word block for n starting with S are moved to the word block for the number of move data starting with D. (When the move destination is 32-bit data, the word block becomes the "specified number of move data x 2 words".)

- S : When move source is 16 bits (.U/.S)



- S : When move source is 32 bits (.D/.L/.F)

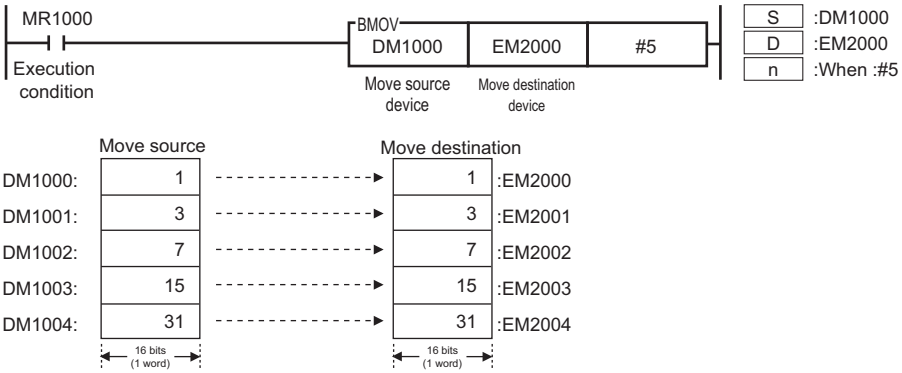


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Move Instructions".

● **Example of format**

BMOV (MR1000, DM1000.U, EM2000, 5)



Point **When [n]=0, the function is not executed.**

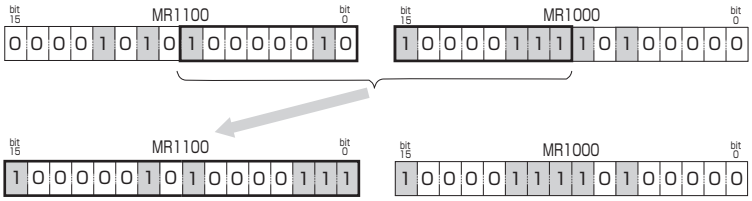
Sample Program

Programmed Script

BMOV (MR1008.U, MR1100.U, 1)

Operation Description

The content of one word (16 bits) starting from MR1008 is block-moved to one word (16 bits) starting from MR1100.



Reference

- Data blocks (bit) at the move source and move destination will be moved successfully even if they partially straddle channels.
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

FMOV Substitute same value in data string

Fill move

FMOV ([execution condition,][*1] Move source, move destination, number of move data)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Move source*2	Specifies the move source or the storage location for the move source data.	.U	.U	.D	.D	.D	*3 .DF	—	—	○	○	○
D	Move destination*2	Specifies the start of the move destination data block.	Handled as same type as S.						—	—	—	○	—
n	Number of move data	Specifies the storage location for the number of data to be moved or the number of move data.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)
*2 When device T, C or CTH is specified, each of their current values becomes the specified destination. When device CTC is specified, the setting value becomes the specified destination.
*3 The ".DF" suffix cannot be specified for device Z.

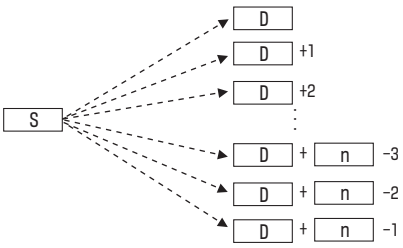
Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

When the execution condition is met (TRUE), the 16-bit / 32-bit data (unsigned / signed) stored in S is moved to the word block of n starting with D. (When the move destination is 32-bit data, the word block becomes the "specified number of move data x 2 words".)

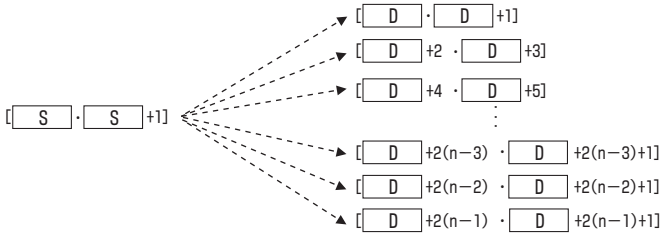
- S : When move source is 16 bits (.U/.S)



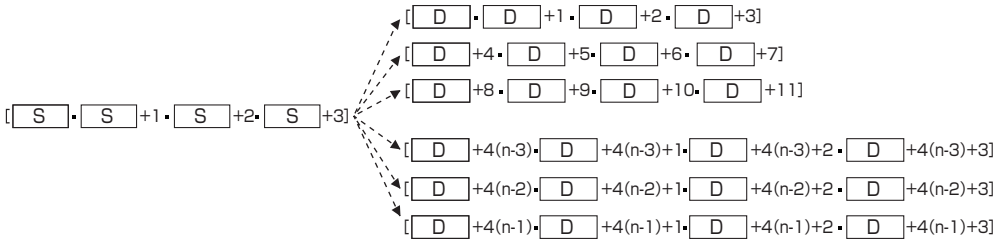
Point

If a negative value is specified for the move source as in FMOV(–1, DM0, 10), 20 words will be moved because the resulting ladder rung is "FMOV.L –1 DM0 10" and the move destination is regarded as 32-bit data.
To set the move source to 16-bit data, write "FMOV(TOS(–1), DM0, 10)."

- S : When move source is 32 bits (.D/.L/.F)



- [S] : When move source is 64 bits (.DF)

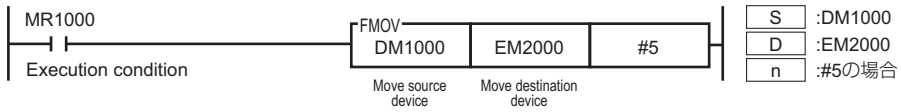


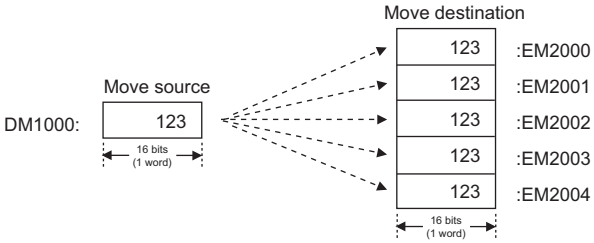
"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Move Instructions".

● Example of format

FMOV (MR1000, DM1000.U, EM2000, 5)





Point **When [n]=0, the function is not executed.**

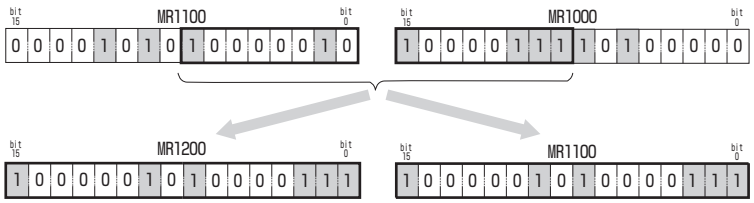
Sample Program

Programmed Script

FMOV (MR1008.U, MR1100.U, 2)

Operation Description

The FMOV function can use bit data for the move source and move destination.
The content of one word (16 bits) starting from MR1008 is block-moved to two words (32 bits) starting from MR1100.



Reference

- Data blocks (bit) at the move source and move destination will be moved successfully even if they partially straddle channels.
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

PMOV Move continuous bit information to a specified position

Partial move

PMOV ([execution condition,][*1] Move source, move destination start bit position, move destination, move destination start bit position, number of move bits)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Move source*2*3	Specifies the storage location for the move source data.	.U	.U	.D	.D	.D	—	—	—	○	*6 ○	○
n1	Move source start bit position*4*5	Specifies the storage location for the start bit position of the move source or the start bit position.	.U	.U	.U	.U	—	—	—	—	○	○	○
D	Move destination*2*3	Specifies the move destination.	Handled as same type as S.						—	—	—	○	—
n2	Move destination start bit position*4*5	Specifies the storage location for the start bit position of the move destination or the start bit position.	.U	.U	.U	.U	—	—	—	—	○	○	○
n3	Number of move bits	Specifies the storage location for the number of data to be moved or the number of move data.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When bit data is specified, it occupies continuous 16 bits (for .U) / 32 bits (for .D).

*3 S and D can be the same target.

*4 When, for example, the suffix is .U, DM1000 has been specified to n1 (n2), and the value is 20 (BIN: 0000 0000 0001 0100), "4" is used as the value of n1 (n2) because the value of the lower 4 bits is 4 (0100).

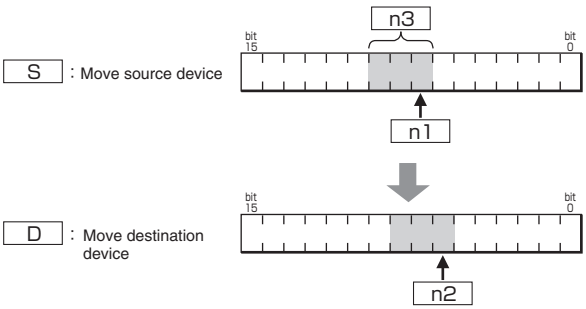
*5 Specify 0 to 15 when the S move source is 16 bits (.U/.S), and specify 0 to 31 when it is 32 bits (.D/.L).

*6 A constant can be specified for the move source only when the following conditions are met.
- KV STUDIO Ver. 6.00 or higher
- Only when the compatible model is a KV-8000/7000 Series CPU unit, a KV-5500/5000/3000 Series CPU unit that has CPU function version 2.0 or higher, or a KV Nano Series base unit.

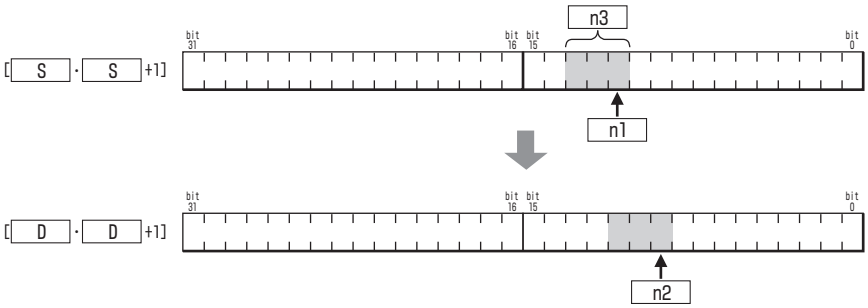
Description of Operation

When the execution condition is met (TRUE), the bits after the bit position specified by n1 of the 16-bit / 32-bit data stored in S are moved to the bits after the bit position specified by n2 of D in blocks (bits) by the number of bits specified by n3.

- S : When move source is 16 bits (.U/.S)



- S : When move source is 32 bits (.D/.L/.F)



"About operation flag (CR2009 to CR2012)", page 1-18

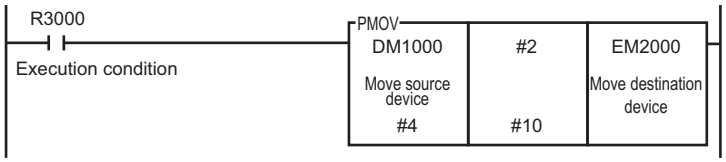
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Move Instructions".

Reference

The original values of bits that are not moved (not destinationed for the move) are held as they are.

● Example of format

PMOV (R3000, DM1000.U, 2, EM2000, 4, 10)



Point

The function is not executed when either of the following conditions is established:

- **When the suffix of S is .U, and n3 =0 or 16 <[n1 + n3] or 16<[n2 + n3].**
- **When the suffix of S is .D, and n3 =0 or 32 <[n1 + n3] or 32<[n2 + n3].**

Sample Program

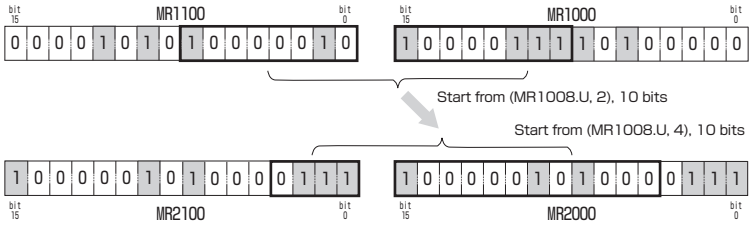
● To move bit data

Programmed Script

PMOV (MR1008.U, 2, MR2004.U, 4, 10)

Operation Description

The ten bits (MR1010 to MR1103) starting from the 2nd bit of MR1008.U are blockmoved (by bit) to the ten bits (MR2008 to MR2101) starting from the 4th bit of MR2004.U.



● When this function is used in combination with other functions

Programmed Script

PMOV (DM3000, EM3001, DM3100, EM3002, ANDA (EM3000, $7))

Operation Description

The ten bits (MR1010 to MR1103) starting from the 2nd bit of MR1008.U are block moved (by bit) to the ten bits (MR2008 to MR2101) starting from the 4th bit of MR2004.U.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "ANDA function", page 7-22
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

MEMO

BYBMOV Move continuous byte string information to specified position (H -> L)

BYBMOV

BYBMOV ([execution condition,][*1] move source byte string start, move source byte offset, move destination byte string storage destination start, move destination byte offset, number of move bytes)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Move source byte string start[*2]	Specifies the start that stores the move source byte string data.	.U	.S	.D	.L	.F	[*3] .DF	.B	.T	—	○	—
n1	Move source byte offset	Specifies the storage location for the position from the move source byte string start or specifies that position (0 to 65535).	.U	.U	.U	.U	—	—	—	—	○	○	○
D	Move destination byte string start[*2]	Specifies the start of the move destination byte string.	.U	.S	.D	.L	.F	[*3] .DF	.B	.T	—	○	—
n2	Move destination byte offset	Specifies the storage location for the position from the move destination byte string start or specifies that position (0 to 65535).	.U	.U	.U	.U	—	—	—	—	○	○	○
n3	Number of move bytes	Specifies the storage location for the number of data to move or the number of move data (1 to 65535).	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2
- For bit data, specify the start of the channel.
- S and D can be the same target.

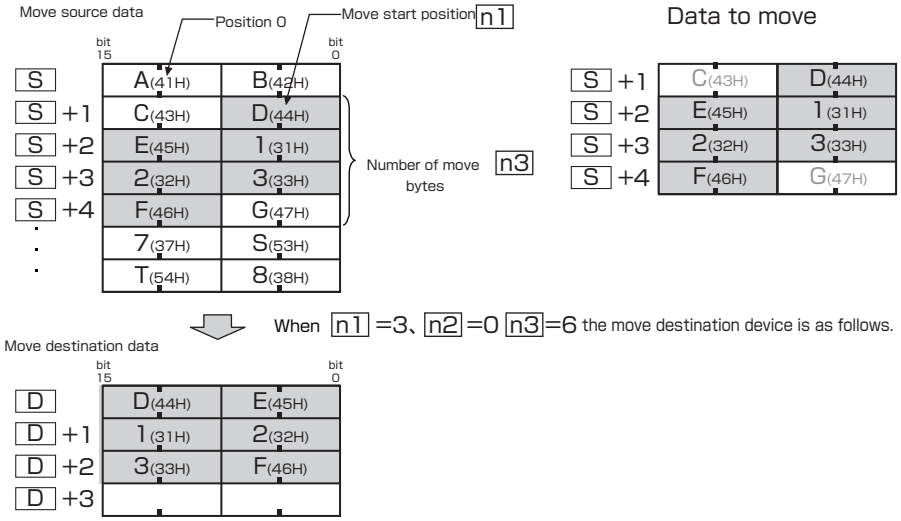
*3 The ".DF" suffix cannot be specified for device Z.

Point
- **The BYBMOV function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can be used with KV-8000/7000 Series CPU units and KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or later.**

"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

When the execution condition is met (TRUE), n3 bytes from the n1 th byte (counting from the high-order byte) of S are moved to the (n2 + n3 − 1)th byte from the n2 th byte (counting from the high-order byte) of D. The original values are saved in the bytes where data is not moved.

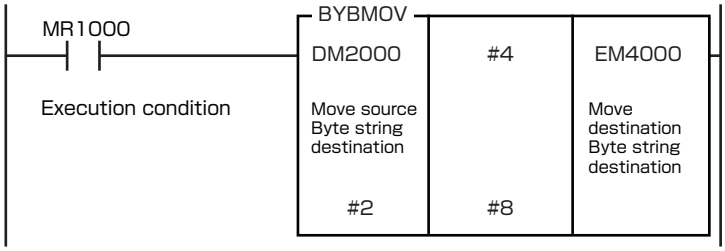


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Move Instructions".

● Example of format

BYBMOV (MR1000, DM2000, 4, EM4000, 2, 8)



Point **When n3 =0, the function is not executed.**

Sample Program

Programmed Script

BYBMOV (DM200, 4, DM400, 2, 8)

Operation Description

The 4th byte to 8th byte (counting from the high-order byte) of the move source byte string data starting with DM200 is moved to the 2nd to 9th bytes (counting from the highorder byte) of the move destination byte string data starting with DM400.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

BYLMOV Move continuous byte string information to specified position (L -> H)

BYLMOV

BYLMOV ([execution condition,][*1] move source byte string start, move source byte offset, move destination byte string storage destination start, move destination byte offset, number of move bytes)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Move source byte string start[*2]	Specifies the start that stores the move source byte string data.	.U	.S	.D	.L	.F	[*3] .DF	.B	.T	—	○	—
n1	Move source byte offset	Specifies the storage location for the position from the move source byte string start or specifies that position (0 to 65535).	.U	.U	.U	.U	—	—	—	—	○	○	○
D	Move destination byte string start[*2]	Specifies the start of the move destination byte string.	.U	.S	.D	.L	.F	[*3] .DF	.B	.T	—	○	—
n2	Move destination byte offset	Specifies the storage location for the position from the move destination byte string start or specifies that position (0 to 65535).	.U	.U	.U	.U	—	—	—	—	○	○	○
n3	Number of move bytes	Specifies the storage location for the number of data to move or the number of move data (1 to 65535).	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2
- For bit data, specify the start of the channel.
- S and D can be the same target.

*3 The ".DF" suffix cannot be specified for device Z.

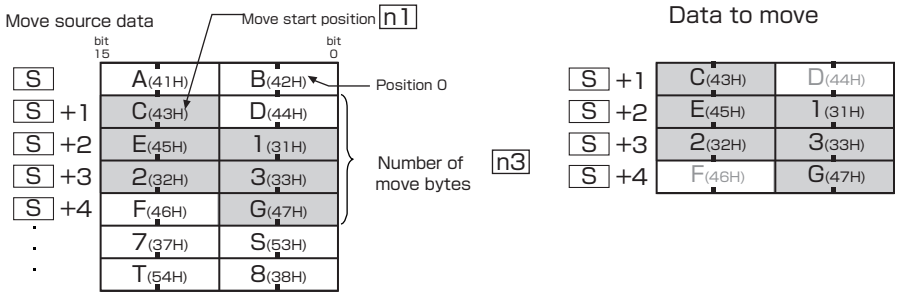
Point

- **The BYLMOV function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can be used only with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or later and KV Nano Series base units.**

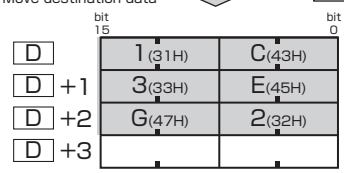
"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

When the execution condition is met (TRUE), n3 bytes from the n1 th byte (counting from the low-order byte) of S are moved to the (n2 + n3 − 1)th byte from the n2 th byte (counting from the low-order byte) of D. The original values are saved in the bytes where data is not moved.



When n1 =3、n2=0 n3=6 , the move destination device is as follows.

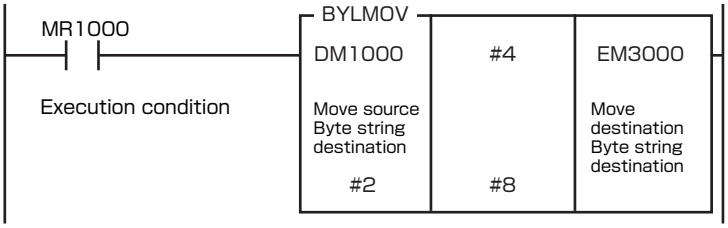


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Move Instructions".

● Example of format

BYLMOV (MR1000, DM1000, 4, EM3000, 2, 8)



Point **When n3 =0, the function is not executed.**

Sample Program

Programmed Script

BYLMOV (DM100, 4, DM200, 2, 8)

Operation Description

The 4th byte to 8th byte (counting from the lower-order byte) of the move source byte string data starting with DM100 is moved to the 2nd to 9th bytes (counting from the lower-order byte) of the move destination byte string data starting with DM200.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

INC Increament by 1

Increment memory

INC ([execution condition,]*1 Increment destination)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Increment destination*2	Specifies the destination to increment by 1.	.U	.S	.D	.L	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When bit data is specified, 16/32 continuous bits are processed.
When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

Description of Operation

The data of D is incremented by 1, and the result is stored in D.

- D : When the increment destination is unsigned 16-bit data (.U)
  An overflow occurs if the operation result is greater than 65535.
  (When the count before the operation is 65535, the count becomes 0, and an overflow occurs.)
- D : When the increment destination is signed 16-bit data (.S)
  An overflow occurs if the operation result is greater than +32767.
  (When the count before the operation is +32767, the count becomes -32768, and an overflow occurs.)
- D : When the increment destination is unsigned 32-bit data (.D)
  An overflow occurs if the operation result is greater than 4294967295.
  (When the count before the operation is 4294967295, the count becomes 0, and an overflow occurs.)
- D : When the increment destination is signed 32-bit data (.L)
  An overflow occurs if the operation result is greater than +2147483647.
  (When the count before the operation is +2147483647, the count becomes -2147483648, and an overflow occurs.)

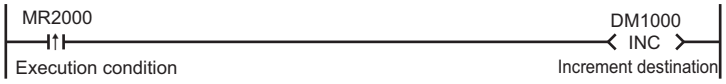
Point Arithmetic operation is performed normally even if an overflow occurs.

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Arithmetic/Comparison Instructions".

● Example of format

INC (LDP (MR2000), DM1000.U)



Sample Program

There are two ways of incrementing the target data, by using operation formulas (DM1000 = DM1000 + 1) and by using the INC function.

When using operation formulas, attention must be paid to reduce accuracy as the right side of the instruction is operated on as an .L suffix regardless of the suffix. However, when using the INC function, you can program instructions without being aware of any drop in accuracy as the instruction is operated on by the suffix specified to the destination.

Programmed Script

```
IF LDP (MR1000) THEN
    DM1000 = DM1000 + 1
    INC (DM1100)
END IF
```

Operation Description

The current values of DM1000 and DM1100 are incremented by 1 at the rising edge (OFF→ON) of MR1000.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

DEC Decrement by 1

Decrement memory

DEC ([execution condition,]*1 Decrement destination)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Decrement destination*2	Specifies the destination to decrement by 1.	.U	.S	.D	.L	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When bit data is specified, 16/32 continuous bits are processed.
When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

Description of Operation

The data of D is decremented by 1, and the result is stored in D.

- D : When the decrement destination is unsigned 16-bit data (.U)
  If operation result is less than 0, underflow will occur.
  (When the count before the operation is 0, the count becomes 65535, and an underflow occurs.)
- D : When the decrement destination is signed 16-bit data (.S)
  If operation result is less than -32768, underflow will occur.
  (When the count before the operation is -32768, the count becomes +32767, and an underflow occurs.)
- D : When the decrement destination is unsigned 32-bit data (.D)
  If operation result is less than 0, underflow will occur.
  (When the count before the operation is 0, the count becomes 4294967295, and an underflow occurs.)
- D : When the decrement destination is signed 32-bit data (.L)
  If operation result is less than -2147483648, underflow will occur.
  (When the count before the operation is -2147483648, the count becomes +2147483647, and an underflow occurs.)

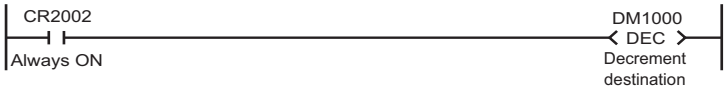
Point Arithmetic operation is performed normally even if an underflow occurs.

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Arithmetic/Comparison Instructions".

● Example of format

DEC (DM1000.U)



Sample Program

There are two ways of decrementing the target data, by using operation formulas (DM1000 = DM1000-1) and by using the DEC function.

When using operation formulas, attention must be paid to reduce accuracy as the right side of the instruction is operated on as an .L suffix regardless of the suffix. However, when using the DEC function, you can program instructions without being aware of any drop in accuracy as the instruction is operated on by the suffix specified to the destination.

Programmed Script

```
IF LDP (MR1000) THEN
    DM1000 = DM1000 - 1
    DEC (DM1100)
END IF
```

Operation Description

The current values of DM1000 and DM1100 are decremented by 1 at the rising edge (OFF→ON) of MR1000.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ROOT Square root

Square root

Return value*1 = ROOT (operation destination)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Operation destination	Specifies the operation destination or the operation destination data.	.U	.S	.D	.L	.F	*3 .DF	—	—	○	○	○
R	Return value	ROOT function type is identical to the arguments.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

Point

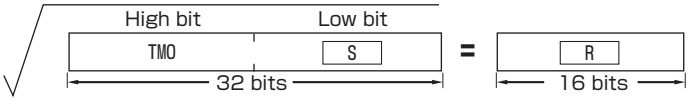
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

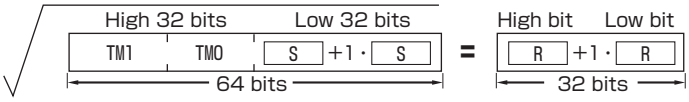
Description of Operation

The square root of the data stored in S is calculated, and the result is stored in R.

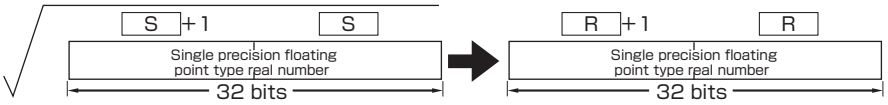
- S : When operation destination is 16 bits (.U/.S)



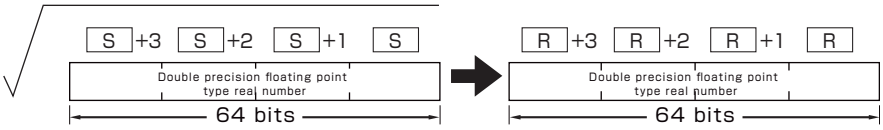
- S : When operation destination is 32 bits (.D/.L)



- S : When the operation destination is single-precision floating point type real number (.F)



- S : When the operation destination is double-precision floating point type real number (.DF)



Reference

When integer is specified for operation destination data:

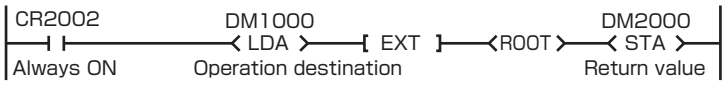
- The square root of 16-bit/32-bit BIN data targeted for conversion is calculated as 32-bit/64-bit BIN data resulting from having extended each of these data, and each result is stored again as 16-bit/32-bit BIN data.
- The decimal point and below in the operation result are discarded.

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Arithmetic/Comparison Instructions".

● Example of format

DM2000.U = ROOT (DM1000.U)



Sample Program

● When using operation formulas

Programmed Script

DM2000.L = **ROOT** (DM1000.S * EM1000.S)

Operation Description

The square root of the product (multiplication) of the data stored in DM1000.S and EM1000.S is calculated, and the result is stored in DM2000.L.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

SQRT Square root

Return value*1 = SQRT (Operation destination)

Square root

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Operation destination	Specifies the operation destination or the operation destination data.	.U	.S	.D	.L	.F	*3 .DF	—	—	○	○	○
R	Return value	SQRT function type is identical to the arguments.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

Point

- **The SQRT function can be used only with KV STUDIO Ver. 4.00 or higher.**
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page -4

Description of Operation

This function is similar to "ROOT function" except function name is different.

"ROOT function", page 7-18

MEMO

ANDA Logical AND operation

ANDA

Return value*1 = ANDA (Operation destination, operation data)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S1	Operation destination*2	Specifies the operation destination or the operation destination data.	.U	.U	.D	.D	—	—	—	—	○	○	○
S2	Operation data*2	Specifies the storage location for the logical AND data or the logical AND data.	Handled as same type as S1.				—	—	—	—	○	○	○
R	Return value	ANDA function type is identical to the arguments.* 3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When bit data is specified, 16/32 continuous bits are processed. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

*3 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*4 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

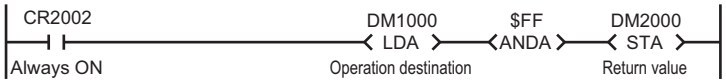
Description of Operation

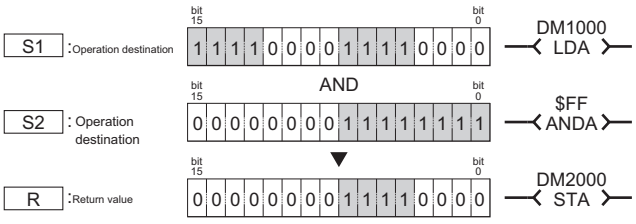
The logical AND operation of the data stored in S1 and binary data stored in S2 is executed.

● Example of format

- S1 : When operation destination is 16 bits (.U/.S)

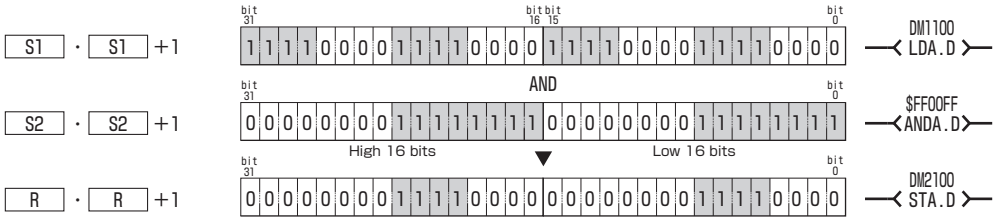
DM2000.U = **ANDA** (DM1000.U, $00FF)





- S1 : When operation destination is 32 bits (.D/.L)

DM2100.D = **ANDA** (DM1100.D, $00FF00FF)



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Logical Operation Functions".

● Logical AND operation truth table

Operation destination	0	0	1	1
Operation data	0	1	0	1
Return value	0	0	0	1

Sample Program

● To calculate the logical AND of the bit data

Programmed Script

DM2000.D = **ANDA** (MR1000.D, DM1100.D)

Operation Description

The logical AND operation of the data stored in MR1000.D and the data stored in DM1100.D is executed.
The result is stored in DM2000.D.

● When this function is used in combination with other functions

Programmed Script

FMOV (**ANDA** (DM1000, $FF), DM2000, 10)

Operation Description

The logical AND operation of the data stored in DM1000 and the operation data $FF (HEX) is executed.
The operation result data is block-moved to ten words starting from DM2000 by the same data.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "FMOV function", page 7-4
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ORA Logical OR operation

ORA

Return value*1 = ORA (Operation destination, operation data)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S1	Operation destination*2	Specifies the operation destination or the operation destination data.	.U	.U	.D	.D	—	—	—	—	○	○	○
S2	Operation data*2	Specifies the storage location for the logical OR data or the logical OR data.	Handled as same type as S1.				—	—	—	—	○	○	○
R	Return value	ORA function type is identical to the arguments.* 3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When bit data is specified, 16/32 continuous bits are processed. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

*3 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*4 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page -4

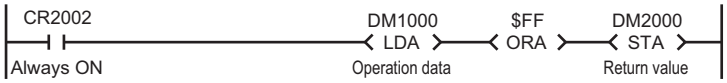
Description of Operation

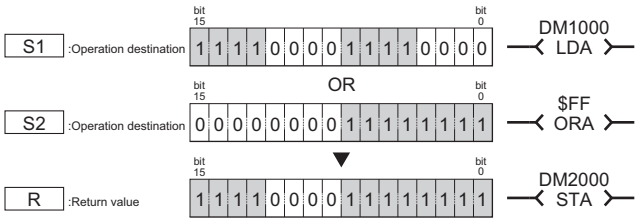
The logical OR operation of the data stored in S1 and binary data stored in S2 is executed.

● Example of format

- S1 : When operation destination is 16 bits (.U/.S)

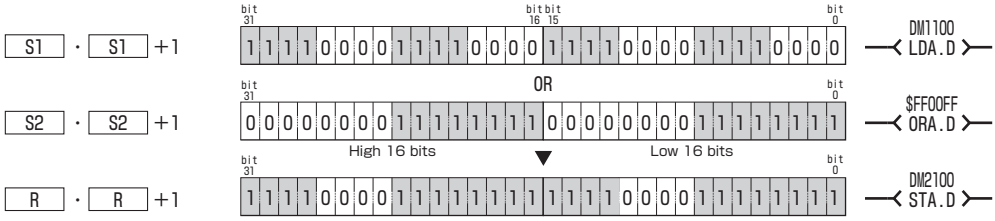
DM2000.U = **ORA** (DM1000.U, $00FF)





- S1 : When operation destination is 32 bits (.D/.L)

DM2100.D = **ORA** (DM1100.D, $00FF00FF)



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Logical Operation Functions".

● Logical OR operation truth table

Operation destination	0	0	1	1
Operation data	0	1	0	1
Return value	0	1	1	1

Sample Program

● To calculate the logical OR of the bit data

Programmed Script

DM2000.D = **ORA** (MR1000.D, DM1100.D)

Operation Description

The data stored in MR1000.D and the data stored in DM1100.D are ORed.
The result is stored in DM2000.D.

● When this function is used in combination with other functions

Programmed Script

R5000.U = **ORA** (TBCD (DM1000), $1111)

Operation Description

The data stored in DM1000 is converted to 4-digit BCD data. The converted 4-digit BCD data and $1111 (HEX) are ORed (converted to an odd number (+1) when each of the digits of the 4-digit BCD are an even number), and the operation result data is output to output relays R5000 to R5015.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "TBCD function", page 7-80
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

EORA Exclusive OR operation

Return value*1 = EORA (Operation destination, operation data)

Exclusive ORAA

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Operation destination*2	Specifies the operation destination or the operation destination data.	.U	.U	.D	.D	—	—	—	—	○	○	○
S2	Operation data*2	Specifies the storage location for the exclusive OR data or the exclusive OR data.	Handled as same type as S1.				—	—	—	—	○	○	○
R	Return value	EORA function type is identical to the arguments.* 3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When bit data is specified, 16/32 continuous bits are processed. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

*3 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*4 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page -4

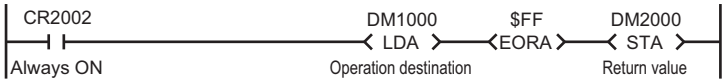
Description of Operation

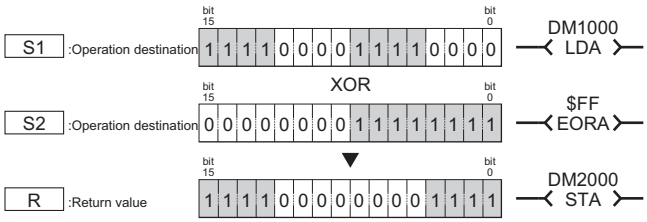
The exclusive OR operation of the data stored in S1 and the binary data stored in S2 is executed.

● Example of format

- S1 : When operation destination is 16 bits (.U/.S)

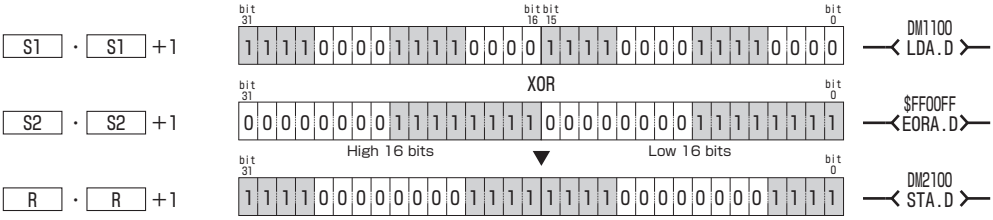
DM2000.U = **EORA** (DM1000.U, $00FF)





- S1 : When operation destination is 32 bits (.D/.L)

DM2100.D = **EORA** (DM1100.D, $00FF00FF)



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Logical Operation Functions".

● Exclusive OR operation truth table

Operation destination	0	0	1	1
Operation data	0	1	0	1
Return value	0	1	1	0

Sample Program

● To calculate the exclusive OR of the bit data

Programmed Script: DM2000.D = **EORA** (MR1000.D, DM1100.D)

Operation Description: The data stored in MR1000.D and the data stored in DM1100.D are exclusive ORed. The result is stored in DM2000.D.

● Example of use

The EORA function is used, for example, to invert the ON and OFF states of only the bits specified by the operation data on data specified by the operation destination as the logic of only arbitrary bits specified as 1 (ON) is inverted.

Programmed Script: R5000.U = **EORA** (R5000.U, $00FF)

Operation Description: Output relays R5000 to R5015 invert and output the current output state (ON/OFF) of R5000 to R5007 at every scan.
(R5008 to R5015 do not change state.)

Reference: The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ENRA Exclusive NOR

Return value*1 = ENRA (Operation destination, operation data)

Exclusive NORAA

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Operation destination*2	Specifies the operation destination or the operation destination data.	.U	.U	.D	.D	—	—	—	—	○	○	○
S2	Operation data*2	Specifies the storage location for the exclusive NOR data or the exclusive NOR data.	Handled as same type as S1.				—	—	—	—	○	○	○
R	Return value	EORA function type is identical to the arguments.* 3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When bit data is specified, 16/32 continuous bits are processed. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

*3 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*4 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page -4

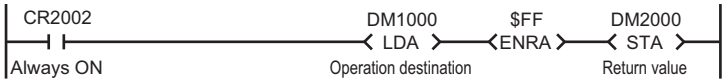
Description of Operation

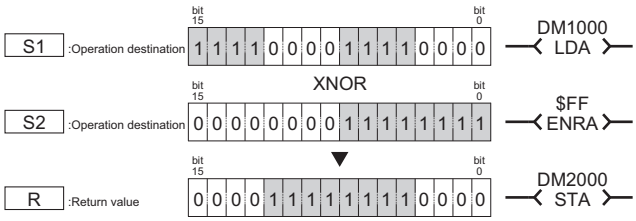
The exclusive NOR operation of the data stored in S1 and the binary data stored in S2 is executed, and the result is stored in R.

● Example of format

- S1 : When operation destination is 16 bits (.U/.S)

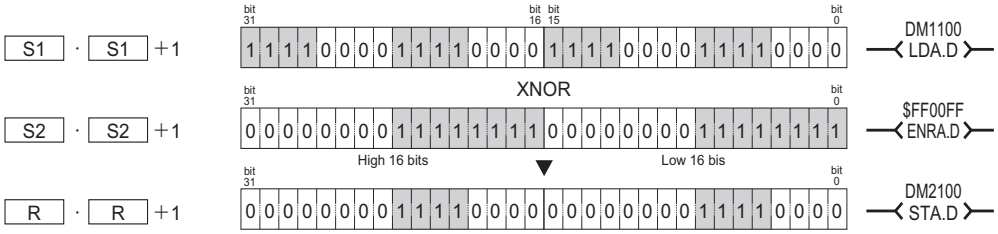
DM2000.U = **ENRA** (DM1000.U, $00FF)





- S1 : When operation destination is 32 bits (.D/.L)

DM2100.D = **ENRA** (DM1100.D, $00FF00FF)



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Logical Operation Functions".

● Exclusive NOR operation truth table

Operation destination	0	0	1	1
Operation data	0	1	0	1
Return value	1	0	0	1

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000 = BCNT (**ENRA** (MR1000.U, MR1100.U))

Operation Description

The number of matching bits in the 16-bit BIN data currently stored in MR1000.U and MR1100.U are calculated, and the result is stored in DM2000.

Reference

Note that this function sometimes cannot be used depending on the argument of the function it is combined with.

"BCNT function", page 8-6

● Example of use

The ENRA function is used to understand bit changes between the previous value and the current value as ON:ON or OFF:OFF matching bits can be extracted.

Programmed Script

R1000.U = **ENRA** (R2000.U, MR2000.U)

Operation Description

Bits matching the previous input state currently stored in MR2000 are extracted at the current input state (ON/OFF) of R2000 to R2015, and the corresponding bits of output relays R1000 to R1015 are turned ON.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

COM Bit inversion

Return value*1 = COM (Operation destination)

Com plement

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Operation destination	Specifies the operation destination or the operation destination data.	.U	.U	.D	.D	—	—	—	—	○	○	○
R	Return value	The type of COM function is identical to argument.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

Point

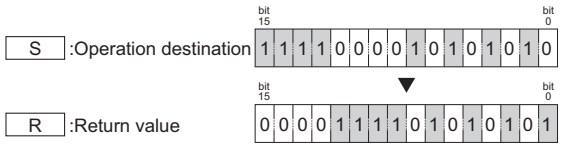
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page -4

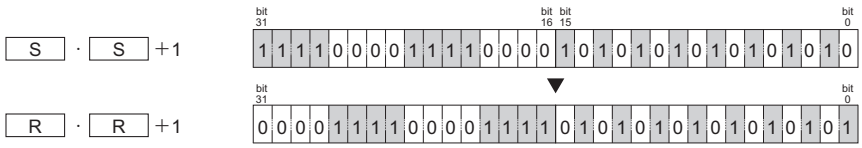
Description of Operation

Each of the bits of the BIN data stored in S is inverted.

- S : When the operation destination is 16 bits



- S : When the operation destination is 32 bits



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Logical Operation Functions".

● Example of format

DM2000.U = **COM** (DM1000.U)

Sample Program

● Example of use

The COM function is used, for example, to invert the current ON and OFF bit states as it can invert the logic of the BIN data specified by the operation destination.

Programmed Script

R5000.U = **COM** (R5000.U)

Operation Description

Output relays R5000 to R5015 invert (alternately turn ON/OFF) and output the current output state at every scan.

Reference

- The above script is equivalent to
  R5000.U = **EORA** (R5000.U, $FFFF).
  "EORA function", page 7-26
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

NEG Invert sign

Return value*1 = NEG (Operation destination)

Negative

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Operation destination	Specifies the operation destination or the operation destination data.	.U	.U	.D	.D	.F	*3 .DF	—	—	○	○	○
R	Return value	TBCD function type is identical to the arguments.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

Point

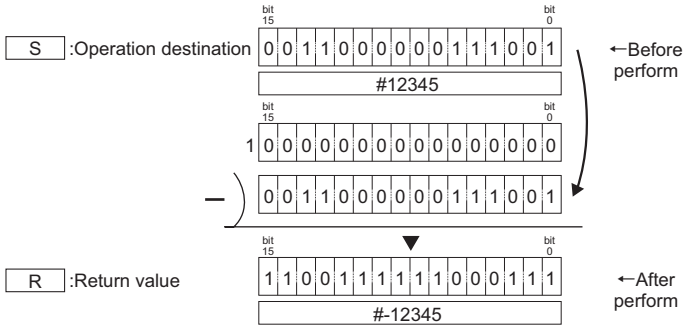
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page -4

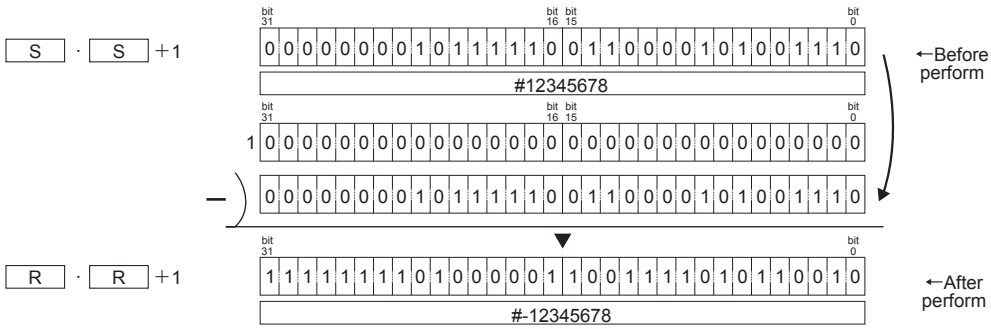
Description of Operation

The plus/minus sign of the BIN data stored in S is inverted (2's complement is calculated).

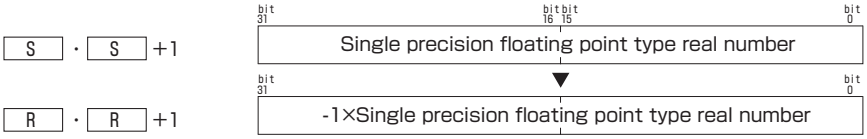
- S : When the operation destination is 16 bits



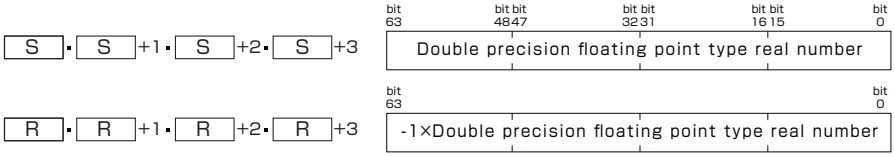
- S : When the operation destination is 32 bits



- S : When the operation destination is single-precision floating point type real number



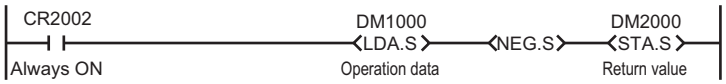
- S : When the operation destination is double-precision floating point type real number



For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Logical Operation Functions".

● Example of format

DM2000.S = **NEG** (DM1000.S)



Sample Program

● To invert the sign of the bit data

Programmed Script

DM2000.U = **NEG** (MR1000.U)

Operation Description

The plus/minus sign of the BIN data stored in MR1000 is inverted.
The result is stored in DM2000.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

Hint

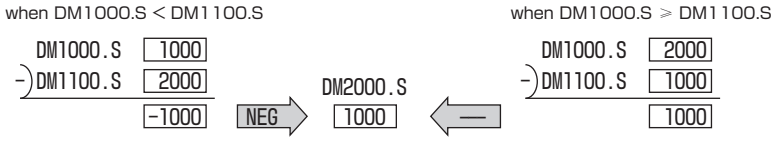
The NEG function is used, for example, to invert the sign of a minus value to calculate the absolute value as it can invert the plus/minus sign of BIN data specified by the operation destination.

Programmed Script

```
IF DM1000.S < DM1100.S THEN
DM2000.S = NEG ( TOS ( DM1000.S - DM1100.S ) ) 'Sign of operation result inverted
ELSE
DM2000.S = TOS ( DM1000.S - DM1100.S )
END IF
```

Operation Description

The absolute value of the operation result of DM1000.S-DM1100.S is stored in DM2000.



Point

When the NEG function is used on signed 16-bit data (.S), the range that can be handled is -32768 to 32767. For this reason, the operation result becomes -32768 when -32768 has been operated on by the NEG function.
At this time, +32768 can be calculated by using the TOL function to extend the data to signed 32-bit data (.L) and inverting its sign.
Also, as the device is handled as a formula by prefixing it with a minus operator (-) instead of using the NEG function, the data is automatically extended to signed 32-bit data (.L), and the sign of -32768 is inverted to find +32768.

MEMO

ABS Absolute value

Return value*1 = ABS (Operation destination)

Absolute value

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Operation destination	Specifies the operation destination or the operation destination data.	.U	.S	.D	.L	.F	*3 .DF	—	—	○	○	○
R	Return value	ABS function type is identical to the arguments.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

Point

- **The return value can be omitted. When omitted, the operation result is stored in internal registers.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

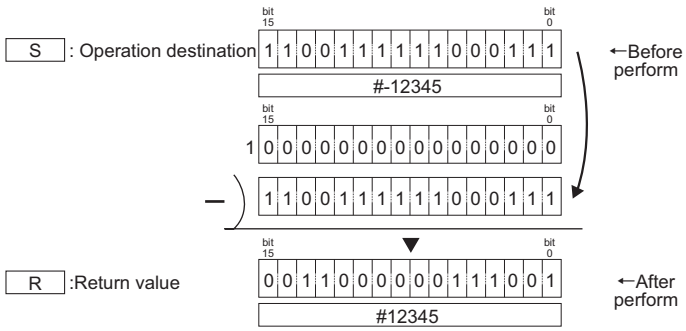
"KV-5500/5000/3000 CPU Function Version", page -4

Description of Operation

The absolute value of binary data saved in S is saved to R.

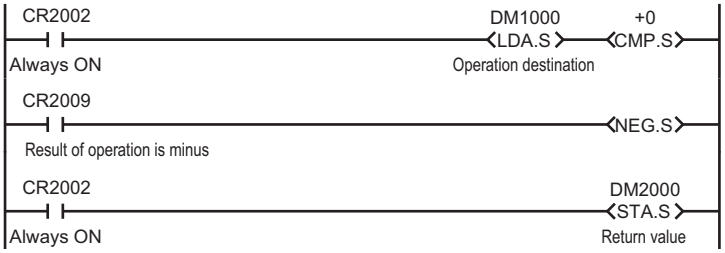
- S :When the operation destination is unsigned 16/32-bit data (.U/.D)
  S is saved to R without processing.

- S :When the operation destination is signed 16/32-bit data (.S/.L) or single/double precision floating point type real number data (.F/.DF)
  When S ≧0, it is saved to R without processing.
  When S ＜0, it is saved to R after the sign is reverted (2's complement operation).



● **Example of format**

DM2000.S = **ABS** (DM1000.S)



SRA Shift right (bit unit)

Return value*1 = SRA (Operation destination, number of right shifts)

Shift right A

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Operation destination	Specifies the operation destination or the operation destination data.	.U	.U	.D	.D	—	—	—	—	○	○	○
n	Number of right shifts	Specifies the storage location for the number of right shifts or the number of right shifts.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	SRA function type is identical to the 1st argument.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.
*3 The ".DF" suffix cannot be specified for device Z.

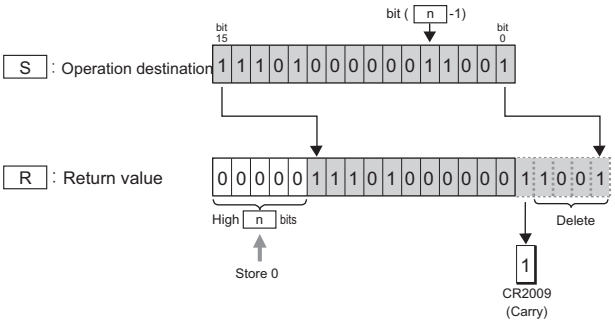
Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

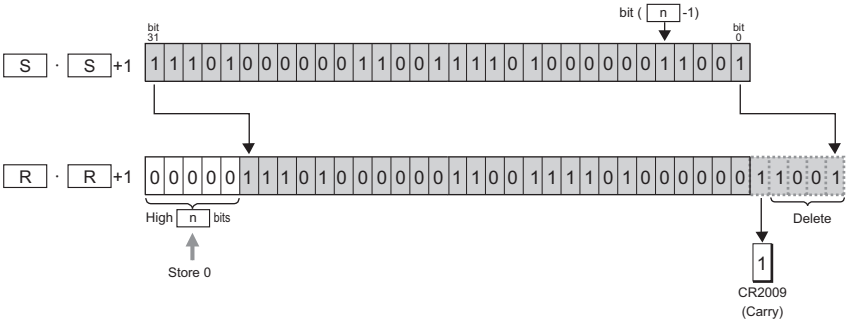
Description of Operation

The data stored in S is right shifted by the number of bits specified with n, and "0" is stored in the bits shifted right from the high-order bit.

- **S : When the operation destination is 16 bits**



- **S : When the operation destination is 32 bits**



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Shift Instructions".

● Example of format

DM2000.U = SRA (DM1000.U, 5)

Point

- **When [n] is "0", the value of [S] is stored in [R].**
- **When [S] is .U and [n] ≧ 17, and [S] is .D and [n] ≧ 33, the operation results will be 0.**

Sample Program

● To shift a bit data right

Programmed Script

DM2000.U = **SRA** (R1000.U, 10)

Operation Description

The data currently stored in R1000.U is shifted to the right by 10 bits.
The result is stored in DM2000.U.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

SLA Shift left (bit unit)

Return value*1 = SLA (Operation destination, number of left shifts)

Shift left A

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Operation destination	Specifies the operation destination or the operation destination data.	.U	.U	.D	.D	—	—	—	—	○	○	○
n	Number of left shifts	Specifies the storage location for the number of left shifts or the number of left shifts.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	SLA function type is identical to the 1st argument.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.
*3 The ".DF" suffix cannot be specified for device Z.

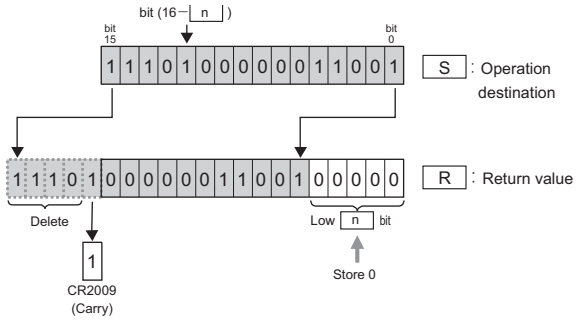
Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

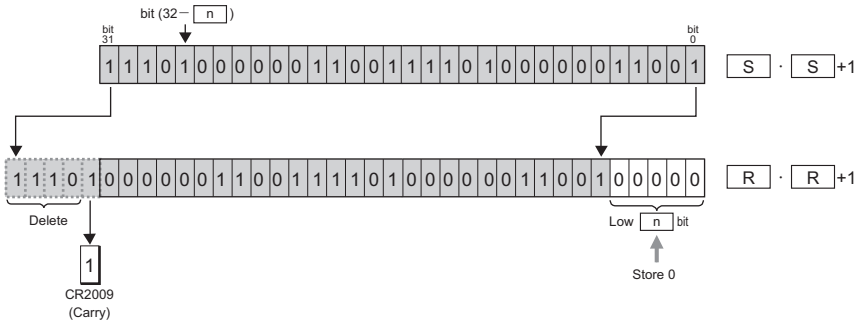
Description of Operation

The data stored in S is left shifted by the number of bits specified with n, and "0" is stored in the bits shifted left from the low-order bit.

- **S : When the operation destination is 16 bits**



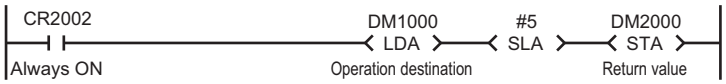
- **S : When the operation destination is 32 bits**



"About operation flag (CR2009 to CR2012)", page 1-18
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Shift Instructions".

● Example of format

DM2000.U = **SLA** (DM1000.U, 5)



Point

- **When [n] is "0", the value of [S] is stored in [R].**
- **When [S] is .U and [n] ≧ 17, and [S] is .D and [n] ≧ 33, the operation results will be 0.**

Sample Program

● To shift a bit data left

Programmed Script

DM2000.U = **SLA** (MR1000.U, 10)

Operation Description

The data currently stored in MR1000.U is shifted to the left by 10 bits.
The result is stored in DM2000.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

Hint

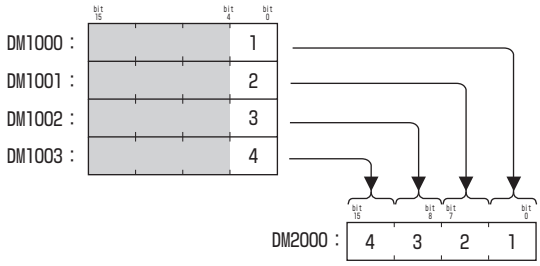
The SLA function is used, for example, to disperse and unite data stored in word data, etc. as it can shift any number of bits to the left.

Programmed Script

DM2000 = ANDA (DM1000, $F)
DM2000 = ORA (DM1000, **SLA** (ANDA (DM1001, $F), 4))
DM2000 = ORA (DM1000, **SLA** (ANDA (DM1002, $F), 8))
DM2000 = ORA (DM1000, **SLA** (ANDA (DM1003, 12))

Operation Description

The BCD data stored in DM1000, DM1001, DM1002, and DM1003 is united as 4-digit BCD one digit at a time from the low-order bit, and stored in DM2000.



Reference

The above script functions in an equivalent way to data conversion instruction UNIN (nibble unit).

"ANDA function", page 7-22
"ORA function", page 7-24
"UNIN function", page 7-94
KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Conversion Instructions" "UNIN Instruction"

ASRA Arithmetic right shift A (bit unit)

Return value*1 = ASRA (Operation destination, number of right shifts)

Arithmetic shift right A

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Operation destination	Specifies the operation destination or the operation destination data.	.U	.S	.D	.L	—	—	—	—	○	○	○
n	Number of arithmetic right shifts	Specifies the storage location for the number of right shifts or the number of right shifts.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	ASRA function type is identical to the 1st argument.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

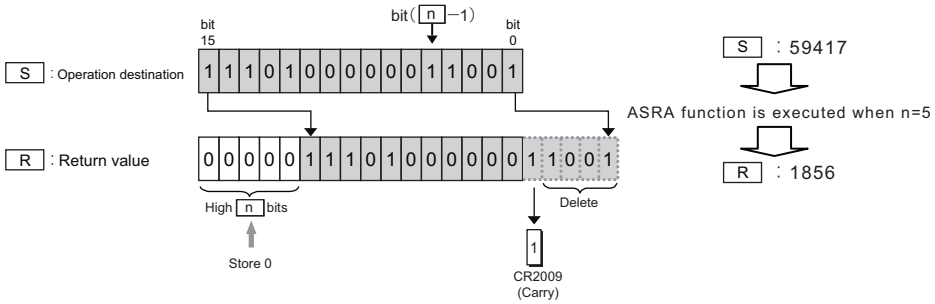
*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.
*3 The ".DF" suffix cannot be specified for device Z.

Point
- **The ASRA function can be used only with KV STUDIO Ver.6.00 or higher.**
- **This function can be used only with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or later and KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

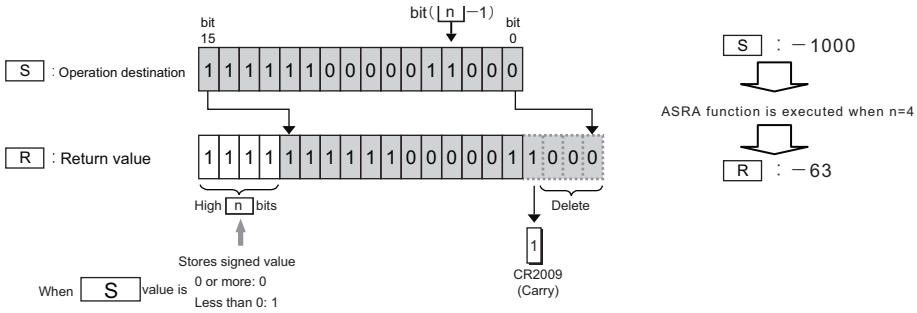
Description of Operation

The data stored in S is right shifted by the number of bits specified with n. When S ≧ 0, "0" is stored in the bits shifted right from the high-order bit, and "1" is stored when S < 0.

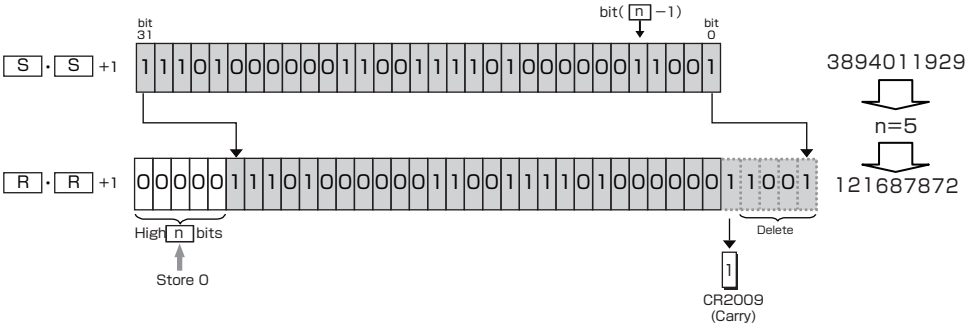
- **S : When the operation destination is unsigned 16-bit data**



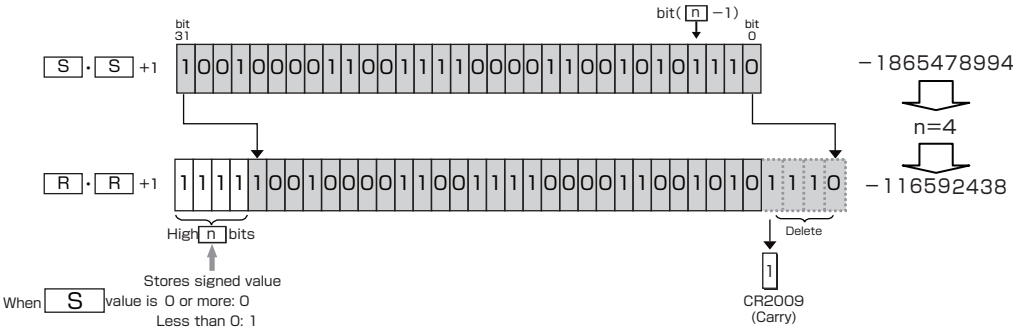
- **S : When the operation destination is signed 16-bit data**



- S : **When the operation destination is unsigned 32-bit data**



- S : **When the operation destination is signed 32-bit data**

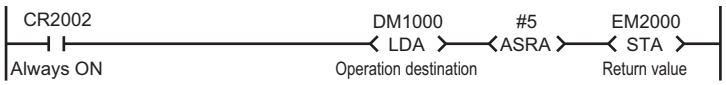


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Shift Instructions".

● Example of format

DM2000.U = ASRA (DM1000.U, 5)



Point

- **When n is "0", the value of S is stored in R.**
- **When S is .U and n ≧ 17, and S is .D and n ≧ 33, the operation results will be 0.**

Sample Program

Programmed Script

DM2000.U = **ASRA** (R1000.U, 10)

Operation Description

The data currently stored in R1000.U is shifted to the right by 10 bits.
The result is stored in DM2000.U.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ASLA Arithmetic left shift (bit unit)

Return value*1 = ASLA (Operation destination, number of left shifts)

Arithmetic shift left A

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Operation destination	Specifies the operation destination or the operation destination data.	.U	.S	.D	.L	—	—	—	—	○	○	○
n	Number of arithmetic left shifts	Specifies the storage location for the number of left shifts or the number of left shifts.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	ASLA function type is identical to the 1st argument.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

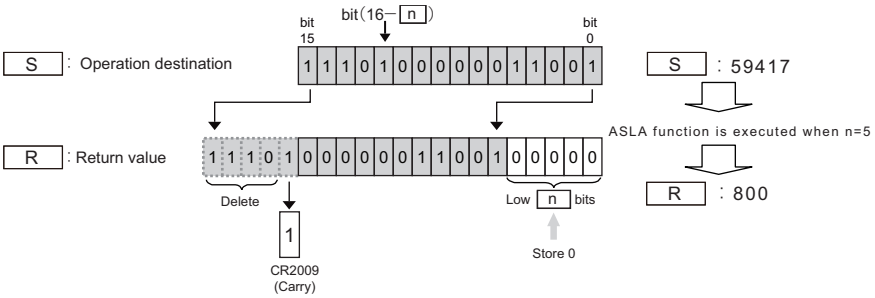
*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.
*3 The ".DF" suffix cannot be specified for device Z.

Point
- **The ASLA function can be used only with KV STUDIO Ver.6.00 or higher.**
- **This function can be used only with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or later and KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

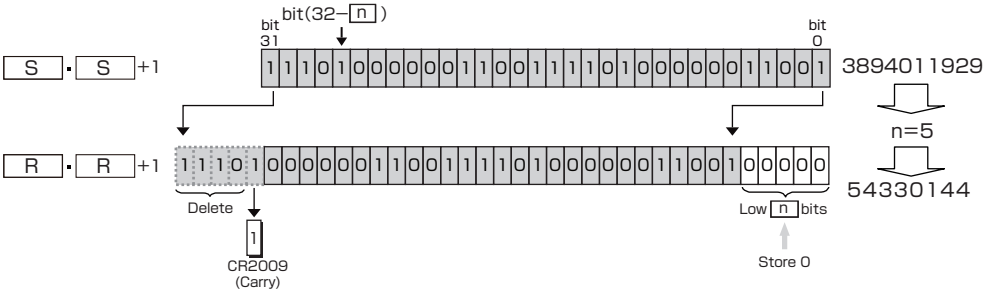
Description of Operation

The data stored in S is left shifted by the number of bits specified with n, and "0" is stored in the bits shifted left from the low-order bit.

- **S : When the operation destination is 16-bit data**



- **S : When the operation destination is 32-bit data**



"About operation flag (CR2009 to CR2012)", page 1-18
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Shift Instructions".

● **Example of format**

DM2000.U = **ASLA** (MR1000.U, 5)

Point

- **When [n] is "0", the value of [S] is stored in [R].**
- **When [S] is .U and [n] ≧ 17, and [S] is .D and [n] ≧ 33, the operation results will be 0.**

Sample Program

Programmed Script

DM2000.U = **ASLA** (R1000.U, 10)

Operation Description

The data currently stored in R1000.U is shifted to the left by 10 bits.
The result is stored in DM2000.U.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

RRA Right rotate (bit unit, with carry)

Return value*1 = RRA (Operation destination, number of right rotates)

Rotate right A

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Operation destination	Specifies the operation destination or the operation destination data.	.U	.U	.D	.D	—	—	—	—	○	○	○
n	Number of right rotates	Specifies the storage location for the number of right rotates or the number of right rotates.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	RRA function type is identical to the 1st argument.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

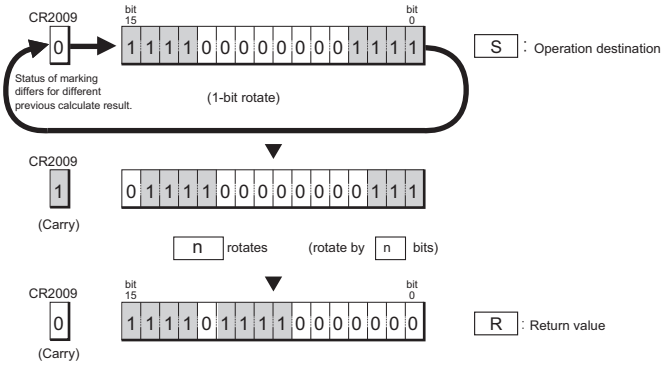
Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

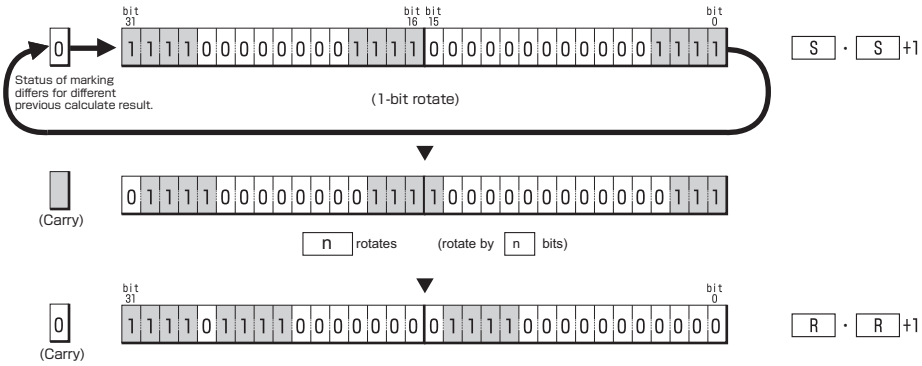
Description of Operation

The data stored in S is rotated to the right by the number of bits specified with n with carry flag (CR2009).

- **S : When the operation destination is 16 bits**



- S : **When the operation destination is 32 bits**

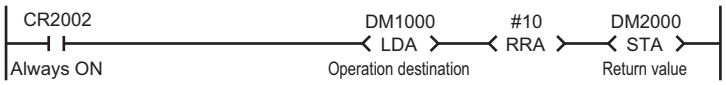


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Shift Instructions".

● Example of format

DM2000.U = **RRA** (DM1000.U, 10)



Point **When the data is rotated to the right, the LSB is stored in carry flag (CR2009).**

Sample Program

● To rotate a bit data right

Programmed Script: DM2000 = **RRA** (R1000.U, 10)

Operation Description: The data currently stored in R1000.U is rotated to the right by 10 bits.
The result is stored in DM2000.

Hint

With the SRA function, the MSB becomes "0" at each shift. However, with the RRA function, the LSB is shifted around to the MSB via the carry flag.

Programmed Script: R5000.D = **RRA** (R5000.D, 1)

Operation Description: Output relays R5000 to R5115 output so that the current output state is repeatedly moving while being rotated by one bit at a time to the right at each scan.

Reference: The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

RLA Left rotate (bit unit, with carry)

Return value*1 = RLA (Operation destination, number of left rotates)

Rotate left A

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Operation destination	Specifies the operation destination or the operation destination data.	.U	.U	.D	.D	—	—	—	—	○	○	○
n	Number of left rotates	Specifies the storage location for the number of left rotates or the number of left rotates.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	RLA function type is identical to the 1st argument.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

Point

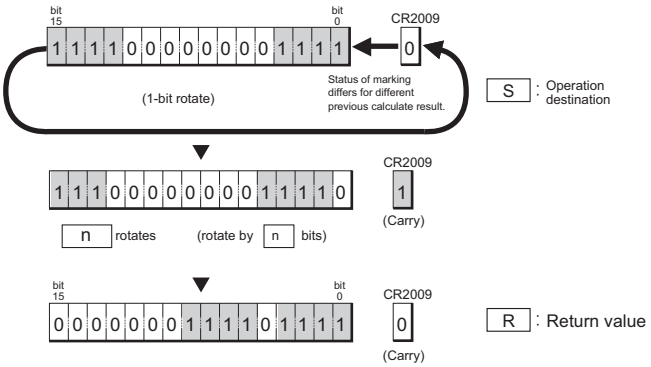
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

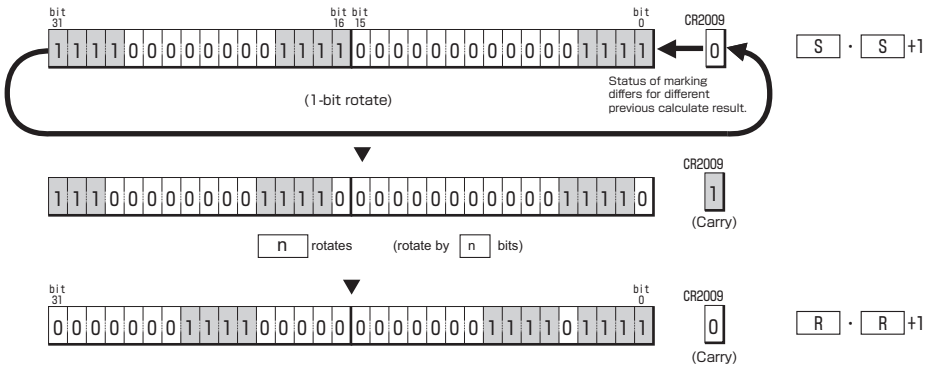
Description of Operation

The data stored in S is rotated to the left by the number of bits specified with n with carry flag (CR2009).

- **S : When the operation destination is 16 bits**



- S : **When the operation destination is 32 bits**

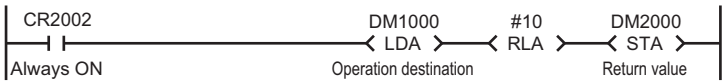


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Shift Instructions".

● Example of format

DM2000.U = **RLA** (DM1000.U, 10)



Point **When the data is rotated to the left, the MSB is stored in carry flag (CR2009).**

Sample Program

● To rotate a bit data left

Programmed Script: DM2000.U = **RLA** (R1000.U, 10)

Operation Description: The data currently stored in R1000.U is rotated to the left by 10 bits.
The result is stored in DM2000.

Hint

With the SLA function, the LSB becomes "0" at each shift. However, with the RLA function, the MSB is shifted around to the LSB via the carry flag.

Programmed Script: R5000.D = **RLA** (R5000.D, 1)

Operation Description: Output relays R5000 to R5031 output so that the current output state is repeatedly moving while being rotated by one bit at a time to the left at each scan.

Reference: The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

RRNCA Right rotate (bit unit, without carry)

Rotate right A without a carry

Return value*1 = RRNCA (Operation destination, number of right rotates)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Operation destination	Specifies the storage location for the operation destination.	.U	.U	.D	.D	—	—	—	—	○	○	○
n	Number of right rotates*2	Specifies the number of right rotates or its storage location.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	RRNCA function type is identical to the 1st argument.* 3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in internal registers.

*2 When a target has been specified to n, operation is as follows when the value of n is 16 or more (when the operation destination has a .U suffix) or 32 or more (when the operation destination has a .D suffix).
In the case of a .U suffix instruction, the remainder value obtained by dividing the value of n by 16 is taken as the number of bits to rotate.
In the case of a .D suffix instruction, the remainder value obtained by dividing the value of n by 32 is taken as the number of bits to rotate.

*3 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*4 The ".DF" suffix cannot be specified for device Z.

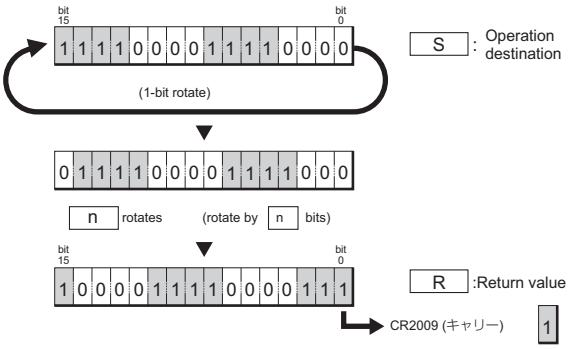
Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

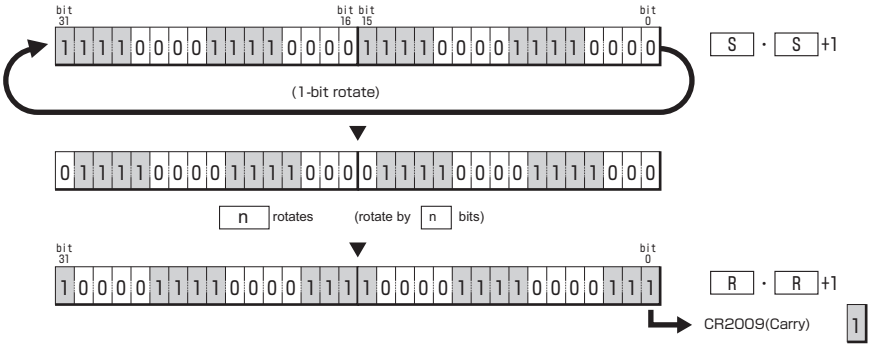
Description of Operation

The data stored in S is rotated to the right by the number of bits specified with n without a carry, and the result is stored in R.

- **S : When the operation destination is 16 bits**



- S : **When the operation destination is 32 bits**



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Shift Instructions".

Reference

The state of the LSB as a result of having performed a rotate is entered to the carry flag (CR2009).

● Example of format

DM2000.U = **RRNCA** (DM1000.U, 10)

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2100.D = TBIN (**RRNCA** (DM1100.D, 16))

Operation Description

The 8-digit BCD data currently stored in DM1100.D is rotated to the right without a carry by 16 bits, the upper and lower words (upper 4 digits and lower 4 digits of the 8-digit BCD) are swapped, and the result is converted to 32-bit BIN data and stored in DM2000.D.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "TBIN function", page 7-82
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

RLNCA Left rotate (bit unit, without carry)

Rotate left A without a carry

Return value*1 = RLNCA (Operation destination, number of left rotates)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Operation destination	Specifies the storage location for the operation destination.	.U	.U	.D	.D	—	—	—	—	○	○	○
n	Number of left rotates*2	Specifies the number of left rotates or its storage location.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	RLNCA function type is identical to the 1st argument.* 3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in internal registers.

*2 When a target has been specified to n, operation is as follows when the value of n is 16 or more (when the operation destination has a .U suffix) or 32 or more (when the operation destination has a .D suffix).
In the case of a .U suffix instruction, the remainder value obtained by dividing the value of n by 16 is taken as the number of bits to rotate.
In the case of a .D suffix instruction, the remainder value obtained by dividing the value of n by 32 is taken as the number of bits to rotate.

*3 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*4 The ".DF" suffix cannot be specified for device Z.

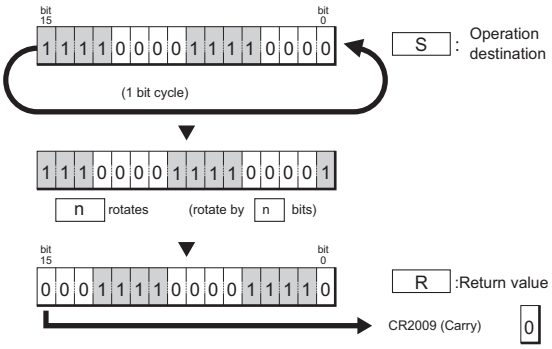
Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

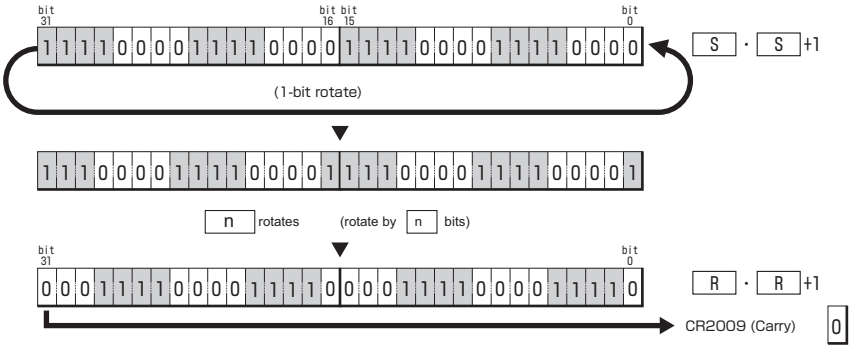
Description of Operation

The data stored in S is rotated to the left by the number of bits specified with n without a carry, and the result is stored in R.

- **S : When the operation destination is 16 bits**



- S : **When the operation destination is 32 bits**



"About operation flag (CR2009 to CR2012)", page 1-18

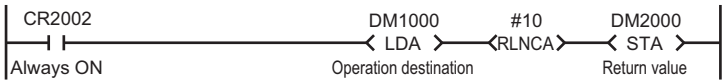
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Shift Instructions".

Reference

The state of the MSB as a result of having performed a rotate is entered to the carry flag (CR2009).

● Example of format

DM2000.U = **RLNCA** (DM1000.U, 10)



Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2100.D = TBIN (**RLNCA** (DM1100.D, 16))

Operation Description

The 8-digit BCD data currently stored in DM1100.D is rotated to the left without a carry by 16 bits, the upper and lower words (upper 4 digits and lower 4 digits of the 8-digit BCD) are swapped, and the result is converted to 32-bit BIN data and stored in DM2000.D.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "TBIN function", page 7-82
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

WSR Shift word device right

Shift word right

WSR ([execution condition,][*1] operation destination, number of operation range data, number of shifts)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Operation destination*2	Specifies the operation destination or the operation destination data.	.U	.U	.D	.D	.D	—	—	—	—	○	—
n1	Number of operation range data	Specifies the storage location for the number of data in the range to be shifted right or the number of right shift range data.	.U	.U	.U	.U	—	—	—	—	○	○	○
n2	Number of shifts	Specifies the storage location for the number of right shifts or the number of right shifts.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

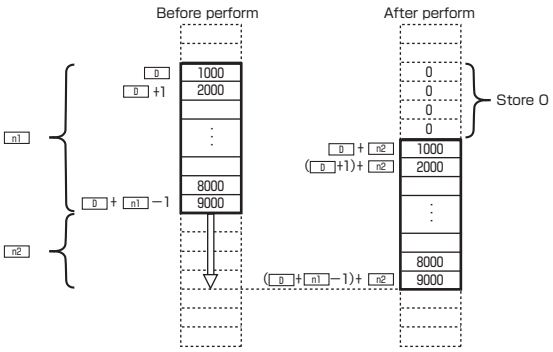
*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When bit data is specified, 16/32 continuous bits are processed.
When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

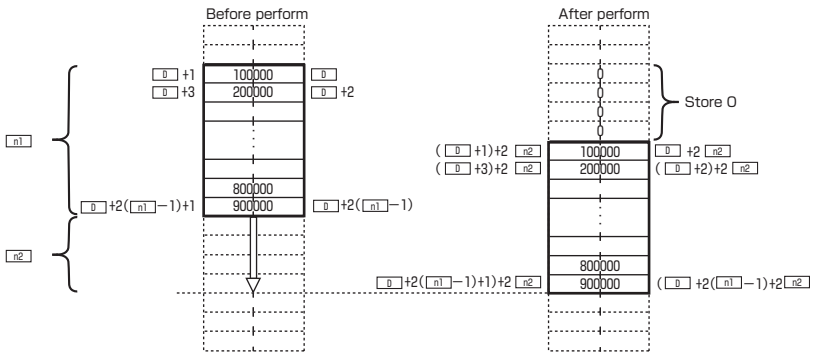
Description of Operation

When the execution condition is established (TRUE), data (16-bit/32-bit) stored in the range of the number of data specified by n1 starting from D is shifted to the right (towards the larger device No. / array subscript) by the number of data specified with n2.
0 is stored in the part where the shift source range does not overlap with the shift target range.

- **D : When the operation destination is 16 bits**



- **D : When the operation destination is 32 bits**



"About operation flag (CR2009 to CR2012)", page 1-18

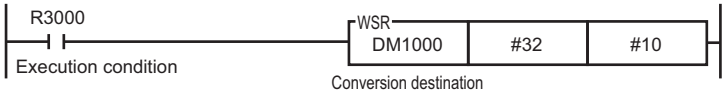
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Shift Instructions".

Reference

When the operation destination is 32-bit data, the number of operation range data and number of shifts becomes "number of specified data × 2 words".

● Example of format

WSR (R3000, DM1000.U, 32, 10)



Point

- **When n1 = 0, the function is not executed.**
- **When n2 = 0, the operation result does not change with before the operation was executed.**

Sample Program

● When this function is used in combination with other functions

Programmed Script

WSR (DM3000, EM3000, ANDA (EM3001, $F))

Operation Description

The 16-bit data currently stored in range of words specified by EM3000 is shifted to the right by the number of words specified by the lower 4 bits (extracted) of EM3001 starting from DM3000 at every scan (no execution condition).
"0" is stored in sections in the shift source range that do not straddle the shift destination range

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "ANDA function"
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

WSL Shift word device left

Shift word left

WSL ([execution condition,][*1] operation destination, number of operation range data, number of shifts)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Operation destination[*2]	Specifies the start of the range to shift left.	.U	.U	.D	.D	.D	—	—	—	—	○	—
n1	Number of operation range data	Specifies the storage location for the number of data in the range to be shifted left or the number of left shift range data.	.U	.U	.U	.U	—	—	—	—	○	○	○
n2	Number of shifts	Specifies the storage location for the number of left shifts or the number of left shifts.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

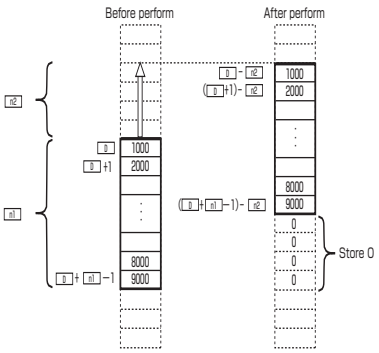
*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When bit data is specified, 16/32 continuous bits are processed.
When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

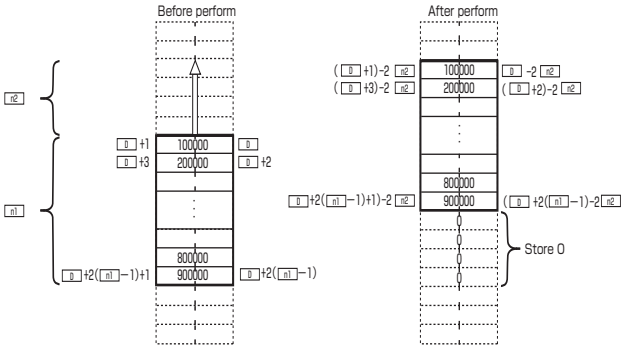
Description of Operation

When the execution condition is established (TRUE), data (16-bit/32-bit) stored in the range of the number of data specified by n1 starting from D is shifted to the left (towards the smaller device No. / array subscript) by the number of data specified with n2.
"0" is stored in sections in the shift source range that do not straddle the shift destination range

- **D : When the operation destination is 16 bits**



- D : **When the operation destination is 32 bits**



"About operation flag (CR2009 to CR2012)", page 1-18

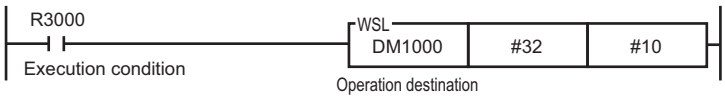
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Shift Instructions".

Reference

When the operation destination is 32-bit data, the number of operation range data and number of shifts becomes "number of specified data × 2 words".

● Example of format

WSL (R3000, DM1000.U, 32, 10)



Point

- **When n1 = 0, the function is not executed.**
- **When n2 = 0, the operation result does not change with before the operation was executed.**

Sample Program

● When this function is used in combination with other functions

Programmed Script

WSL (DM3000, EM3000, ANDA (EM3001, $F))

Operation Description

The 16-bit data currently stored in range of words specified by EM3000 is shifted to the left by the number of words specified by the lower 4 bits (extracted) of EM3001 starting from DM3000 at every scan (no execution condition).
"0" is stored in sections in the shift source range that do not straddle the shift destination range

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "ANDA function"
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

BSR Shift bit device right

Bit shift right

BSR ([execution condition,][*1] operation destination, number of operation range destinations, number of shifts)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Operation destination*2*3	Specifies the start of the range to shift right.	—	—	—	—	—	—	.B	—	—	○	—
n1	Number of operation range data	Specifies the storage location for the number of destinations in the range to be shifted right or the number of right shift range destinations.	.U	.U	.U	.U	—	—	—	—	○	○	○
n2	Number of shifts	Specifies the storage location for the number of right shifts or the number of right shifts.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When bit data has been specified, the program will function normally even if the bit data straddles channels. When word data has been specified, the LSB of n1 word data is shifted. Bits other than the LSB do not change.

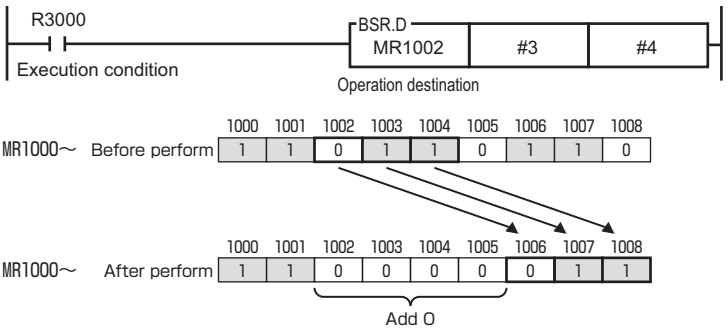
*3 T, C, CR, CTC, CTH, Z, CM cannot be used.

Description of Operation

When the execution condition is established (TRUE), the state of n1 devices starting from D is shifted to the right (towards the larger device No. / array subscript) by n2 shifts. When D is word data, only the LSB of each word device is targeted for shifting. (Bits other than the LSB do not change.) "OFF" is stored in sections (LSB in the case of word data) in the shift source range that do not straddle the shift destination range.

● Example of format

BSR (R3000, MR1002, 3, 4)



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Shift Instructions".

Point

- **When n2 = 0, the operation result does not change with before the operation was executed.**
- **The function is not executed when either of the following conditions is established:**
  - **When the shift destination exceeds the device or variable range**
  - **When n1 is "0".**

Sample Program

● When this function is used in combination with other functions

Programmed Script

BSR (DM3000.B, EM3000, ANDA (EM3001, $F))

Operation Description

The state of the LSBs of the number of word data specified by EM3000 starting from DM3000 is shifted to the right by the count specified by the lower four bits (extracted) of EM3001 at every scan (no execution condition).
OFF is stored in the least significant bit of the shift source range that does not overlap with the shift target range.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "ANDA function", page 7-22
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

BSL Shift bit device left

Bit shift left

BSL ([execution condition,][*1] operation destination, number of operation range destinations, number of shifts)

Argument/Return Value		Description	Recognized Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Operation destination*2*3	Specifies the start of the range to shift left.	—	—	—	—	—	—	.B	—	—	○	—
n1	Number of operation range data	Specifies the storage location for the number of destinations in the range to be shifted left or the number of left shift range destinations.	.U	.U	.U	.U	—	—	—	—	○	○	○
n2	Number of shifts	Specifies the storage location for the number of left shifts or the number of left shifts.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When bit data has been specified, the program will function normally even if the bit data straddles channels. When word data has been specified, the LSB of n1 word data is shifted. Bits other than the LSB do not change.

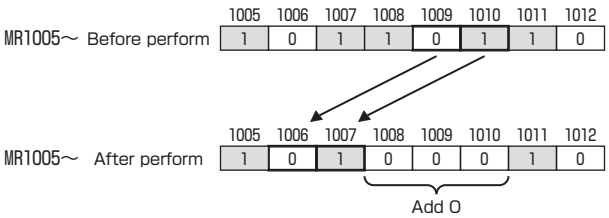
*3 T, C, CR, CTC, CTH, Z, CM cannot be used.

Description of Operation

When the execution condition is established (TRUE), the state of n1 devices starting from D is shifted to the left (towards the smaller device No. / array subscript) by n2 shifts. D is word data, only the LSB of each word device is targeted for shifting. (Bits other than the LSB do not change.)
"OFF" is stored in sections (LSB in the case of word data) in the shift source range that do not straddle the shift destination range.
"0" is stored in sections in the shift source range that do not straddle the shift destination range

● Example of format

BSL (R3000, MR1009, 2, 3)



"About operation flag (CR2009 to CR2012)", page 1-18
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Shift Instructions".

Point

- **When n2 = 0, the operation result does not change with before the operation was executed.**
- **The function is not executed when either of the following conditions is established:**
  - **When D − n1 >0.**
  - **When n1 is "0".**

Sample Program

● When this function is used in combination with other functions

Programmed Script

BSL (DM3000.B, EM3000, ANDA (EM3001, $F))

Operation Description

The state of the LSBs of the number of word data specified by EM3000 starting from DM3000.B is shifted to the left by the count specified by the lower four bits (extracted) of EM3001 at every scan (no execution condition).
OFF is stored in the least significant bit of the shift source range that does not overlap with the shift target range.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "ANDA function", page 7-22
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

LIMIT Upper/lower limit value control

Return value*1 = LIMIT (Lower limit value, upper limit value, input value)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Lower limit*2 (min. output threshold value)	Specifies the lower limit value or the storage location for the lower limit value.	.U	.S	.D	.L	—	—	—	—	○	○	○
S2	Upper limit*2 (max. output threshold value)	Specifies the upper limit value or the storage location for the upper limit value.	Handled as same type as S1.				—	—	—	—	○	○	○
S3	Input value*2	Specifies the upper / lower limit control input value or the storage location for the input value.	Handled as same type as S1.				—	—	—	—	○	○	○
R	Return value	LIMIT function type is identical to the 1st argument.*3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in working memory.

*2
- When bit data is specified, 16/32 continuous bits are processed. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in 16/32 bits straddling the next channel.
- When specifying CTH, current value is processed. When using CTC, process setting value.

*3 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

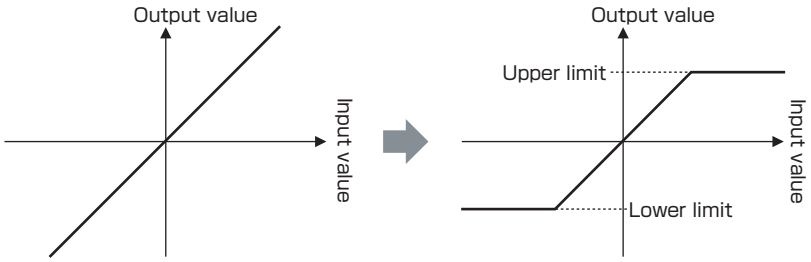
*4 The ".DF" suffix cannot be specified for device Z.

Point
- **When using KV-1000, please use system macro "LIMITS" and "LIMITL".**
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

Upper/lower limit control is performed for S3 via S2 and S1, and the result is stored in R.



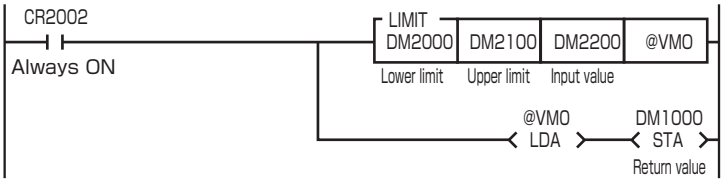
"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Control Instructions".

● Example of format

- [S1] : When lower limit value is 16 bits (.U/.S)

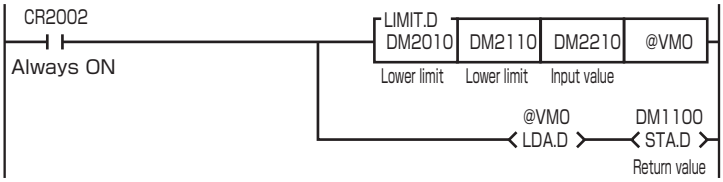
DM1000.U = LIMIT (DM2000.U, DM2100, DM2200)



Condition	Value stored in [R]
[S1] (Lower limit value) > [S3] (Input value)	[S1] (Lower limit value)
[S2] (Upper limit value) < [S3] (Input value)	[S2] (Upper limit value)
[S1] (Lower limit value) ≦ [S3] (Input value) ≦ [S2] (Upper limit value)	[S3] (Input value)

- [S1] : When lower limit value is 32 bits (.D/.L)

DM1100.D = LIMIT (DM2010.D, DM2110, DM2210)



Condition	Value stored in [R · R+1]
[S1 · S1+1] > [S3 · S3+1] (Lower limit value) (Input value)	[S1 · S1+1] (Lower limit value)
[S2 · S2+1] > [S3 · S3+1] (Upper limit value) (Input value)	[S2 · S2+1] (Upper limit value)
[S1 · S1+1] ≦ [S3 · S3+1] ≦ [S2 · S2+1] (Lower limit value) (Input value) (Upper limit value)	[S3 · S3+1] (Input value)

BANDC Dead band control

Return value*1 = BANDC (Lower limit for dead band, upper limit for dead band, input value)

Dead band control

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Dead band (non-output range) lower limit value*2	Specifies the lower limit value or the storage location for the lower limit value.	.S	.S	.L	.L	—	—	—	—	○	○	○
S2	Dead band (non-output range) upper limit value*2	Specifies the upper limit value or the storage location for the upper limit value.	Handled as same type as S1.				—	—	—	—	○	○	○
S3	Input value*2	Specifies the dead band control input value or the storage location for the input value.	Handled as same type as S1.				—	—	—	—	○	○	○
R	Return value	BANDC function type is identical to the 1st argument.*3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in working memory.

*2
- When bit data is specified, 16/32 continuous bits are processed. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in 16/32 bits straddling the next channel.
- When specifying CTH, current value is processed. When using CTC, process setting value.

*3 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

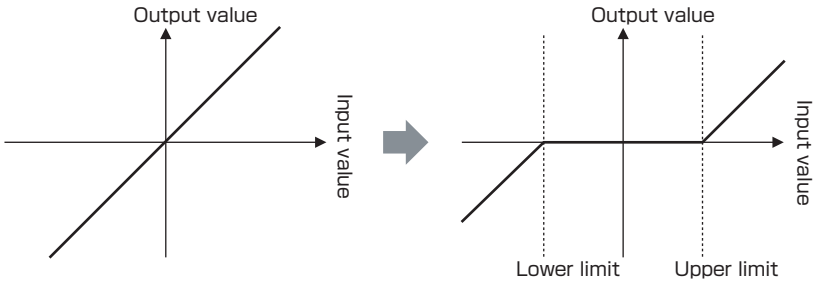
*4 The ".DF" suffix cannot be specified for device Z.

Point
- **When using KV-1000, please use system macro "BANDS" and "BANDL".**
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

Dead band control is performed for S3 via S2 and S1, and the result is stored in R.



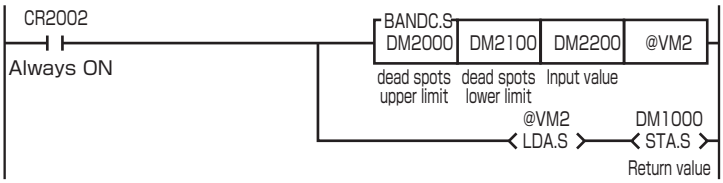
"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Control Instructions".

● Example of format

- [S1] : When dead band (no output area) lower limit value is 16 bits (.S)

DM1000.S = BANDC (DM2000.S, DM2100, DM2200)



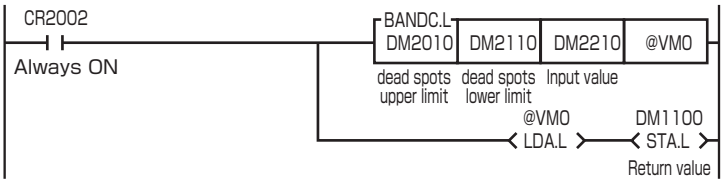
Condition	Value stored in [R]
[S1] (Lower limit value) > [S3] (Input value)	[S3] (Input value) − [S1] (Lower limit value)
[S2] (Upper limit value) < [S3] (Input value)	[S3] (Input value) − [S2] (Upper limit value)
[S1] (Lower limit value) ≦ [S3] (Input value) ≦ [S2] (Upper limit value)	0

∗The value with 16-bit symbol within the BIN data range is stored in [R].

- [R] ≦ −32768 → [R] = −32768
- [R] ≧ +32767 → [R] = +32767

- [S1] : When dead band (no output area) lower limit value is 32 bits (.L)

DM1100.L = BANDC (DM2010.L, DM2110, DM2210)



Condition	Value stored in [R · R+1]
[S1 · S1+1] > [S3 · S3+1] (Lower limit value) (Input value)	[S1 · S1+1] (Lower limit value)
[S2 · S2+1] > [S3 · S3+1] (Upper limit value) (Upper limit value)	[S2 · S2+1] (Upper limit value)
[S1 · S1+1] ≦ [S3 · S3+1] ≦ [S2 · S2+1] (Lower limit value) (Input value) (Upper limit value)	[S3 · S3+1] (Input value)

∗The value with 32-bit symbol within the BIN data range is stored in [R/R+1].

- [R/R+1] ≦ −2147483648 → [R/R+1] = −2147483648
- [R/R+1] ≧ +2147483647 → [R/R+1] = +2147483647

ZONE Zone control

Return value*1 = ZONE (Negative bias, positive bias, input value)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Negative bias*2	Specifies the negative bias value added to the input value or the storage location for the negative bias value.	.S	.S	.L	.L	—	—	—	—	○	○	○
S2	Positive bias*2	Specifies the positive bias value added to the input value or the storage location for the positive bias value.	Handled as same type as S1.				—	—	—	—	○	○	○
S3	Input value*2	Specifies the zone control input value or the storage location for the input value.	Handled as same type as S1.				—	—	—	—	○	○	○
R	Return value	ZONE function type is identical to the 1st argument.*3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in working memory.

*2
- When bit data is specified, 16/32 continuous bits are processed. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in 16/32 bits straddling the next channel.
- When specifying CTH, current value is processed. When using CTC, process setting value.

*3 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*4 The ".DF" suffix cannot be specified for device Z.

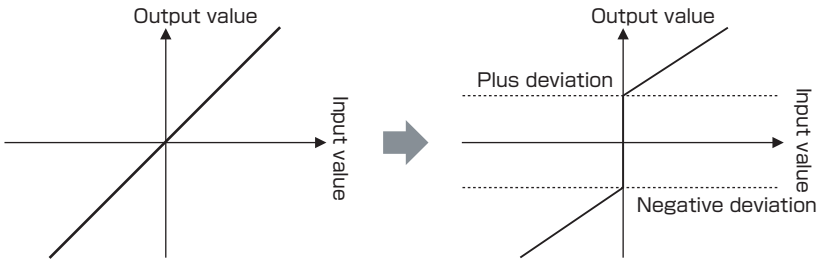
Point

- **ZONE function is special function of KV-5500/5000/3000. When using KV-1000, please use system macro "ZONES" "ZONEL".**
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

Zone control is performed for S3 via S2 and S1, and the result is stored in R.



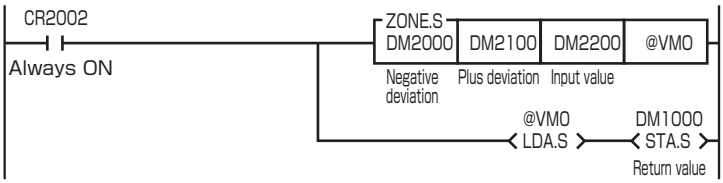
"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Control Instructions".

● Example of format

- [S1] : When negative bias value is 16 bits (.S)

DM1000.S = **ZONE** (DM2000.S, DM2100, DM2200)



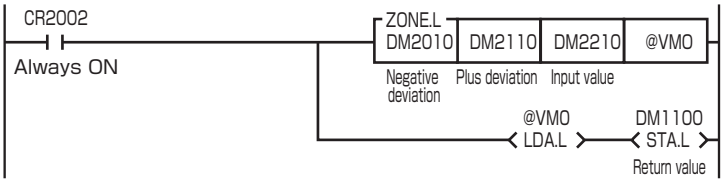
Condition	Value stored in [R]
[S3] (Input) ＜0	[S3] (Input value) + [S1] (Negative bias)
[S3] (Input) ＝0	0
[S3] (Input) ＞0	[S2] (Input value) + [S2] (Positive bias)

∗The value with 16-bit symbol within the BIN data range is stored in [R].

- [R] ≦ −32768 → [R] = −32768
- [R] ≧ +32767 → [R] = +32767

- [S1] : When negative bias value is 32 bits (.L)

DM1100.L = **ZONE** (DM2010.L, DM2110, DM2210)



Condition	Value stored in [[R]·[R]+1]
[[S1]·[S1]+1] > [[S3]·[S3]+1] (Negative bias) (Input value)	[[S1]·[S1]+1] (Negative bias)
[[S2]·[S2]+1] > [[S3]·[S3]+1] (Positive bias) (Input value)	[[S2]·[S2]+1] (Positive bias)
[[S1]·[S1]+1] ≦ [[S3]·[S3]+1] ≦ [[S2]·[S2]+1] (Negative bias) (Input value) (Positive bias)	[[S3]·[S3]+1] (Input value)

∗The value with 32-bit symbol within the BIN data range is stored in [[R]/[R]+1].

- [[R]/[R]+1] ≦ −2147483648 → [[R]/[R]+1] = −2147483648
- [[R]/[R]+1] ≧ +2147483647 → [[R]/[R]+1] = +2147483647

APR Linear approximation (Scaling)

Return value*1 = APR (Input value, start of data table, number of data records)

Linear approximation

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Input value*2	Specifies the input value of polyline approximation or the storage location for the input value.	.U	.S	.D	.L	.F	*5 .DF	—	—	○	○	○
S2	Start of data table*3	Specifies the start to store the polyline data table.	Handled as same type as S1.				—	—	—	—	—	○	—
n	Number of data records	Specifies the number of polyline data records or the storage location for the number of records.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	APR function type is identical to the 1st argument.*4	.U	.S	.D	.L	.F	*5 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in working memory.
*2
- When bit data is specified, 16/32/64 continuous bits are processed. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16/32/64 bits straddling the next channel.
- When specifying CTH, current value is processed. When using CTC, process setting value.

*3 When S1 type is .F or .DF, T and C cannot be specified.
*4 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.
*5 The ".DF" suffix cannot be specified for device Z.

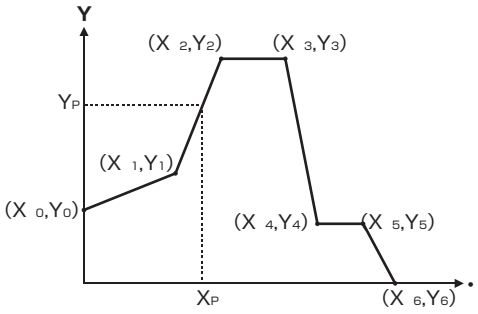
Point

- **When using KV-1000, please use system macro "APRS" "APRL" "APRF".**
- **The argument ".DF" can be used only with KV STUDIO Ver.9.0 or later.**
- **The argument ".DF" can be specified only with KV-8000 series CPU units and KV-7000 series CPU units with CPU function version 2.0 or later.**
- **The return value ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The return value ".DF" can be specified only with KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-7500/7300 CPU Function Version", page 3
"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

- S1 : When input value is 16 bits (.U/.S)
  Approximate calculation is performed for the "input value" of linear approximation stored in S1 according to the "polyline data table" stored with n records starting with S2.
  The linear approximate calculation searches in which range of polyline data table S1 is located, and the linear approximate calculation is performed at previous and next two points. The result is stored in R.



$$Y_P = \frac{Y_2 - Y_1}{X_2 - X_1}(X_P - X_1) + Y_1$$

Broken data form

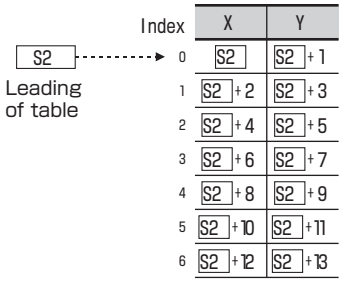
S2 Leading of table

Index	X	Y
0	0	10
1	8	13
2	14	20
3	20	20
4	23	6
5	26	6
6	28	0

n Number of records

Please remember store data of X in ascending order.

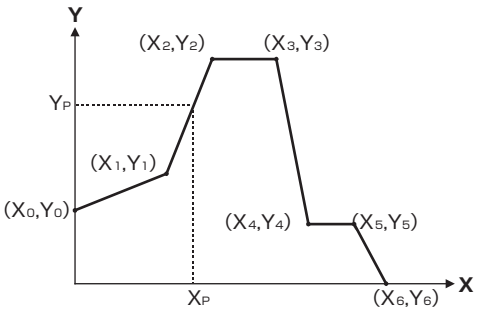
Please set X value of polyline data table according to ascending sequence. ($X_0 < X_1 \ldots < X_k \ldots < X_n$)
Please store the polyline data table from [S2] in the order of (X_0, Y_0), (X_1, Y_1)…(X_{n-1}, Y_{n-1}), (X_n, Y_n).



Please store the number of polyline data table records to [n] (number of records). In addition, [n] shall take the value more than 2.

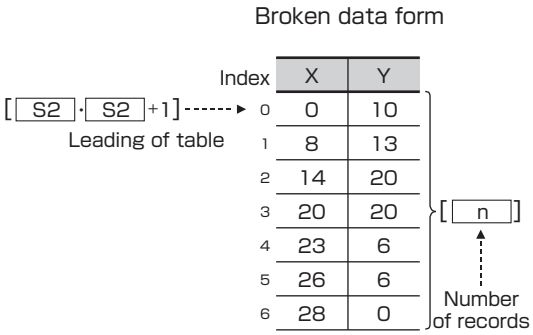
When [S1](input value)<X_0, Y_0 is stored in [R].
When [S1](input value)<X_n, Y_n is stored in [R].
When [S1](input value)<X_k, Y_k is stored in [R].

- [S1] : When input value is 32 bits (.D/.L)
  Approximate calculation is performed for the "input value" of linear approximation stored in [S1/S1+1] according to the "polyline data table" stored with [n] records starting with [S2/S2+1].
  The linear approximate calculation searches in which range of polyline data table [S1/S1+1] is located, and the linear approximate calculation is performed at previous and next two points.
  The calculation result is stored in [R/R+1].



$$Y_P = \frac{Y_2 - Y_1}{X_2 - X_1}(X_P - X_1) + Y_1$$

Broken data form



Please store the polyline data table from [S2/S2+1] in the order of (X_0, Y_0), (X_1, Y_1)…(X_{n-1}, Y_{n-1}), (X_n, Y_n).
Please store the number of polyline data table records to [n] (number of records).
In addition, [n] shall take the value more than 2.

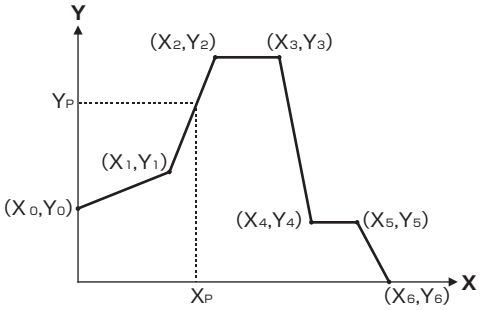
When [S1/S1+1]<X_0, Y_0 is stored in [R/R+1].
When [S1/S1+1]<X_n, Y_n is stored in [R/R+1].
When [S1/S1+1]<X_k, Y_k is stored in [R/R+1] (k=1~n).

- [S1] : When input value is 32 bits (.F)

  Approximate calculation (single-precision floating point type) is performed for the "input value" of linear approximation stored in [S1/S1+1] according to the "polyline data table" stored with [n] records starting with [S2/S2+1].

  The linear approximate calculation searches in which range of polyline data table [S1/S1+1] is located, and the linear approximate calculation is performed at previous and next two points.

  The calculation result is stored in [R/R+1].



$$Y_P = \frac{Y_2 - Y_1}{X_2 - X_1}(X_P - X_1) + Y_1$$

Broken data form

[S2·S2+1] → Leading of table

Index	X	Y
0	0	5.2
1	3.3	6.8
2	5.7	13.9
3	7.9	13.9
4	9.9	2.1
5	12.2	2.1
6	15.5	0

[n] Number of records

Please store the polyline data table from [S2/S2+1] in the order of (X_0, Y_0), (X_1, Y_1)…(X_{n-1}, Y_{n-1}), (X_n, Y_n).

Please set X value of polyline data table according to ascending sequence. ($X_0 < X_1 \ldots < X_k \ldots < X_n$)

Please store the number of polyline data table records to [n] (number of records).

In addition, [n] shall take the value more than 2.

When [S1/S1+1]$<X_0$, Y_0 is stored in [R/R+1].

When [S1/S1+1]$<X_n$, Y_n is stored in [R/R+1].

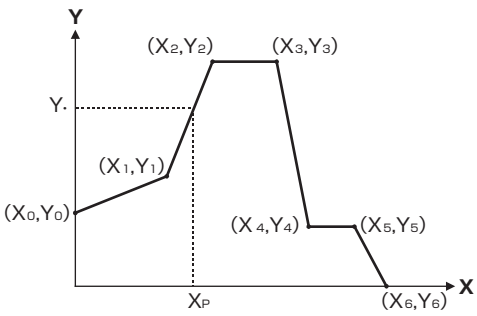
When [S1/S1+1]$=X_k$, Y_k is stored in [R/R+1] (k=1~n).

- [S1] : When input value is 64 bits (.DF)

  Approximate calculation (double-precision floating point type) is performed for the "input value" of linear approximation stored in [[S1]/[S1]+1/[S1]+2/[S1]+3] according to the "polyline data table" stored with [n] records starting with [[S2]/[S2]+1/[S2]+2/[S2]+3].

  The linear approximate calculation searches in which range of polyline data table [[S1]/[S1]+1/[S1]+2/[S1]+3] is located, and the linear approximate calculation is performed at previous and next two points.

  The calculation result is stored in [[R]/[R]+1/[R]+2/[R]+3].



$$Y_P = \frac{Y_2 - Y_1}{X_2 - X_1}(X_P - X_1) + Y_1$$

Broken data form

[[S2]·[S2]+1
·[S2]+2
·[S2]+3]
Leading of table

Index	X	Y
0	0	5.2
1	3.3	6.8
2	5.7	13.9
3	7.9	13.9
4	9.9	2.1
5	12.2	2.1
6	15.5	0

[[n]]
Number of records

Please store the polyline data table from [[S2]/[S2]+1/[S2]+2/[S2]+3] in the order of (X_0, Y_0), (X_1, Y_1)…(X_{n-1}, Y_{n-1}), (X_n, Y_n).

Please set X value of polyline data table according to ascending sequence. ($X_0 < X_1 \ldots < X_k \ldots < X_n$)

Please store the number of polyline data table records to [[n]] (number of records).

In addition, [[n]] shall take the value more than 2.

When [[S1]/[S1]+1/[S1]+2/[S1]+3]<X_0, Y_0 is stored in [[R]/[R]+1/[R]+2/[R]+3].

When [[S1]/[S1]+1/[S1]+2/[S1]+3]<X_n, Y_n is stored in [[R]/[R]+1/[R]+2/[R]+3].

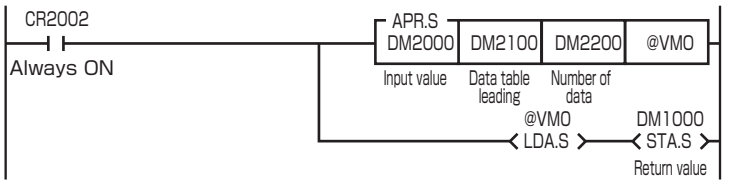
When [[S1]/[S1]+1/[S1]+2/[S1]+3]=X_k, Y_k is stored in [[R]/[R]+1/[R]+2/[R]+3] (k=1～n).

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Control Instructions".

● Example of format

DM1000.S = APR (DM2000.S, DM2100, DM2200)



Sample Program

● To use as a scaling function

Programmed Script

```
DM0.S=0
DM1.S=-1000
DM2.S=+255
DM3.S=+1000
DM100.S=APR(CM1630.s,DM0.S,2)
```

Operation Description

Scales the KV Nano Series analog volume CM1630 (0 to 255) to -1000 to +1000 and stores it in DM100.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

MEMO

RAMP Ramp signal

Ramp signal

RAMP ([execution condition,]*1 input value, change rate setting, time unit setting, output value)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Input value*2	Specifies the input value or the storage location for the input value (-32768 to 32767).	.S	.S	.S	.S	—	—	—	—	○	○	○
S2	Change rate setting*2	Specifies the change rate per time unit or the storage location for the change rate. (0 to 65535)	.U	.U	.U	.U	—	—	—	—	○	○	○
S3	Time unit setting*2*3	Specifies the time unit. (0 to 2)	.U	.U	.U	.U	—	—	—	—	○	○	○
D	Output value*2	Stores the results of the input value plus the inclination amount at the set interval.	.S	.S	.S	.S	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2
- When bit data is specified, 16 continuous bits are processed. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16 bits straddling the next channel.
- When specifying CTH, current value is processed. When using CTC, process setting value.
- T, C, CTH, CTC, and Z process the low-order 16 bits.

*3 Specify the time unit setting in the range of 0 to 2.

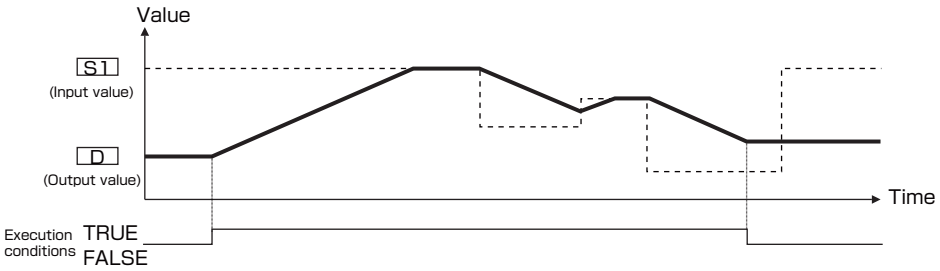
Explanation of S3 time unit setting

Value	Time unit
0	1 second unit
1	1 minute unit
2	1 hour unit

Point
- **The RAMP function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

When the execution condition is met (TRUE), the value of D is approached at the specified inclination (S2/S3) until it matches S1.

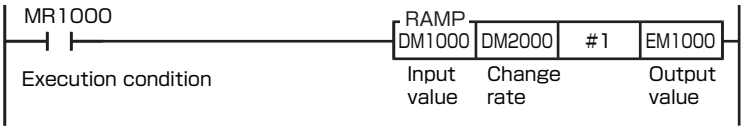


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Control Instructions".

● **Example of format**

RAMP (MR1000, DM1000.S, DM2000.U, 1, EM1000.S)



Point When S3 ≧ 3, the function is not executed.

Sample Program

Programmed Script

RAMP (DM1500.S, DM2500.U, 2, EM1500.S)

Operation Description

The output value stored in EM1500.S is changed toward the input value at the specified inclination (DM2000/time).

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

TPOUT Time proportional output

Time proportional output

TPOUT ([execution condition,]*1 ON time, cycle, output relay)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S1	ON time*2	Specifies the time to keep the output relay ON with a ms unit (0 to 65535).	.U	.U	.U	.U	—	—	—	—	○	○	○
S2	Cycle*2	Specifies the output relay ON/OFF cycle with a ms unit (1 to 65535).	.U	.U	.U	.U	—	—	—	—	○	○	○
D	Output relay*3	Specifies the target to be turned ON/ OFF.	—	—	—	—	—	—	.B	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

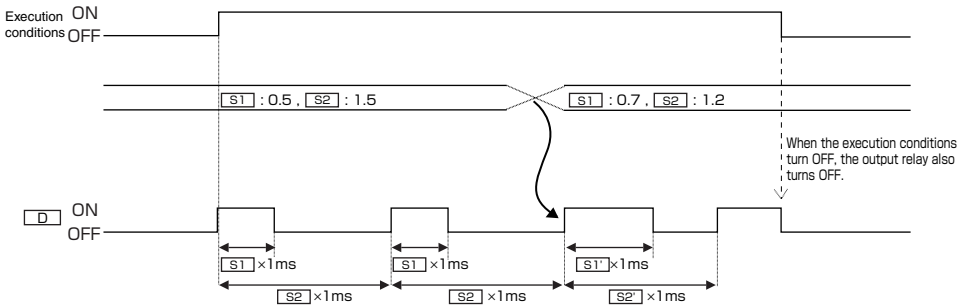
*2
- When bit data is specified, 16 continuous bits are processed. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16 bits straddling the next channel.
- When specifying CTH, current value is processed. When using CTC, process setting value.
- T, C, CTH, CTC, and Z process the low-order 16 bits.

*3
- When word data is specified, the LSB is the storage destination for the results. (Bits other than the LSB do not change.)
- Even if a relay other than the leading channel (R1002, R1005, etc.) is specified, the output will function correctly crossing to the next channel.
- CTC, CTH, Z cannot be used.

Point
- **The TPOUT function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, KV-5500/ 5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

When the execution condition is met (TRUE), D is turned ON for S1 (ms) at S2 (ms).

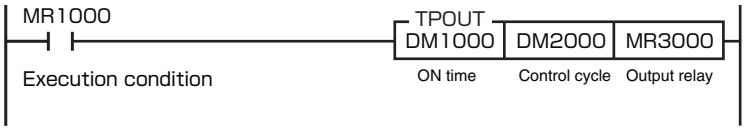


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Control Instructions".

● **Example of format**

TPOUT (MR1000, DM1000, DM2000, MR3000)



Point

- **The function is not executed when either of the following conditions is established:**
  - When S2 < S1
  - When S2 =0
- **An error within ±10μs+1 scan time occurs.**
- **When 0 is specified for S1, the function does not turn ON for even one scan.**
- **If writing is attempted during RUN, the cycle (S2) will be initialized, and the ON time (S1) will be counted after the write during RUN ends.**
- **This instruction cannot be used in interrupt programs or in the initialization module.**
- **Caution is required when using in a subroutine program, between the STP to STE instructions, between the STG to JMP/ENDS instructions or between the CJ (NCJ) to LABEL instructions.**
  Refer to "Cautions for using Timer Instruction" in the User's Manual of the CPU unit used.
- **Caution is required when using write during RUN.**
  Refer to "Writing during RUN" in the User's Manual of the CPU unit used.

Sample Program

Programmed Script

TPOUT (DM1500, EM1000, MR2000)

Operation Description

The MR2000 ON (DM1500ms) and OFF (EM1000-DM1500ms) are switched at the EM1000ms cycle at each scan (no execution conditions).

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

LLFLT Lead-lag filter

Lead-lag filter

LLFLT ([execution condition,]*1 input value, parameter, output value)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Input value*2*3	Specifies the filter input value or the storage location for the input value (−32768 to 32767).	.S	.S	.S	.S	—	—	—	—	○	○	○
D1	Parameter*2*4	Specifies the start to store the parameter.	.U	.U	.U	.U	—	—	—	—	—	○	—
D2	Output value*2	Stores the storage location for the output value (−32768 to 32767).	.S	.S	.S	.S	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When bit data is specified, 16 continuous bits are processed. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16 bits straddling the next channel.

*3
- When specifying CTH, current value is processed. When using CTC, process setting value.
- T, C, CTH, CTC and Z process the low-order 16 bits.

*4
- When bit data is specified, it occupies 48 continuous bits.
- For bit data, specify the start of the channel.
- When word data is specified, 3 continuous words are occupied.

Point
- **The LLFLT function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

When the execution condition is met (TRUE), the lead-lag operation is executed for S at D1, and stored in D2.

D2 is initialized with the value S at the rising edge of the execution condition.

D1 Parameter description

Parameter		Description
D1	Sampling cycle (T_S)	Specifies the cycle time for executing the instruction. (1 to 60000ms) Specify the smaller value than Lag/Lead time.
D1+1	Lag time (T1)	Specifies the time constant for the lag element. (0 to 65535ms) Lag is disabled if 0 is specified.
D1+2	Lead time (T2)	Specifies the time constant for the lead element. (0 to 65535ms) Lead is disabled if 0 is specified.

For details on lead-lag operation, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "LLFLT Instruction".

"About operation flag (CR2009 to CR2012)", page 1-18

● **Example of format**

LLFLT (MR1000, DM1000.S, EM1000, DM3000.S)

Point

- **If writing is attempted during RUN, the sampling cycle (D1 +1) will be initialized, and the output value (D2) will be initialized with the input value (S).**
- **This instruction cannot be used in interrupt programs or in the initialization module.**
- **Caution is required when using in a subroutine program, between the STP to STE instructions, between the STG to JMP/ENDS instructions or between the CJ (NCJ) to LABEL instructions.**
  Refer to "Cautions for using Timer Instruction" in the User's Manual of the CPU unit used.
- **Caution is required when using write during RUN.**
  Refer to "Writing during RUN" in the User's Manual of the CPU unit used.

Sample Program

Programmed Script

LLFLT (DM1500.S, EM2000, DM2000.S)

Operation Description

At each scan (no execution conditions), the lead-lag operation is executed on the value stored in DM1500 using the DM1001 (lag element time constant) and EM1002 (lead element time constant) at the EM1000 sampling cycle.
The operation results are stored in DM3000.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

TBCD Convert BIN data to BCD data

Return value*1 = TBCD (conversion destination) (Convert BCD)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Conversion destination	Specifies the conversion destination or the conversion destination data.	.U	.U	.D	.D	—	—	—	—	○	○	○
R	Return value	TBCD function type is identical to the arguments.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

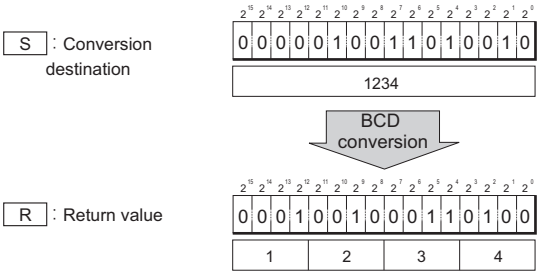
*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.
*3 The ".DF" suffix cannot be specified for device Z.

Point
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

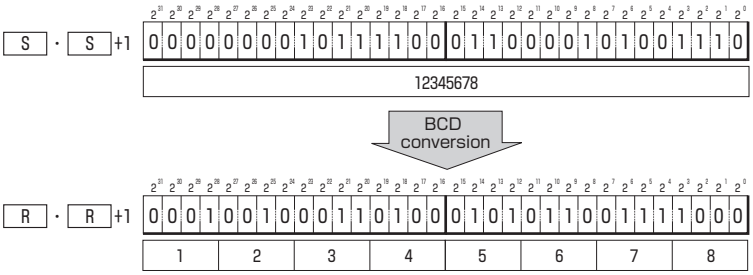
Description of Operation

The 16-bit BIN data stored in S is converted to 4-digit BCD (16-bit) data or 32-bit BIN data is converted to 8-digit BCD (32-bit) data, and is stored in R.

- S : When the conversion destination is 16 bits



- S : When the conversion destination is 32 bits



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● **Example of format**

DM2000.U = TBCD (DM1000.U)

Point **When the range 0 to 9999 is exceeded when [S] is 16 bits, or the range 0 to 99999999 is exceeded when it is 32 bits, the value of [S] is stored in [R].**

Sample Program

● **To specify the bit data**

Programmed Script: DM2000.D = **TBCD** (MR1000.D)

Operation Description: The BIN data stored in MR1000.D is converted to 8-digit BCD (32-bit) data.
The conversion result is stored in DM2000.D.

● **Example of use**

The TBCD function is used, for example, to output the current values as they are via BCD as it can convert BIN data specified by the conversion destination to BCD data.

Programmed Script: R1000.U = **TBCD** (TOU (DM1000 + 1000))

Operation Description: The value of the result obtained by adding 1000 to the content of DM1000 is converted to 4-digit BCD, and is output to output relays R1000 to R1015.

Reference: The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

TBCD
TBIN
MPX
DMX
GRY
RGRY
DISN
UNIN
DISB
UNIB
SWAP
BSWAP
DECO
ENCO
TOU
TOS
TOD
TOL
CPMSET
CPMGET

TBIN

Convert BCD data to BIN data

Return value*1 = TBIN (conversion destination)

Convert BIN

Argument/Return Value		Description	Supported Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Conversion destination	Specifies the conversion destination or the conversion destination data.	.U	.U	.D	.D	—	—	—	—	○	○	○
R	Return value	TBIN function type is identical to the 1st argument.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

Point

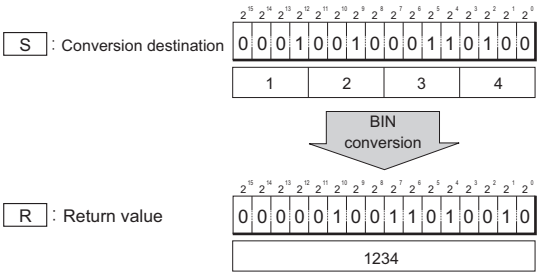
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

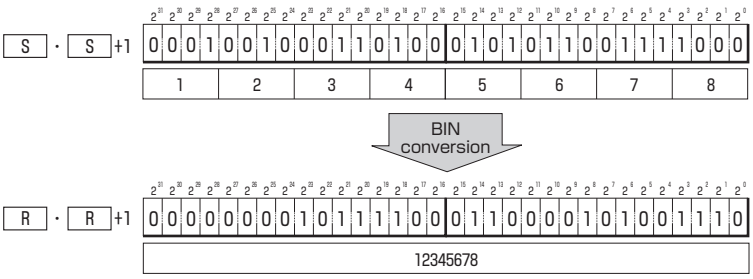
Description of Operation

The 4-digit (16-bit) data stored in S is converted to 16-bit BIN data or 8-digit BCD (32-bit) data is converted to 32-bit BIN data, and is stored in R.

- S : When the conversion destination is 16 bits



- S : When the conversion destination is 32 bits

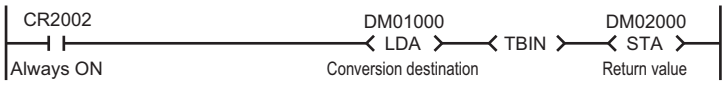


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● Example of format

DM2000.U = TBIN (DM1000.U)



Point **When [S] is not BCD data, the value of [S] is stored in [R].**

Sample Program

● To specify the bit data

Programmed Script: DM2000.D = **TBIN** (MR1000.D)

Operation Description: The 8-digit BCD (32 -bit) data stored in MR1000.D is converted to BIN data. The conversion result is stored in DM2000.D.

● Example of use

The TBIN function is used, for example, to input the current BCD value as it is as BIN data as it can convert BCD data specified by the conversion destination to BIN data.

Programmed Script: DM2000.U = TOU (**TBIN** (R2000.U) + 1000)

Operation Description: The states (values) of relays R2000 to R2015 to which 4-digit BCD data is currently being output are input, converted to BIN data, 1000 is added to the result, and is stored in DM2000.

Reference: The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

TBCD
TBIN
MPX
DMX
GRY
RGRY
DISN
UNIN
DISB
UNIB
SWAP
BSWAP
DECO
ENCO
TOU
TOS
TOD
TOL
CPMSET
CPMGET

MPX 4-bit data decode

Return value*1 = MPX (conversion destination, conversion digit position)

Decode

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Conversion destination*3*4	Specifies the conversion destination or the conversion destination data.	.U	.U	.D	.D	—	—	—	—	—	○	○
n	Conversion digit position*2	Specifies the conversion digit position. (0 to 3)	—	—	—	—	—	—	—	—	○	—	—
R	Return value	.U type MPX function available.* 5	.U	.S	.D	.L	.F	*6 .DF	—	—	—	○	—

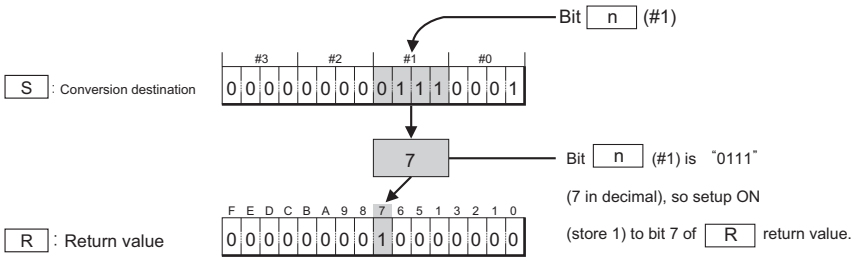
*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 The number of digits 0 to 3 (4 bits/1 digit) is specified to the conversion digit position by a constant. (No target can be specified.)
*3 When 32-bit data is specified for the conversion destination, the low-order 16 bits (low-order word) will be converted (effective conversion bits).
*4 Even when a relay other than the leading relay (e.g. R1002 or R1005) of the channel for the bit data has been specified for the conversion destination, the function operates normally straddling the next channel.
*5 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.
*6 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

The 4 bits (bit 0 to 3) specified by n of the 16-bit data stored in S are converted to numbers (0 to 15). The result of turning ON only the bit specified by the values after conversion to numbers is stored in R.



For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● Example of format

DM2000 = **MPX** (DM1000, 2)

Sample Program

● To decode a bit data to 16-bit BIN data

Programmed Script

DM2000 = **MPX** (MR1000.U, 3)

Operation Description

The value of the 3rd digit (four bits, bit 12 to bit 15) of the data stored in MR1000.U is decoded, and the result is stored in DM2000.

Ladder conversion

● When this function is used in combination with other functions

Programmed Script

R2000.U = **MPX** (TBIN (DM1000), 1)

Operation Description

The 16-bit BIN data currently stored in DM1000 is converted to 4-digit BCD (16-bit) data, and the value of the 10's digit (1st digit) of the conversion result is decoded to turn ON the corresponding output relays (R2000 to R2015).

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "TBIN function", page 7-82
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

DMX 4-bit data encode

Return value*1 = DMX (conversion destination) (Decode)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Conversion destination*2*3	Specifies the conversion destination or the operation destination data.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	.U type DMX function available.*4	.U	.S	.D	.L	.F	*5 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When 32-bit data is specified for the conversion destination, the low-order 16 bits (low-order word) will be converted (effective bits).

*3 Even when a relay other than the leading relay (e.g. R1002 or R1005) of the channel for the bit data has been specified for the conversion destination, the function operates normally straddling the next channel.

*4 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.

*5 The ".DF" suffix cannot be specified for device Z.

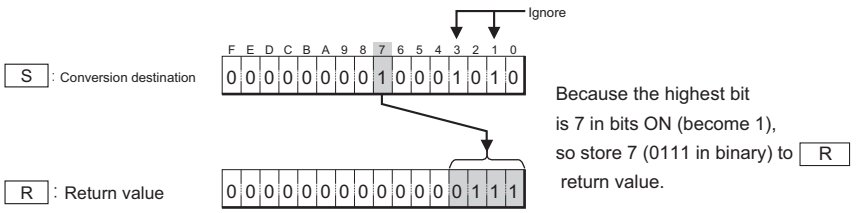
Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

The MSB positions (0 to 15) in the ON bits of the 16-bit data stored in S is stored in R as 16-bit (stored in low-order four bits) data.

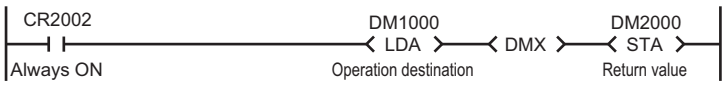


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● Example of format

DM2000 = DMX (DM1000)



Point

When the value of S is "0", "0" is stored in R.

Sample Program

● To encode a bit data to 4-bit numerical value data

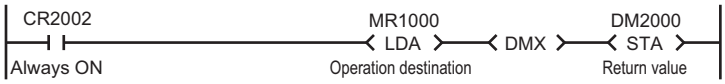
Programmed Script

DM2000 = **DMX** (MR1000.U)

Operation Description

The MSB position in the ON bits of the data stored in MR1000.D is stored in DM2000.

Ladder conversion



● When this function is used in combination with other functions

Programmed Script

R2000.U = TBCD (**DMX** (MR1000.U))

Operation Description

The MSB position (sensor No.) in the ON bits is converted to 4-digit BCD (16-bit) data by the sensor input relay (MR1000 to MR1015), and the result is output to the BCD output relays (R2000 to R2015).

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "TBCD function", page 7-80
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

GRY Convert BIN data to gray code

Return value*1 = GRY (conversion destination)
Gray code conversion

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Conversion destination	Specifies the conversion destination or the operation destination data.	.U	.U	.D	.D	—	—	—	—	○	○	○
R	Return value	GRY function type is identical to the argument.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.
*3 The ".DF" suffix cannot be specified for device Z.

Point
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

The 16-bit or 32-bit BIN data stored in S is converted to gray code, and is stored in R.
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● Example of format

DM2000.U = GRY (DM1000.U)

Gray code table

Decimal	bit														
	16	15	14	13	12	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
4	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
to															
31	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
32	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
to															
255	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
256	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
to															
65535	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Sample Program

● To specify the bit data

Programmed Script

DM2000.D = **GRY** (MR1000.D)

Operation Description

The BIN data stored in MR1000.D is converted to gray code.
The conversion result is stored in DM2000.D.

● Example of use

The GRY function is used, for example, to output the current value as it is as gray code as it can convert BIN data specified by the conversion destination to gray code.

Programmed Script

R1000.U = **GRY** (TOU (DM1000 + 128))

Operation Description

The value of the result obtained by adding 128 to the content of DM1000 is converted to gray code, and is output to output relays R1000 to R1015.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

RGRY Convert gray code to BIN data

Gray code reverse conversion

Return value*1 = RGRY (conversion destination)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Conversion destination*2	Specifies the conversion destination or the operation destination data.	.U	.U	.D	.D	—	—	—	—	○	○	○
R	Return value	RGRY function type is identical to the argument.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or later, and KV Nano Series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

The gray code stored in S is converted to 16-bit or 32-bit BIN data, and is stored in R.
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● Example of format

DM2000.U = **RGRY** (DM1000.U)

Gray code table

Decimal	bit														
	16	15	14	13	12	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
4	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
to															
31	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
32	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
to															
255	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
256	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
to															
65535	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Sample Program

● To specify the bit data

Programmed Script

DM2000.D = **RGRY** (MR1000.D)

Operation Description

The gray code stored in MR1000.D is converted to BIN data.
The conversion result is stored in DM2000.D.

● Example of use

The RGRY function is used, for example to input the current value of the gray code as it is as BIN data as it can convert gray code specified by the conversion destination to BIN data.

Programmed Script

DM2000.U = TOU (**RGRY** (R2000.U) – 128)

Operation Description

The states (values) of relays R2000 to R2015 to which gray code is currently being output are input, converted to BIN data, 128 is subtracted from the result, and is stored in DM2000.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

DISN Disperse nibble

Disperse N

DISN ([execution condition,]*1 Operation destination, disperse data storage destination, number of data to be dispersed)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Operation destination*2*4	Specifies the storage location for the disperse nibble data.	.U	.U	.D	.D	—	—	.B	—	—	○	—
D	Disperse data storage destination*2*4	Specifies the start to store the dispersed nibble data.	Handled as same type as S.					—	—	—	—	○	—
n	Number of disperse data*3	Specifies the number of half-byte data to be dispersed.	.U	.U	.U	.U	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When bit data is specified, S processed 16/32 continuous bits. D processes 64/128 continuous bits.
When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled straddling the next channel.
When word data is specified, S processes 1/2 continuous words. D processes 4/8 continuous words.

*3 0 to 4 when the data suffix specified by S is .U or .B, and 0 to 8 when it is .D.

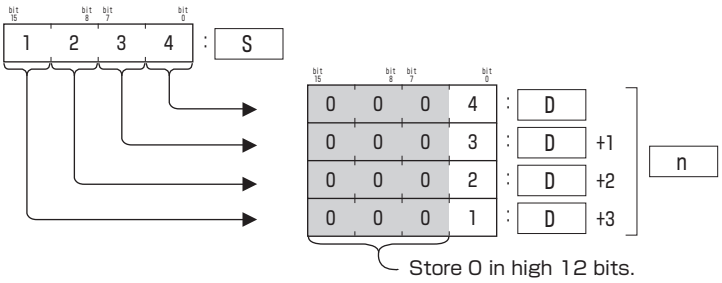
*4 CTC, CTH, Z cannot be specified.

Point **When using KV-1000, please use DISN instruction in ladder program.**

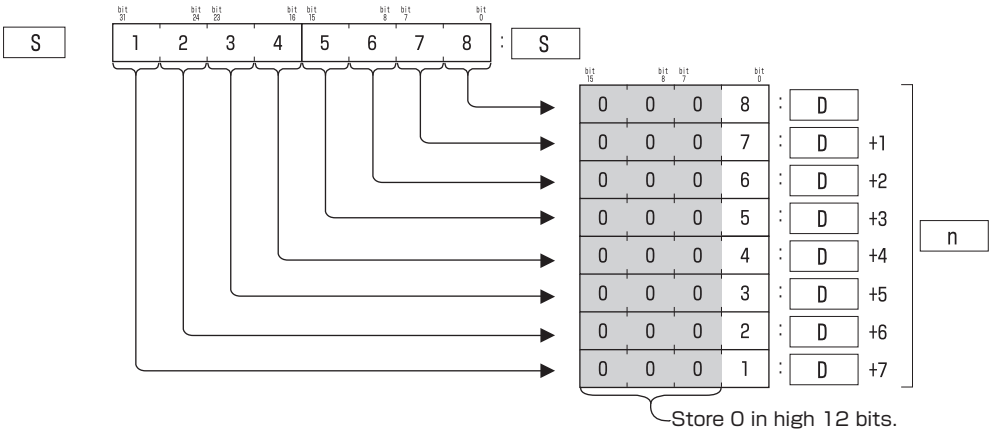
Description of Operation

When the execution condition is established (TRUE), data specified by S is dispersed into n nibbles (4-bit), and the result is stored in D.
Store 0 in high-order 4 bits.

- S : When the operation destination is unsigned 16-bit (.U)



- S : When the operation destination is unsigned 32-bit data (.D)

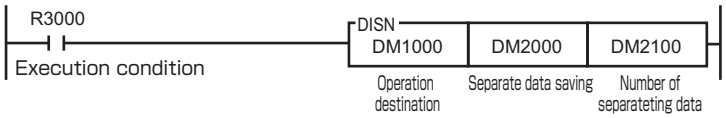


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● Example of format

DISN (R3000, DM1000.U, DM2000, DM2100.U)



Point **When n is 0, the function is invalid.**

TBCD
TBIN
MPX
DMX
GRY
RGRY
DISN
UNIN
DISB
UNIB
SWAP
BSWAP
DECO
ENCO
TOU
TOS
TOD
TOL
CPMSET
CPMGET

UNIN Unite nibble

Unite N

UNIN ([execution condition,]*1 Operation destination, unite data storage destination, number of data to be united)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Operation destination*2*4	Specifies the storage location for the unite nibble data.	.U	.U	.D	.D	—	—	.B	—	—	○	—
D	Unite data storage destination*2*4	Specifies the start to store the unite nibble data.	Handled as same type as S.					—	—	—	—	○	—
n	Number of unite data*3	Specifies the number of half-byte data to be united.	.U	.U	.U	.U	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When bit data is specified, S processes 64/128 continuous bits. D processes 16/32 continuous bits. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled straddling the next channel.
When word data is specified, S processes 4/8 continuous words. D processes 1/2 continuous words.

*3 0 to 4 when the data suffix specified by S is .U or .B, and 0 to 8 when it is .D.

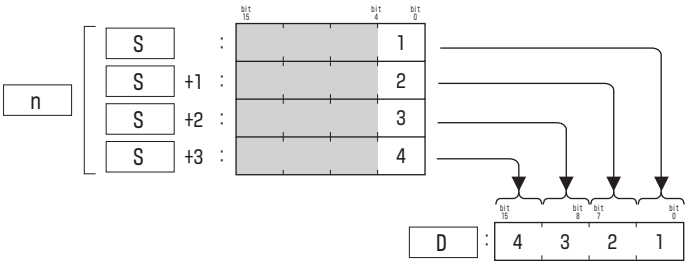
*4 CTC, CTH, Z cannot be specified.

Point **When using KV-1000, please use UNIN instruction in ladder program.**

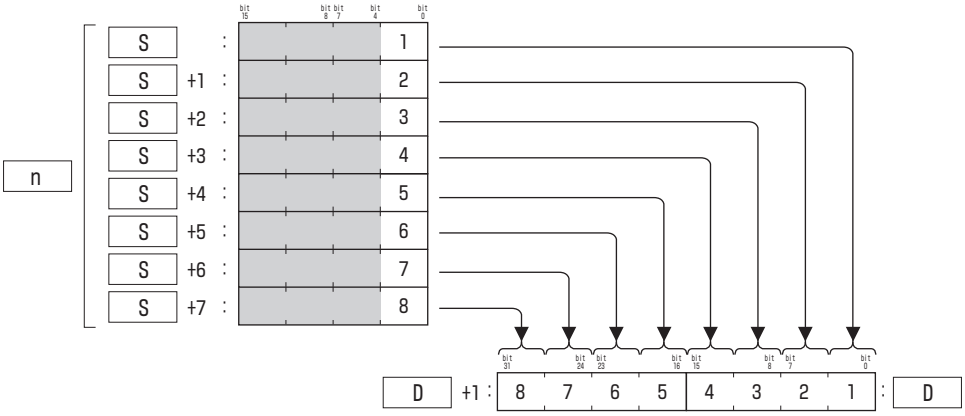
Description of Operation

When the execution condition is established (TRUE), low-order 4 bits of the unite data starting from S are united to n BIN data, and the result is stored in D.

- S : When the operation destination is unsigned 16-bit (.U)



- S : When the operation destination is unsigned 32-bit data (.D)

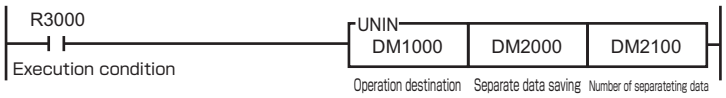


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● Example of format

UNIN (R3000, DM1000.U, DM2000, DM2100.U)



Point **When n is 0, the function is invalid.**

TBCD
TBIN
MPX
DMX
GRY
RGRY
DISN
UNIN
DISB
UNIB
SWAP
BSWAP
DECO
ENCO
TOU
TOS
TOD
TOL
CPMSET
CPMGET

DISB Disperse byte

Device B

DISB ([execution condition,]*1 Operation destination, disperse data storage destination, number of data to be dispersed)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Operation destination*2*4	Specifies the storage location for the disperse byte data.	.U	.U	.D	.D	—	—	.B	—	—	○	—
D	Disperse data storage destination*2*4	Specifies the start to store the disperse byte data.	Handled as same type as S.					—	—	—	—	○	—
n	Number of disperse data*3	Specifies the number of byte data to be dispersed.	.U	.U	.U	.U	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When bit data is specified for S, D processes 16/32 continuous bits. 32/64 continuous bits are processed.
When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled straddling the next channel.
When word data is specified, S processes 1/2 continuous words. D processes 2/4 continuous words.

*3 0 to 2 when the data suffix specified by S is .U or .B, and 0 to 4 when it is .D.

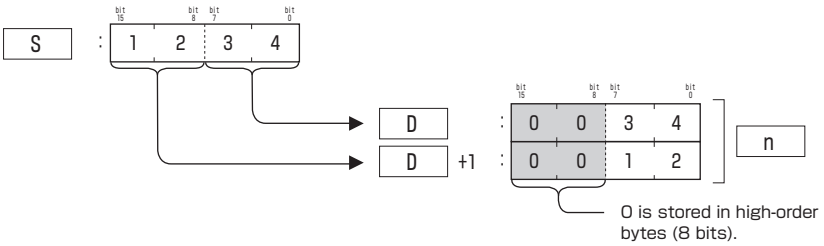
*4 CTC, CTH, Z cannot be specified.

Point **When using KV-1000, please use DISB instruction in ladder.**

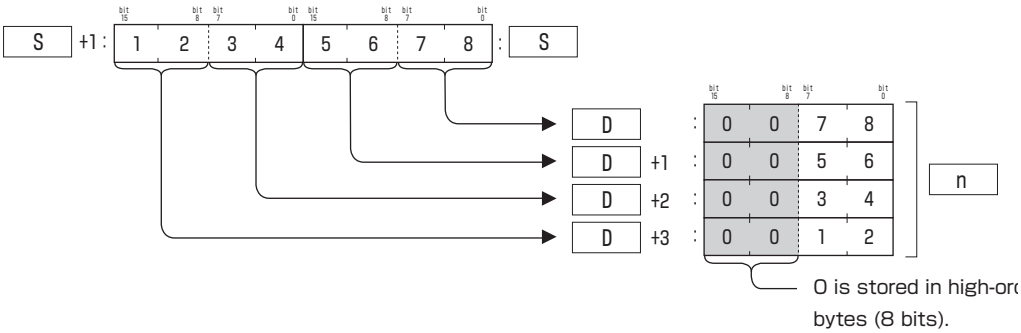
Description of Operation

When the execution condition is established (TRUE), data specified by S is dispersed into n bytes (8-bit), and the result is stored in D.
Store 0 in high-order 8 bits.

- S : When the operation destination is unsigned 16-bit (.U)



- S : When the operation destination is unsigned 32-bit data (.D)



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● Example of format

DISB (R3000, DM1000.U, DM2000, DM2100.U)

Point **When n is 0, the function is invalid.**

TBCD
TBIN
MPX
DMX
GRY
RGRY
DISN
UNIN
DISB
UNIB
SWAP
BSWAP
DECO
ENCO
TOU
TOS
TOD
TOL
CPMSET
CPMGET

UNIB Unite byte

Unite B

UNIB ([execution condition,]*1 Operation destination, unite data storage destination, number of data to be united)

Argument/Return Value		Description	Supported Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Operation destination*2*4	Specifies the storage location for the unite byte data.	.U	.U	.D	.D	—	—	.B	—	—	○	—
D	Unite data storage destination*2*4	Specifies the start to store the unite byte data.	Handled as same type as S.					—	—	—	—	○	—
n	Number of unite data*3	Specifies the number of byte data to be united.	.U	.U	.U	.U	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When bit data is specified, S processes 32/64 continuous bits. D processes 16/32 continuous bits.
When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled straddling the next channel.
When word data is specified, S processes 2/4 continuous words. D processes 1/2 continuous words.

*3 0 to 2 when the data suffix specified by S is .U or .B, and 0 to 4 when it is .D.

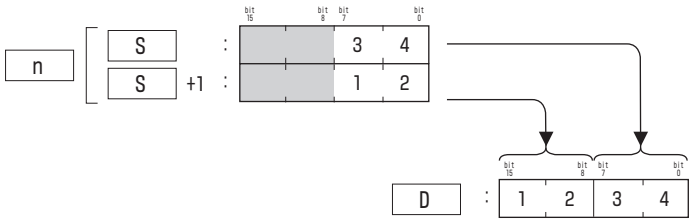
*4 CTC, CTH, Z cannot be specified.

Point **When using KV-1000, please use UNIB instruction in ladder**

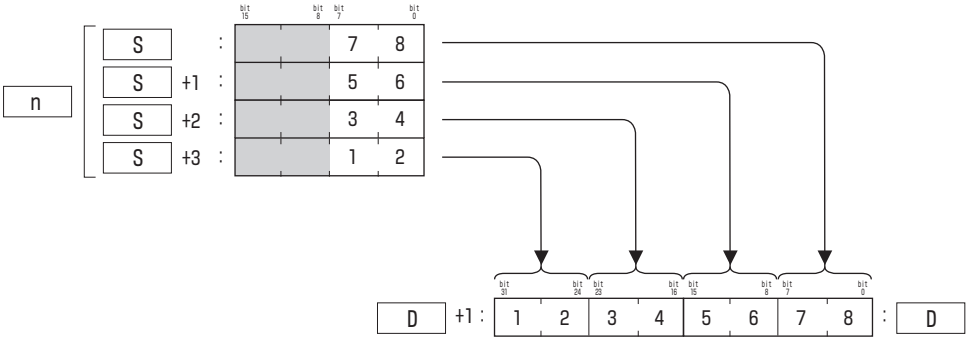
Description of Operation

When the execution condition is established (TRUE), low-order 8 bits of the unite data starting from S are united to n BIN data, and the result is stored in D.

- S : When the operation destination is unsigned 16-bit (.U)



- S : When the operation destination is unsigned 32-bit data (.D)

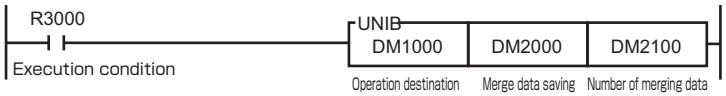


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● Example of format

UNIB (R3000, DM1000.U, DM2000, DM2100.U)



Point **When n is 0, the function is invalid.**

TBCD
TBIN
MPX
DMX
GRY
RGRY
DISN
UNIN
DISB
UNIB
SWAP
BSWAP
DECO
ENCO
TOU
TOS
TOD
TOL
CPMSET
CPMGET

SWAP High lower byte conversion

Return value*1 = SWAP (swap destination)

Convert BCD

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Swap destination	Specifies the swap destination or the swap destination data.	.U	.U	.D	.D	—	—	—	—	○	○	○
R	Return value	SWAP function type is identical to the argument.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the exchange result is stored in internal registers.

*2 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

Point

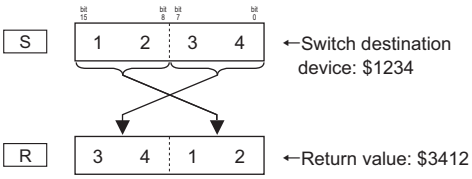
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

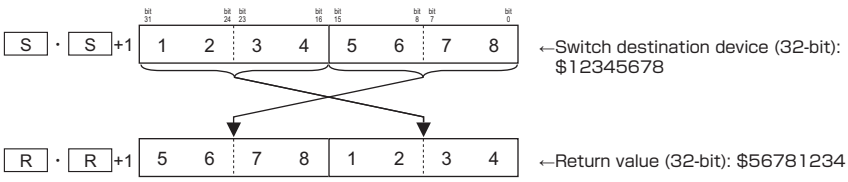
Description of Operation

The values of high-order byte (high-order 8 bits) and low-order byte (low-order 8 bits) of 16-bit data stored in S are swapped, and the result is stored to R.

- S : When the swap destination is 16 bits



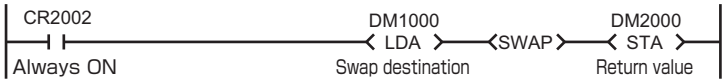
- S : When the swap destination is 32 bits



For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● Example of format

DM2000.U = **SWAP** (DM1000.U)



Sample Program

● To specify the bit data

Programmed Script

DM2000.D = **SWAP** (MR1000.D)

Operation Description

Swap the values of high-order word (MR1000 to MR1015) and low-order word (MR1100 to MR1115) of 32-bit data stored in MR1000.D.
The swap result is stored in DM2000.D.

● When this function is used in combination with other functions

Programmed Script

R2000.U = **SWAP** (TBCD (TOU (DM1000 + 100)))

Operation Description

Add 100 to the content of DM1000, convert the result value to 4-bit BCD, then swap high-order 2 bits and low-order 2 bits of the BCD data and output to output relay R2000 to R2015.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "TBCD function", page 7-80
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

BSWAP Block swap

Block swap

BSWAP ([execution condition,]*1 swap destination, number of destination data items)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
D	Swap destination	Specifies the swap destination.	.U	.U	.U	.U	—	—	.B	—	—	○	—
n	Number of target data items	Specifies the number to be swapped.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

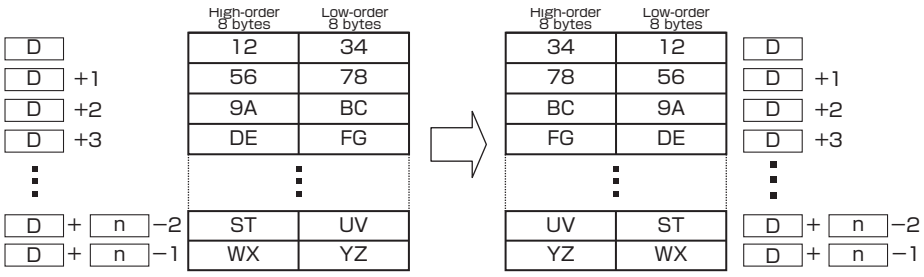
*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

Point

- **The BSWAP function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

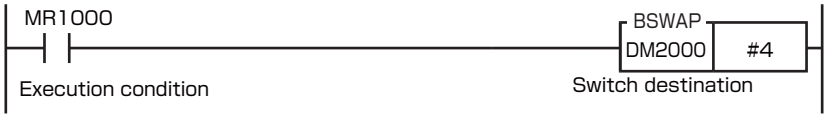
When the execution condition is established (TRUE), the high-order bytes (high-order 8 bytes) and low-order bytes (low-order 8 bytes) of the n data items starting with D are swapped.



For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● Example of format

BSWAP (MR1000, DM2000, 4)



Point

When n =0, the function is not executed.

Sample Program

Programmed Script

BSWAP (DM3000, 8)

Operation Description

The high-order bytes and low-order bytes of DM3000 to DM3007 are swapped.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

TBCD
TBIN
MPX
DMX
GRY
RGRY
DISN
UNIN
DISB
UNIB
SWAP
BSWAP
DECO
ENCO
TOU
TOS
TOD
TOL
CPMSET
CPMGET

DECO 8-bit decode

8-bit decode

DECO ([execution condition,]*1 Conversion destination, storage destination, conversion destination range)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Conversion destination	Specifies the start of the conversion range.	.U	.U	.U	.U	—	—	—	—	—	○	—
D	Storage location*2 *3	Specifies the storage location for the decode result.	—	—	—	—		—	.B	—	—	○	—
n	Conversion destination range	Specifies the number of ranges (number of bits within range 1 to 8) to convert.	*4 .U	*4 .U	*4 .U	*4 .U	—	—	—	—	○	*4 ○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When word data has been specified, the LSB of continuous 2^n word data becomes the decode result storage destination data. (Bits other than the LSB do not change.)

*3 CTC, CTH, Z cannot be used.

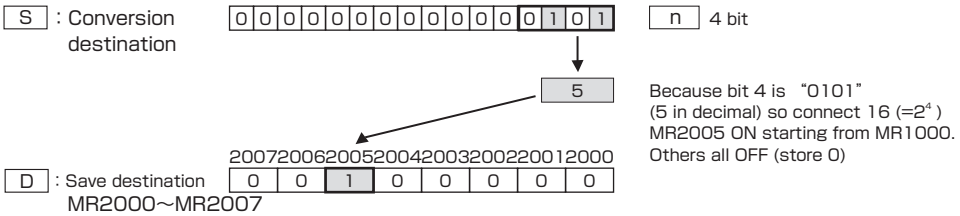
*4 A target can be specified for the number of conversion target ranges only when the following conditions are met.

- When the KV STUDIO version is Ver. 6.00 or higher
- Only when the compatible model is a KV-8000/7000 Series CPU unit, a KV-5500/5000/3000 Series CPU unit that has CPU function version 2.0 or higher, or a KV Nano Series base unit.

Description of Operation

When the execution condition is established (TRUE), the states (number of bits) specified by n starting from the bit data specified by S is converted to a numerical value (0 to 255). The bit of the start No. + numerical value (No.) in the continuou 2^{n} bit data starting from D is turned ON, and other bits are all turned OFF.

When S is a word data, n indicates the number of bits to convert to numerical values, and when the storage destination is a word data, the LSB of the word data of the start No. + numerical value (No.) in the continuous 2^{n} word data starting from the specified word data is turend ON, and the LSBs of other word data are all turned OFF. (Bits other than the LSB do not change.)

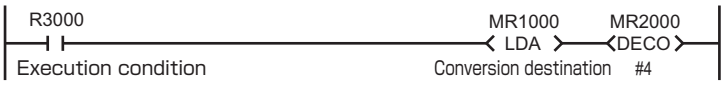


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● **Example of format**

DECO (R3000, MR1000.U, MR2000, 4)



Point **When [n]=0, the function is not executed.**

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

DECO (ORA (DM3000, DM3001), MR3000, 5)

Operation Description

The lower five bits of the logical "OR" operation result for the data currently stored in DM3000 and DM3001 are decoded at every scan (no execution condition).
The bit of "start No. + converted numerical value" (nth value) in the continuous 2^5 bits starting from MR3000 is turned ON, and all other bits are turned OFF.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "ORA function", page 7-24
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

TBCD
TBIN
MPX
DMX
GRY
RGRY
DISN
UNIN
DISB
UNIB
SWAP
BSWAP
DECO
ENCO
TOU
TOS
TOD
TOL
CPMSET
CPMGET

ENCO 8-bit encode

Return value*1 = ENCO (conversion destination, number of conversion destination ranges)

8-bit encode

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Conversion destination*2*3	Specifies the start of the conversion range.	—	—	—	—	—	—	.B	—	—	○	—
n	Number of conversion destination ranges	Specifies the number of conversion ranges (number of bits). (1 to 256)	*5 .U	*5 .U	*5 .U	*5 .U	—	—	—	—	○	*5 ○	○
R	Return value	.U type ENCO function available.* 4	.U	.S	.D	.L	.F	*6 .DF	—	—	—	○	—

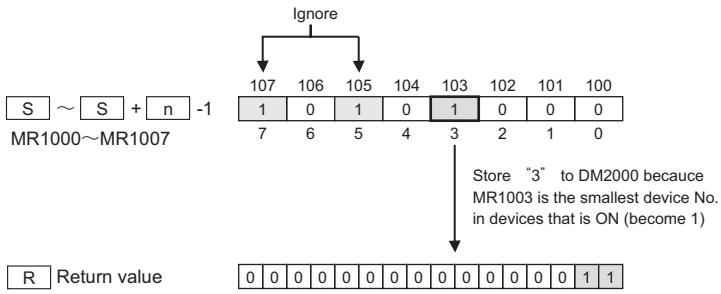
*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 When word data has been specified, the LSB of continuous n word data becomes the decode destination.
*3 CTC, CTH, Z cannot be used.
*4 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.
*5 A target can be specified for the number of conversion target ranges only when the following conditions are met.
- When the KV STUDIO version is Ver. 6.00 or higher
- Only when the compatible model is a KV-8000/7000 Series CPU unit, a KV-5500/5000/3000 Series CPU unit that has CPU function version 2.0 or higher, or a KV Nano Series base unit.

*6 The ".DF" suffix cannot be specified for device Z.

Point
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

The smallest device position (0 to 255) of the ON bits in the states (number of bits) specified by n starting with the bit data specified by S is stored in R as 16-bit data.
When S is word data, n indicates the number of words, and the smallest bit position (0 to 255) of the word with the LSB ON in the word data of conversion destination range number is stored in R as 16-bit data.

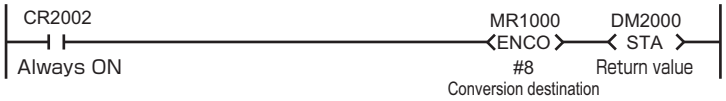


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● **Example of format**

DM2000.U = **ENCO** (MR1000, 8)



Point **When [n]=0, the function is not executed.**

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

DM2000.U = TBCD (**ENCO** (MR3000, 32))

Operation Description

The position (16-bit data) with the LSB ON in 32 bit data (MR3000 to MR3115) starting from MR3000 is converted to 4-digit BCD data and stored in DM2000.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "TBCD function", page 7-80
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

TBCD
TBIN
MPX
DMX
GRY
RGRY
DISN
UNIN
DISB
UNIB
SWAP
BSWAP
DECO
ENCO
TOU
TOS
TOD
TOL
CPMSET
CPMGET

TOU Convert data type to .U
TOS Convert data type to .S
TOD Convert data type to .D
TOL Convert data type to .L

Return value*1 = TOU (conversion destination) (U conversion)

Return value*1 = TOS (conversion destination) (S conversion)

Return value*1 = TOD (conversion destination) (D conversion)

Return value*1 = TOL (conversion destination) (L conversion)

Argument/Return Value		Description	Supported Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Conversion destination	Specifies the storage location for the conversion destination data or the conversion data.	.U	.S	.D	.L	.F	*2 .DF	—	—	○	○	○
R	Return value*1	TOU function: .U type. TOS function: .S type. TOD function: .D type. TOL fucntion: .L type.	.U	.S	.D	.L	.F	*2 .DF	—	—	—	○	—

*1 If a type different from the function type specified in the return value, it will be automatically converted during assignment.

*2 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

- TOU
  The data stored in S is converted to unsigned 16-bit BIN data, and the result is stored in R.

Reference

- When 32-bit BIN data has been specified, the lower 16 bits are targeted for conversion.
- When single/double precision floating point type real number has been specified as the conversion destination data, the lower 16 bits of the result after having converted to unsigned 32-bit BIN data is targeted for conversion.
- The suffix of the storage location for the return value should be set to .U.

- TOS
  The data stored in [S] is converted to signed 16-bit BIN data, and the result is stored in [R].

Reference
- When 32-bit BIN data has been specified, the lower 16 bits are targeted for conversion.
- When single/double precision floating point type real number has been specified as the conversion destination data, the lower 16 bits of the result after having converted to signed 32-bit BIN data is targeted for conversion.
- The suffix of the storage location for the return value should be set to .S.

- TOD
  The data stored in [S] is converted to unsigned 32-bit BIN data, and the result is stored in [R].

Reference
- When 16-bit BIN data has been specified to the conversion destination data, the 32-bit data of the result of having extended the data is targeted for conversion.
- When single/double precision floating point type real number has been specified as the conversion destination data, the result after having converted to unsigned 32-bit BIN data is targeted for conversion.
- The suffix of the storage location for the return value should be set to .D.

- TOL
  The data stored in [S] is converted to signed 32-bit BIN data, and the result is stored in [R].

Reference
- When 16-bit BIN data has been specified to the conversion destination data, the 32-bit data of the result of having extended the data is targeted for conversion.
- When single/double precision floating point type real number has been specified as the conversion destination data, the result after having converted to signed 32-bit BIN data is targeted for conversion.
- The suffix of the storage location for the return value should be set to .L.

● Example of format

DM2000.L = **TOL** (DM1000.S)

Sample Program

The operation formula programmed by the script is operated on as signed 32-bit data (.L).
When converting to a ladder from script, and the suffix of the storage location for the operation result is other than ".L", the type conversion warning will occur.
To avoid the type conversion warning, use each of the "TOU, TOS, TOD, TOL" functions to convert the operation result to the same type as the storage location.

● When the type of the storage location for the operation result is unsigned 32-bit data ".D"

Programmed Script

DM1000.D = **TOD** ((DM2000 + 50) * DM2100 + 10)

Operation Description

The operation result (operated on by .L) of the right side of "=" is converted to unsigned 32-bit data ".D", and substituted (stored) to DM1000.D on the left side.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

CPMSET CIP message creation

CPMSET ([execution condition,]*1, input data storage destination, parameter, output data storage destination)

Argument/Return Value		Description	Supported Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Input data storage destination*2	Specifies the start of the addtional data.	.U	.S	.D	.L	.F	.DF	.B	.T	—	○	—
S2	Parameter	Specifies the data type, data size (in bytes) or storage location of the additional send data.*2	.U	.U	.U	.U	—	—	—	—	○	○	○
D	Output data storage destination*2	Specifies the start of the created send data.	.U	.S	.D	.L	.F	.DF	.B	.T	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When specifying bit data, specify the start of the channel.

Point

- **The CPMSET function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

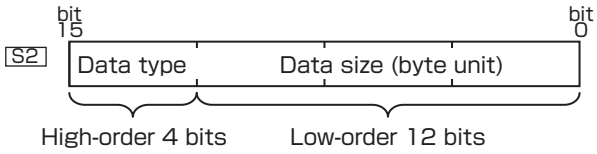
Description of Operation

When the execution condition is established (TRUE), the data stored in S1 is added after the data size (in bytes) stored in D with the data type specified by the high-order 4 bits of S2 and the data size (in bytes) specified by the low-order 12 bits.

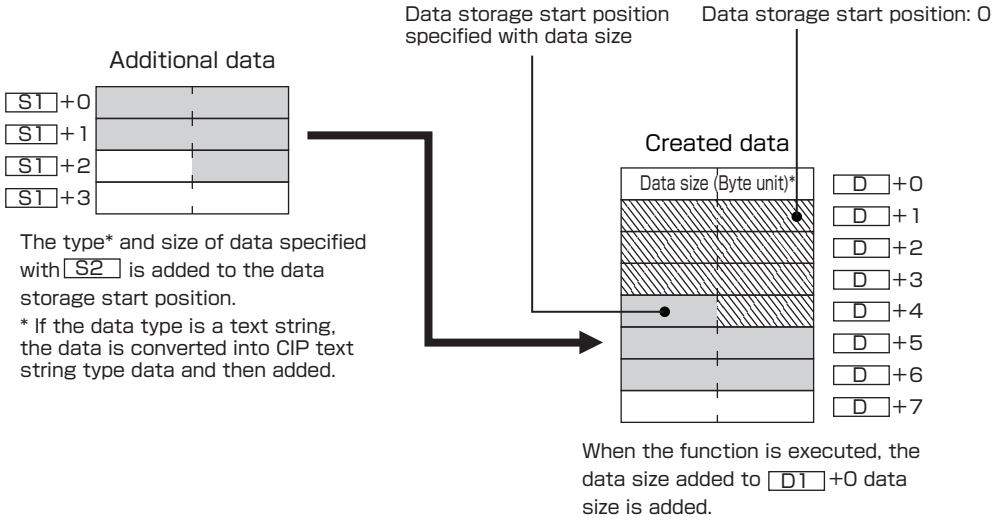
When adding data for the first time, 0 is stored in D before the function is executed.

The CPMSET function is a function for creating CIP data type text string data and numeric data.

Refer to "KV EtherNet/IP Function User's Manual" for details on the CIP data types.



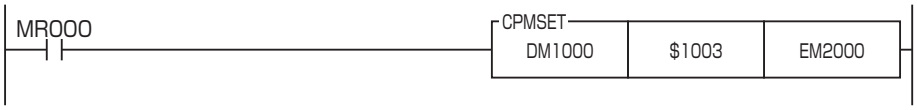
High-order 4 bits	Low-order 12 bits
0: Other than text string	1 to 4095 (001H to FFFH): Specifies data size
1: Text string	0 to 1999 (000H to 7CFH): Specifies number of characters to convert into CIP text string type data. If 0 is specified, characters up to NUL(00H) are converted. If the specified number of characters contains NUL(00H), 00H is added to the data following NULL(00H).
2 or more: Reserved for system	—



For details, please refer to 📖 "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● Example of format

CPMSET (MR0,DM1000,$1003,EM2000)



TBCD
TBIN
MPX
DMX
GRY
RGRY
DISN
UNIN
DISB
UNIB
SWAP
BSWAP
DECO
ENCO
TOU
TOS
TOD
TOL
CPMSET
CPMGET

CPMGET CIP message retrieval

CPMGET ([execution condition,]*1, input data storage destination, parameter, input data offset, output data storage destination)

Argument/Return Value		Description	Supported Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Input data storage destination*2	Specifies the start of the storage location for the receive data.	.U	.S	.D	.L	.F	.DF	.B	.T	—	○	—
S2	Parameter	Specifies the data type, data size (in bytes) or storage location of the receive data to be retrieved.* 2	.U	.U	.U	.U	—	—	—	—	○	○	○
D1	Receive data byte offset*2	Specifies the byte offset from the start to specify the receive data retrieval position or the storage location.	.U	.S	.D	.L	—	—	—	—	—	○	—
D2	Output data storage destination*2	Specifies the start to store the retrieved data.	.U	.S	.D	.L	.F	.DF	.B	.T	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When specifying bit data, specify the start of the channel.

Point

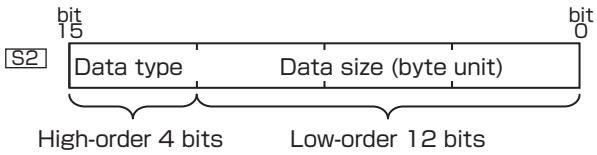
- **The CPMGET function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

When the execution condition is established (TRUE), the data after the offset position specified by D1 of the receive data stored in S1 is stored in D2 with the data type specified by the high-order 4 bits of S2 and the data size (in bytes) specified by the low-order 12 bits of S2. After storing the data, D1 is updated.

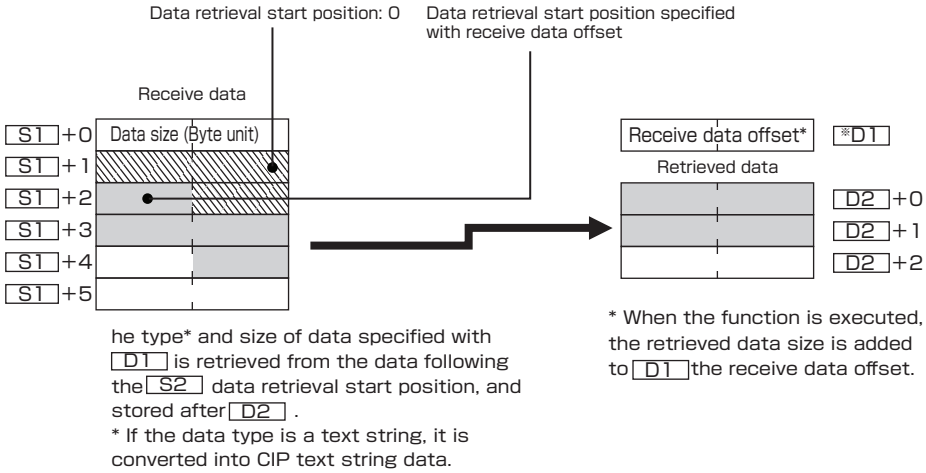
The CPMGET function is used to retrieve the text string data and numeric data of the received CIP data type.

Refer to "KV EtherNet/IP Function User's Manual" for details on the CIP data types.



High-order 4 bits	Low-order 12 bits
0: Other than text string	1 to 4095 (001H to FFFH): Specifies data size

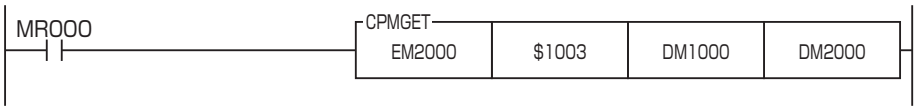
High-order 4 bits	Low-order 12 bits
1: Text string	0 to 1999 (000H to 7CFH): Specifies number of characters to convert into CIP text string type data. If 0 is specified, characters up to NUL(00H) are converted. If the specified number of characters contains NUL(00H), 00H is added to the data following NULL(00H).
2 or more: Reserved for system	—



For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Data Conversion Instructions".

● Example of format

CPMGET (MR0, EM2000, $1003, DM1000, DM2000)



FLOAT

Convert BIN data to single precision floating point type real number data

Single precision floating point conversion

Return value*1 = FLOAT (conversion destination)

Argument/Return Value		Description	Supported Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Conversion destination	Specifies the conversion destination or the conversion destination data.	.U	.U	.D	.D	—	*3 .DF	—	—	○	○	○
R	Return value	.F type FLOAT function available.*2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When the type other than .F is specified by return value, type conversion will be automatically conducted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

Point

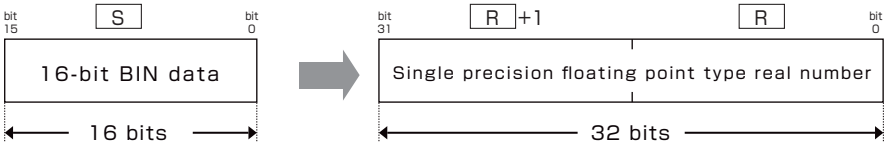
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

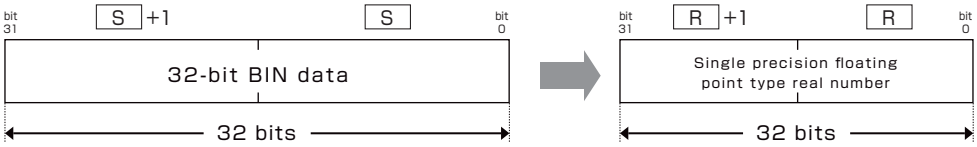
Description of Operation

The binary data (16-bit/32-bit (unsigned/signed), double precision floating point type real number data) stored in S is converted into single precision floating point type real number data, and the result is stored in R.

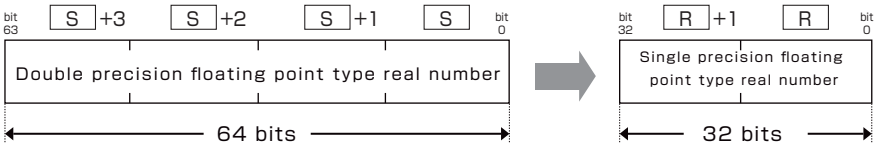
- S : When the type of conversion destination is 16-bit



- S : When the type of conversion destination is 32-bit



- S : When the conversion destination is double precision floating point type real number



For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

● **Example of format**

DM2000.F = **FLOAT** (DM1000.U)

Point **When [S] integer value exceeds -16777216 to 16777215 (24-bit BIN) range, the 25th bit onwards is rounded.**

Sample Program

● **Example of use**

Programmed Script

DM2000.F = **FLOAT** (DM1000.U + 123)

Operation Description

The operation result of DM1000.U+123 is converted to single precision floating point type real number data, and is stored in DM2000.F.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

FLOAT
TOF
DFLOAT
TODF
INTG
INT
DISF
UNIF
EXP
LOG
LOG10
RAD
DEG
SIN
COS
TAN
ASIN
ACOS
ATAN
ATAN2

TOF

Convert BIN data to single precision floating point type real number data

Return value = TOF (conversion destination)

F conversion

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Conversion destination	Specifies the target that stores the conversion destination data or the conversion data.	.U	.U	.D	.D	.F	*2 .DF	—	—	○	○	○
R	Return value	.F type TOF function available.* 1	.U	.S	.D	.L	.F	*2 .DF	—	—	—	○	—

*1 When the type other than .F is specified by return value, type conversion will be automatically conducted during conversion.

*2 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

The data stored in S is converted to single precision floating point type real number data, and the result is stored in R.

"FLOAT function", page 7-114

● Example of format

DM2000.F = **TOF** (DM1000.D)

Reference

- The conversion destination data is converted into 32-bit single precision floating point type real number with the FLOAT instruction corresponding to the data type (16-bit/32-bit (unsigned/signed)).
- If the conversion destination data is a double precision floating point type real number, it is converted into 32-bit single precision floating point type real number by the DFTOF instruction.
- The .F suffix must be specified for the storage location of the return value.

MEMO

FLOAT
TOF
DFLOAT
TODF
INTG
INT
DISF
UNIF
EXP
LOG
LOG10
RAD
DEG
SIN
COS
TAN
ASIN
ACOS
ATAN
ATAN2

DFLOAT

Convert BIN data to double precision floating point type real number data

Double precision floating point conversion

Return value*1 = DFLOAT (conversion destination)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Conversion destination	Specifies the target that stores the conversion destination data or the conversion data.	.U	.U	.D	.D	.F	—	—	—	○	○	○
R	Return value	.DF type DFLOAT function available.*2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in 64-bit internal registers.

*2 When the type other than .DF is specified by return value, type conversion will be automatically conducted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

Point

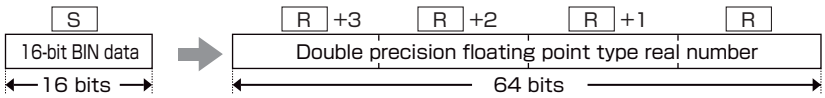
- **The DFLOAT function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be used with KV-8000/7000 Series CPU units, KV-5500/ 5000/ 3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

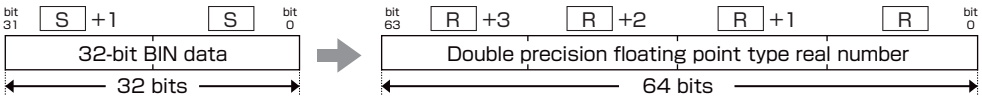
Description of Operation

The binary data (16-bit/32-bit (unsigned/signed), single precision floating point type real number data) stored in S is converted into double precision floating point type real number data, and the result is stored in R.

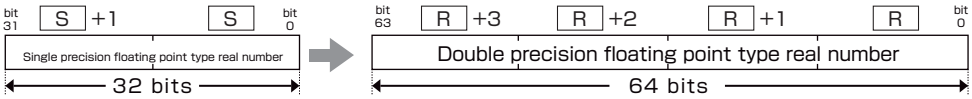
- S : When the type of conversion destination is 16-bit



- S : When the type of conversion destination is 32-bit



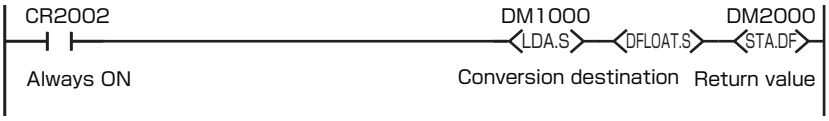
- S : When the type of conversion destination is single precision floating point type real number



For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

● **Example of format**

DM2000.DF = **DFLOAT** (DM1000.S)



Sample Program

Programmed Script

DM2500.DF = **DFLOAT** (DM1500.L + 1000)

Operation Description

The operation result of DM1500.L+1000 is converted to double precision floating point type real number data, and is stored in DM2500.DF.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

TODF

Convert BIN data to double precision floating point type real number data

Return value = TODF (conversion destination)

DF conversion

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Conversion destination	Specifies the target that stores the conversion destination data or the conversion data.	.U	.U	.D	.D	.F	*2 .DF	—	—	○	○	○
R	Return value	.DF type TODF function available.* 1	.U	.S	.D	.L	.F	*2 .DF	—	—	—	○	—

*1 When the type other than .DF is specified by return value, type conversion will be automatically conducted during conversion.

*2 The ".DF" suffix cannot be specified for device Z.

Point

- **The TODF function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be used with KV-8000/7000 Series CPU units, KV-5500/ 5000/ 3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

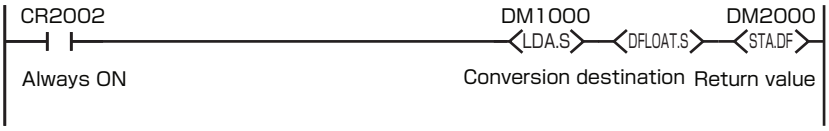
Description of Operation

The data stored in S is converted to double precision floating point type real number data, and the result is stored in R.

"DFLOAT function", page 7-118

● Example of format

DM2000.DF = **TODF** (DM1000.S)



Reference

- The conversion destination data is converted into 64-bit double precision floating point type real number with the DFLOAT instruction corresponding to the data type (16-bit/ 32-bit (unsigned/signed), single precision floating point type real number).
- The .DF suffix must be specified for the storage location of the return value.

MEMO

INTG

Convert single/double precision floating point type real number data to BIN data

Return value*1 = INTG (conversion destination)

Integer conversion

Argument/Return Value		Description	Supported Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Conversion destination	Specifies the conversion destination or the conversion destination data.	—	—	—	—	.F	*3 .DF	—	—	○	○	○
R	Return value	.L type INTG function available.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 C, T, CTC, CTH, Z cannot be used.

*3 When the type other than .L is specified by return value, type conversion will be automatically conducted during conversion.

*4 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **The range of the single precision floating point type real number is shown below.**

  −3.4E+38 ≦ N ≦ −1.4E−45
  N = 0
  1.4E−45 ≦ N ≦ 3.4E+38
  (Number of effective digits: approx. 7 digits)

  (For the KV Nano Series, an absolute value in a range of 1.4E-45 to 1.2E-38 cannot be used.)
- **The range of the double precision floating point type real number is shown below.**

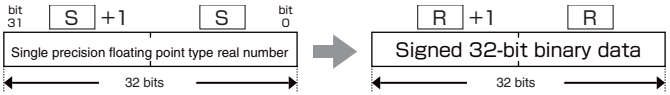
  −1.79E+308 ≦ N ≦ −2.23E−308
  N = 0
  2.23E−308 ≦ N ≦ 1.79E+308

  (Number of effective digits: approx. 16 digits)

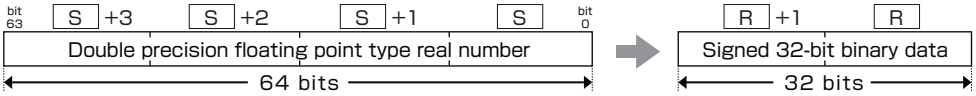
Description of Operation

The single/double precision floating point type real number data stored in S is converted to signed 32-bit BIN data, and the result is stored in R.

- S : When the type of conversion destination is single precision floating point type real number



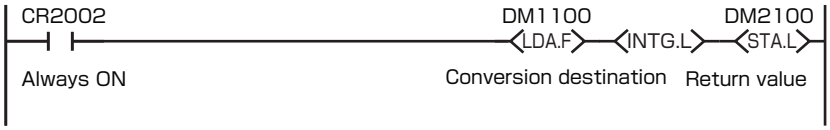
- S : When the type of conversion destination is double precision floating point type real number



For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

● **Example of format**

DM2100.L = **INTG** (DM1100.F)



Point

- **When [S] is outside the single precision floating point type real number range, or when the converted data is outside the signed 16/32-bit BIN range, the bit pattern of [S] is stored in [R].**
- **When [S] is outside the double precision floating point type real number range, or when the converted data is outside the signed 16/32-bit BIN range, the lower 32-bit pattern of [S] is stored in [R].**

Sample Program

● **Example of use**

Programmed Script

DM2000.L = **INTG** (DM1000.F + 1.25)

Operation Description

The operation result of the single precision floating point type real number of DM1000.F+1.25 is converted to a signed 32-bit data integer, and the conversion result is stored in DM2000.L.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

INT

Convert single precision floating point type real number data to BIN data

Return value*1 = INT (conversion destination)

Integer conversion

Argument/Return Value		Description	Supported Type								Const	Device/ Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Conversion destination*2	Specifies the conversion destination or the conversion destination data.	—	—	—	—	.F	*4 .DF	—	—	○	○	○
R	Return value	.L type INT function available.*3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 C, T, CTC, CTH, Z cannot be used.

*3 When the type other than .L is specified by return value, type conversion will be automatically conducted during conversion.

*4 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **Refer to "INTG function", page 7-122 for details on the range of the single precision floating point type real number and double precision floating point type real number.**

Description of Operation

This function is similar to "INTG function" except function name is different.

MEMO

DISF

Disperse mantissa/exponent of single precision floating point type real number

Disperse F

DISF ([execution condition,]*1 Operation destination, mantissa storage destination, exponent storage destination)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Operation destination*2*4	Specifies the destination of the single precision floating point type real number whose mantissa and exponent are to be dispersed.	—	—	—	—	.F	—	—	—	*3 ○	○	○
D1	Mantissa storage destination*2	Specifies the destination to store the dispersed mantissa.	—	.L	.L	—	—	—	—	—		○	
D2	Exponent storage destination	Specifies the destination to store the dispersed exponent.	.S	.S	—	—	—	—	—	—		○	
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When bit data is specified, 32 continuous bits are processed. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled straddling the next channel.

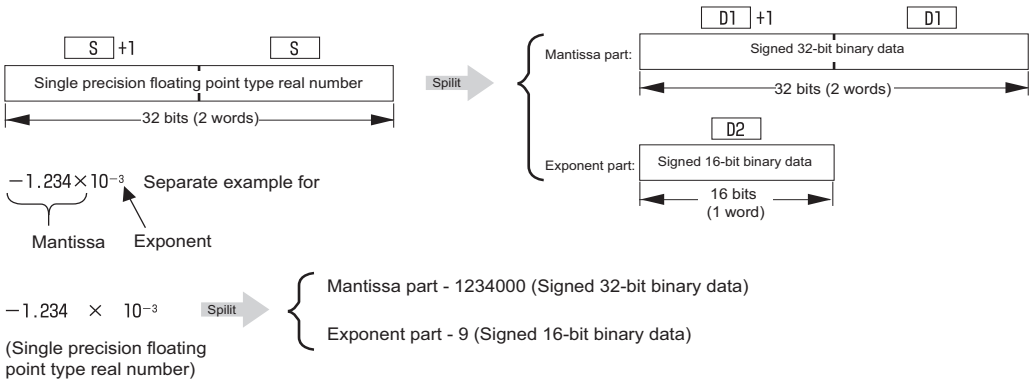
*3 The constant setting is the range of single precision floating point type real numbers.

*4 C, T, CTC, CTH, Z cannot be used.

Point **Refer to "INTG function", page 7-122 for details on the range of the single precision floating point type real number.**

Description of Operation

When the execution condition is established (TRUE), the mantissa section and exponent section of the 32-bit single precision floating point type real number specified by S are dispersed, the mantissa section is stored as signed 32-bit BIN data to the specified D1, and the exponent section is stored as signed 16-bit BIN data to the specified D2.

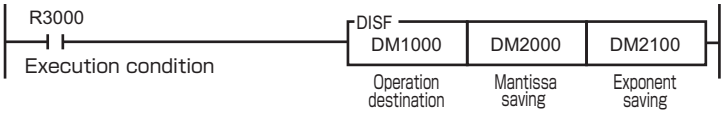


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

● **Example of format**

DISF (R3000, DM1000.F, DM2000.L, DM2100.S)



Point

- **With the DISF function, an error sometimes occurs as single precision floating point type real number (BIN) are being converted to a mantissa section (decimal) and an exponent section (decimal).**
- **The function is not executed when the data to disperse is outside the single precision floating point type real number range.**

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

DISF (LOG (DM3000.F), EM3000.L, EM3100)

Operation Description

The mantissa section and exponent section of the value resulting from having converted the natural logarithm of the single precision floating point type real number currently stored in DM3000.F is dispersed, the mantissa section is stored in EM3000.L, and the exponent section is stored in EM3100.S at every scan (no execution condition).

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "LOG function", page 7-132
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

UNIF

Unite mantissa and exponent to single precision floating point type real number

Return value*1 = UNIF (mantissa storage destination, exponent storage destination)

Unite F

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S1	Mantissa storage destination*2	Specifies the destination that stores the mantissa to unite.	—	—	.L	.L	—	—	—	—	—	○	○
S2	Exponent storage destination	Specifies the destination that stores the exponent to unite.	.S	.S	.S	.S	—	—	—	—	—	○	○
R	Return value	.F type UNIF function available.*3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When bit data is specified, 32 continuous bits are processed. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled straddling the next channel.

*3 When the type other than .F is specified by return value, type conversion will be automatically conducted during conversion.

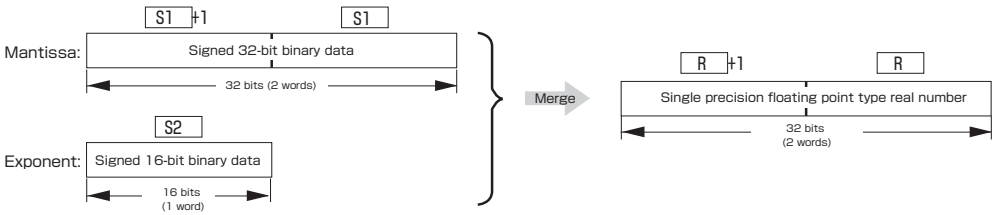
*4 The ".DF" suffix cannot be specified for device Z.

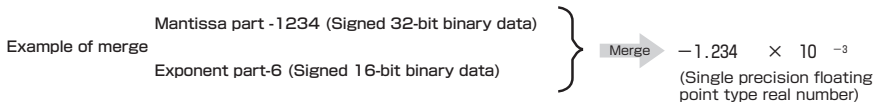
Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

The signed 32-bit BIN data specified by S1 is taken as the mantissa, the signed 16-bit BIN data specified by S2 is taken as the exponent, and the united 32-bit single precision floating point type real number value is stored in R.



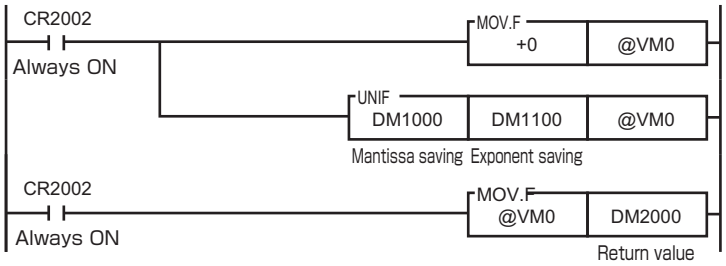


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

● **Example of format**

DM2000.F = **UNIF** (DM1000.L, DM1100.S)



Point

- **With the UNIF function, an error sometimes occurs as a mantissa section (DEC) and an exponent section (DEC) are being converted to single precision floating point type real number (BIN).**
- **When the operation result is outside the single precision floating point type real number range, "0" is stored in R.**

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

DM2000.F = EXP (**UNIF** (EM3000.L, EM3100.S))

Operation Description

The mantissa section currently stored in EM3000 (.L) is united with the exponent section currently stored in EM3100 (.S), the exponent function is calculated taking the resulting single precision floating point type real number value as the exponent, and the result is stored in DM2000.F.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "EXP function", page 7-130
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

EXP Exponent function operation

Return value*1 = EXP (Operation destination)

Exponent arithmetic operation

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Operation destination	Specifies the operation destination or the operation destination data.	.F	.F	.F	.F	.F	*3 .DF	—	—	○	○	○
R	Return value	The EXP function uses the .F type if the argument is U/S/D/L/F, and the .DF type for DF. Argument types are automatically converted to function types.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

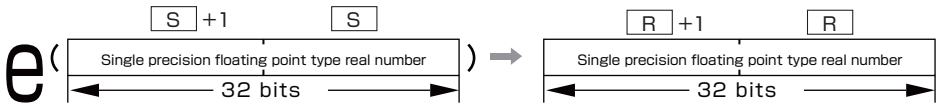
*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 If a type different from the function type specified in the return value, it will be automatically converted during conversion.
*3 The ".DF" suffix cannot be specified for device Z.

Point

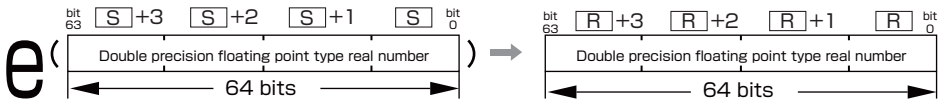
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **Refer to "INTG function", page 7-122 for details on the range of the single precision floating point type real number and double precision floating point type real number.**

Description of Operation

- S : When the suffix of the conversion destination is .U/.S/.D/.L/.F
  The exponent function value is obtained with the single precision floating point type real number stored in [S+1/S] as the exponent.
  The result is a 32-bit single-precision floating point type real number and stored in [R+1/R].



- S : When the suffix of the conversion destination is .DF
  The exponent function value is obtained with the double precision floating point type real number stored in [S+3/S+2/S+1/S] as the exponent.
  The result is a 64-bit double-precision floating point type real number and stored in [R+3/R+2/R+1/R].



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

● **Example of format**

DM2100.F = **EXP** (DM1100.F)

Point **When the value to operate on or the operation result is outside the single precision floating point type real number range, the value of [S] is stored in [R].**

Sample Program

● **Example of use**

Programmed Script

DM2000.F = **EXP** (DM1000.D)

Operation Description

The 32-bit BIN data stored in DM1000.D is automatically made into single precision floating point type real number, the value of the result is taken as the exponent, and e^x is calculated and stored in DM2000.F. (DM1000.D does not need to be converted by the FLOAT function, etc.)

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "FLOAT function", page 7-114
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

LOG Natural logarithm operation

Return value*1 = LOG (Operation destination)

Natural logarithm operation

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Operation destination	Specifies the operation destination or the operation destination data.	.F	.F	.F	.F	.F	*3 .DF	—	—	○	○	○
R	Return value	The LOG function uses the .F type if the argument is U/S/D/L/F, and the .DF type for DF. Argument types are automatically converted to function types.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

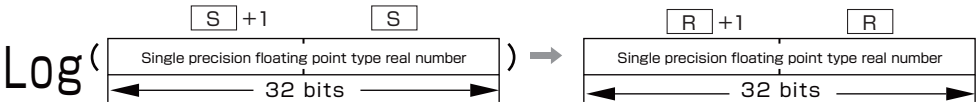
*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 If a type different from the function type specified in the return value, it will be automatically converted during conversion.
*3 The ".DF" suffix cannot be specified for device Z.

Point

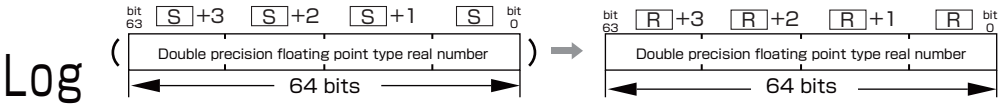
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **Refer to "INTG function", page 7-122 for details on the range of the single precision floating point type real number and double precision floating point type real number.**

Description of Operation

- S : When the suffix of the conversion destination is .U/.S/.D/.L/.F
  The natural logarithm of the single precision floating point type real number stored in [S+1/S] is obtained.
  The result is a 32-bit single-precision floating point type real number and stored in [R+1/R].



- S : When the suffix of the conversion destination is .DF
  The natural logarithm of the double precision floating point type real number stored in [**S+3/S+2/S+1/S**] is obtained.
  The result is a 64-bit double-precision floating point type real number and stored in [R+3/R+2/R+1/R].

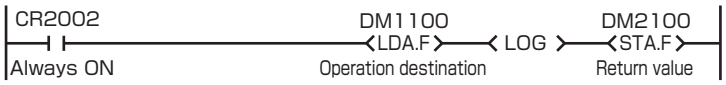


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

● **Example of format**

DM2100.F = **LOG** (DM1100.F)



Point **When the value to be operated on is 0 or a minus value, or the operation result is outside the floating point type real number range, the value of [S] is stored in [R].**

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

DM2000.L = INTG (**LOG** (DM1000.F))

Operation Description

The value resulting from having calculated the natural logarithm of the single precision floating point type real number stored in DM1000.F is converted to signed 32-bit BIN data and stored in DM2000.L.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "INTG function", page 7-122
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

LOG10 Constant logarithm operation

Return value*1 = LOG10 (Operation destination)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Operation destination	Specifies the operation destination or the operation destination data.	.F	.F	.F	.F	.F	*3 .DF	—	—	○	○	○
R	Return value	The LOG10 function uses the .F type if the argument is U/S/D/L/F, and the .DF type for DF. Argument types are automatically converted to function types.*2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

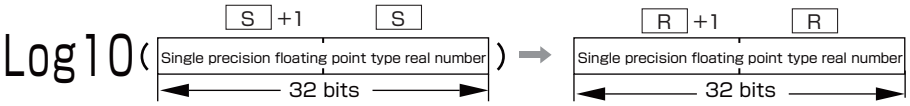
*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 If a type different from the function type specified in the return value, it will be automatically converted during conversion.
*3 The ".DF" suffix cannot be specified for device Z.

Point

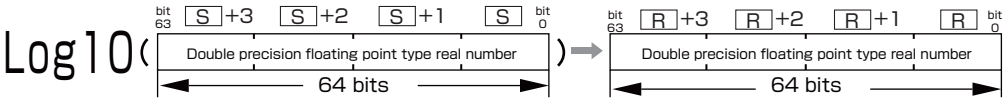
- **The LOG10 function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **Refer to "INTG function", page 7-122 for details on the range of the single precision floating point type real number and double precision floating point type real number.**

Description of Operation

- S : When the suffix of the conversion destination is .U/.S/.D/.L/.F
  The constant logarithm of the single precision floating point type real number stored in [S+1/S] is obtained.
  The result is a 32-bit single-precision floating point type real number and stored in [R+1/R].



- S : When the suffix of the conversion destination is .DF
  The constant logarithm of the double precision floating point type real number stored in [S+3/S+2/S+1/S] is obtained.
  The result is a 64-bit double-precision floating point type real number and stored in [R+3/R+2/R+1/R].

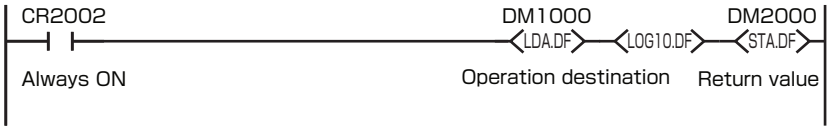


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

● **Example of format**

DM2000.DF = **Log10** (DM1000.DF)



Sample Program

Programmed Script

DM2500.L = INTG (**LOG10** (DM1500.DF))

Operation Description

The value resulting from having calculated the natural logarithm of the double precision floating point type real number stored in DM1500.DF is converted to signed 32-bit BIN data and stored in DM2500.L.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

FLOAT
TOF
DFLOAT
TODF
INTG
INT
DISF
UNIF
EXP
LOG
LOG10
RAD
DEG
SIN
COS
TAN
ASIN
ACOS
ATAN
ATAN2

RAD Convert degree (°) to radian unit

Return value*1 = RAD (conversion destination) (Radian)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
[S]	Conversion destination	Specifies the conversion destination or the conversion destination data.	.F	.F	.F	.F	.F	*3 .DF	—	—	○	○	○
[R]	Return value	The RAD function uses the .F type if the argument is U/S/D/L/F, and the .DF type for DF. Argument types are automatically converted to function types.*2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

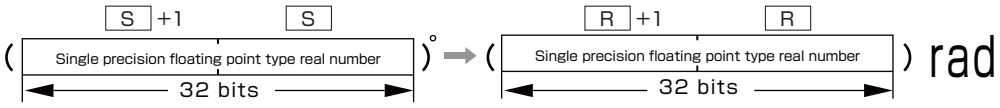
*1 The return value can be omitted. When omitted, the conversion result is stored in internal registers.
*2 If a type different from the function type specified in the return value, it will be automatically converted during conversion.
*3 The ".DF" suffix cannot be specified for device Z.

Point

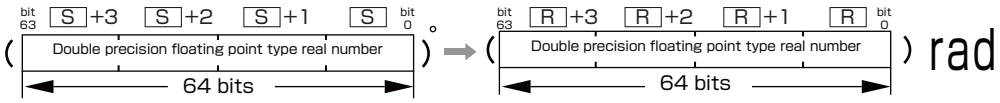
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **Refer to "INTG function", page 7-122 for details on the range of the single precision floating point type real number and double precision floating point type real number.**

Description of Operation

- [S] : When the suffix of the conversion destination is .U/.S/.D/.L/.F
  The angle (in °) of the single precision floating point type real number stored in [[S] +1/[S]] is converted into radian unit.
  The result is a 32-bit single-precision floating point type real number and stored in [[R]+1/[R]].



- [S] : When the suffix of the conversion destination is .DF
  The angle (in °) of the double precision floating point type real number stored in [[S]+3/[S]+2/[S]+1/[S]] is converted into radian unit.
  The result is a 64-bit double-precision floating point type real number and stored in [[R]+3/[R]+2/[R]+1/[R]].



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

Reference

Angle (°) and radian (rad) are calculated as follows:

$$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745\ (\text{rad}),\ 1\ (\text{rad}) = \frac{180}{\pi} \fallingdotseq 57.23^\circ$$

● Example of format

DM2100.F = **RAD** (DM1100.F)

Point

When the value to operate on is outside the single precision floating point type real number range, the value of [S] is stored in [R].

Sample Program

● Example of use

Programmed Script

DM2000.F = **RAD** (DM1000.D)

Operation Description

The angle (in °) of the 32-bit BIN data stored in DM1000.D is automatically converted to single precision floating point type real number, and the result is converted to radian unit and stored in DM2000.F.
(DM1000.D does not need to be converted by the FLOAT function, etc.)

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

DEG

Convert radian to degree (°) unit

Return value*1 = DEG (conversion destination)

Degree

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Conversion destination	Specifies the conversion destination or the conversion destination data.	.F	.F	.F	.F	.F	*3 .DF	—	—	○	○	○
R	Return value	The DEG function uses the .F type if the argument is U/S/D/L/F, and the .DF type for DF. Argument types are automatically converted to function types.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in internal registers.
*2 If a type different from the function type specified in the return value, it will be automatically converted during conversion.
*3 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **Refer to "INTG function", page 7-122 for details on the range of the single precision floating point type real number and double precision floating point type real number.**

Description of Operation

- S : When the suffix of the conversion destination is .U/.S/.D/.L/.F
  The angle (in radians) of the single precision floating point type real number stored in [S +1/S] is converted into degree (°) unit.
  The result is a 32-bit single-precision floating point type real number and stored in [R +1/R].

- S : When the suffix of the conversion destination is .DF
  The angle (in radians) of the double precision floating point type real number stored in [S +3/S +2/S +1/S] is converted into degree (°) unit.
  The result is a 64-bit double-precision floating point type real number and stored in [R +3/R +2/R +1/R].

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

Reference

Angle (°) and radian (rad) are calculated as follows:

$$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745\ (\text{rad}),\ 1\ (\text{rad}) = \frac{180}{\pi} \fallingdotseq 57.23^\circ$$

● **Example of format**

DM2100.F = **DEG** (DM1100.F)

Point **When the value to operate on is outside the single precision floating point type real number range, the value of [S] is stored in [R].**

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

DM2000.L = INTG (**DEG** (DM1000.F))

Operation Description

Convert the angle (in radians) of single precision floating point type real number stored in DM1000.F to degree (°) unit, convert the result to signed 32-bit BIN data, and then store it in DM2000.L.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "INTG function", page 7-122
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

SIN Angle (radian) to sine value

Return value*1 = SIN (Operation destination) (Sine)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Operation destination	Specifies the operation destination or the operation destination data.	.F	.F	.F	.F	.F	*3 .DF	—	—	○	○	○
R	Return value	The SIN function uses the .F type if the argument is U/S/D/L/F, and the .DF type for DF. Argument types are automatically converted to function types.*2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 If a type different from the function type specified in the return value, it will be automatically converted during conversion.
*3 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **The range of the single precision floating point type real number which can be handled with this function is as follows:**

  −1.6777216E+7 ≦ N ≦ −1.4E−45
  N = 0
  1.4E−45 ≦ N ≦ 1.6777216E+7
  (For the KV Nano Series, an absolute value in a range of 1.4E-45 to 1.2E-38 cannot be used.)

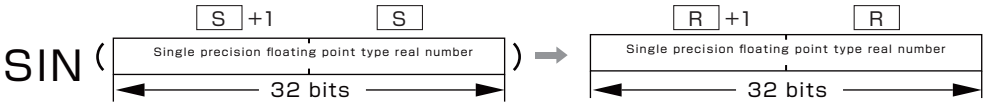
  (effective number of digits: approx. 7 digits)
- **The range of the double precision floating point type real number which can be handled with this function is as follows:**

  −9.007199254740992E+15 ≦ N ≦ −2.23E−308
  N = 0
  +2.23E−308 ≦ N ≦ +9.007199254740992E+15

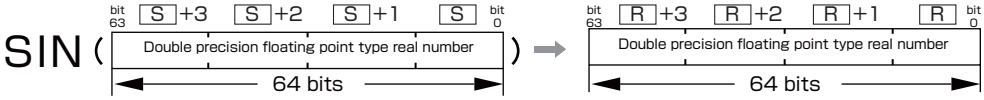
  (effective number of digits: approx. 16 digits)

Description of Operation

- [S] : When the suffix of the conversion destination is .U/.S/.D/.L/.F
  The sine (SIN) value of the single precision floating point type real number angle (in radians) stored in [[S]+1/[S]] is obtained.
  The result is a 32-bit single-precision floating point type real number and stored in [[R]+1/[R]].



- [S] : When the suffix of the conversion destination is .DF
  The sine (SIN) value of the double precision floating point type real number angle (in radians) stored in [S +3/ S +2/ S +1/ S] is obtained.
  The result is a 64-bit double-precision floating point type real number and stored in [R +3/ R +2/ R +1/ R].



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

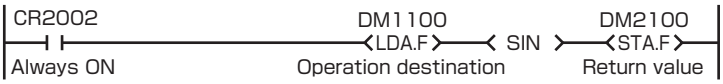
Reference

Angle (°) and radian (rad) are calculated as follows:

$$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745\ (\text{rad}),\ 1\ (\text{rad}) = \frac{180}{\pi} \fallingdotseq 57.23^\circ$$

● Example of format

DM2100.F = **SIN** (DM1100.F)



Point

When the value to be operated on is outside the single precision floating point type real number range that can be handled by the SIN function, the value of [S] is stored in [R].

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000.F = **SIN** (RAD (DM1000.F))

Operation Description

The angle (in degrees) of the single-precision floating point type real number stored in DM1000.F is converted to radians, the sine value is calculated, and the result is stored in DM2000.F.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "RAD function", page 7-136
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

COS Angle (radian) to cosine value

Return value*1 = COS (Operation destination)

Cosine

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Operation destination	Specifies the operation destination or the operation destination data.	.F	.F	.F	.F	.F	*3 .DF	—	—	○	○	○
R	Return value	The COS function uses the .F type if the argument is U/S/D/L/F, and the .DF type for DF. Argument types are automatically converted to function types.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 If a type different from the function type specified in the return value, it will be automatically converted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **The range of the single precision floating point type real number which can be handled with this function is as follows:**

  −1.6777216E+7 ≦ N ≦ −1.4E−45
  N = 0
  1.4E−45 ≦ N ≦ 1.6777216E+7
  (For the KV Nano Series, an absolute value in a range of 1.4E-45 to 1.2E-38 cannot be used.)

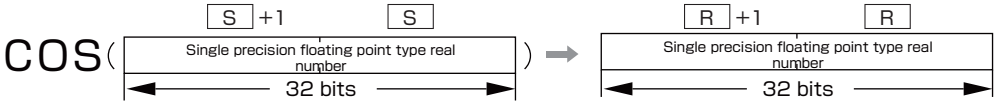
  (effective number of digits: approx. 7 digits)
- **The range of the double precision floating point type real number which can be handled with this function is as follows:**

  −9.007199254740992E+15 ≦ N ≦ −2.23E−308
  N = 0
  +2.23E−308 ≦ N ≦ +9.007199254740992E+15

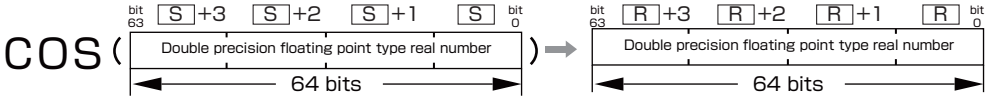
  (effective number of digits: approx. 16 digits)

Description of Operation

- [S] : When the suffix of the conversion destination is .U/.S/.D/.L/.F
  The cosine (COS) value of the single precision floating point type real number angle (in radians) stored in [[S]+1/[S]] is obtained.
  The result is a 32-bit single-precision floating point type real number and stored in [[R]+1/[R]].



- [S] : When the suffix of the conversion destination is .DF
  The cosine (COS) value of the double precision floating point type real number angle (in radians) stored in [[S]+3/[S]+2/[S]+1/[S]] is obtained.
  The result is a 64-bit double-precision floating point type real number and stored in [[R]+3/[R]+2/[R]+1/[R]].



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

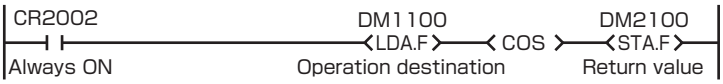
Reference

Angle (°) and radian (rad) are calculated as follows:

$$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745\ (\mathrm{rad}),\ 1\ (\mathrm{rad}) = \frac{180}{\pi} \fallingdotseq 57.23^\circ$$

● Example of format

DM2100.F = **COS** (DM1100.F)



Point

When the value to be operated on is outside the single precision floating point type real number range that can be handled by the COS function, the value of [S] is stored in [R].

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000.F = **COS** (RAD (DM1000.F))

Operation Description

The angle (° unit) of the single precision floating point type real number stored in DM1000.F is converted to radian unit, and the cosine value is calculated and the result is stored in DM2000.F.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "RAD function", page 7-136
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

TAN Angle (radian) to tangent value

Return value*1 = TAN (Operation destination)

Tangent

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Operation destination	Specifies the operation destination or the operation destination data.	.F	.F	.F	.F	.F	*3 .DF	—	—	○	○	○
R	Return value	The TAN function uses the .F type if the argument is U/S/D/L/F, and the .DF type for DF. Argument types are automatically converted to function types.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 If a type different from the function type specified in the return value, it will be automatically converted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **The range of the single precision floating point type real number which can be handled with this function is as follows:**

  −1.6777216E+7 ≦ N ≦ −1.4E−45
  N = 0
  1.4E−45 ≦ N ≦ 1.6777216E+7
  (For the KV Nano Series, an absolute value in a range of 1.4E-45 to 1.2E-38 cannot be used.)

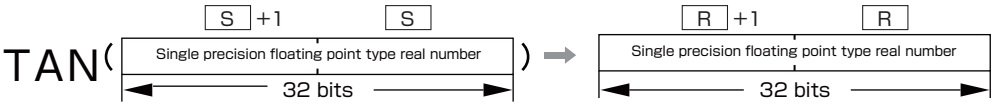
  (effective number of digits: approx. 7 digits)
- **The range of the double precision floating point type real number which can be handled with this function is as follows:**

  −9.007199254740992E+15 ≦ N ≦ −2.23E−308
  N = 0
  +2.23E−308 ≦ N ≦ +9.007199254740992E+15

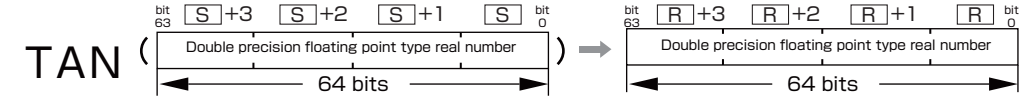
  (effective number of digits: approx. 16 digits)

Description of Operation

- [S] : When the suffix of the conversion destination is .U/.S/.D/.L/.F
  The tangent (TAN) value of the single precision floating point type real number angle (in radians) stored in [[S]+1/[S]] is obtained.
  The result is a 32-bit single-precision floating point type real number and stored in [[R]+1/[R]].



- [S] : When the suffix of the conversion destination is .DF
  The tangent (TAN) value of the double precision floating point type real number angle (in radians) stored in [[S]**+3/**[S]**+2/**[S]**+1/**[S]] is obtained.
  The result is a 64-bit double-precision floating point type real number and stored in [[R]+3/ [R]+2/ [R]+1/ [R]].



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

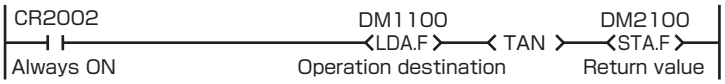
Reference

Angle (°) and radian (rad) are calculated as follows:

$$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745\ (\text{rad}),\ 1\ (\text{rad}) = \frac{180}{\pi} \fallingdotseq 57.23^\circ$$

● Example of format

DM2100.F = **TAN** (DM1100.F)



Point

When the value to be operated on is outside the single precision floating point type real number range that can be handled by the TAN function, the value of [S] is stored in [R].

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000.F = **TAN** (RAD (DM1000.F))

Operation Description

The angle (in °) of the single precision floating point type real number stored in DM1000.F is converted to radian unit, the tangent value is calculated and the result is stored in DM2000.F.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "RAD function", page 7-136
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ASIN Sine value to angle (radians)

Return value*1 = ASIN (Operation destination)

Arc sine

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Operation destination	Specifies the operation destination or the operation destination data.	.F	.F	.F	.F	.F	*3 .DF	—	—	○	○	○
R	Return value	The ASIN function uses the .F type if the argument is U/S/D/L/F, and the .DF type for DF. Argument types are automatically converted to function types.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

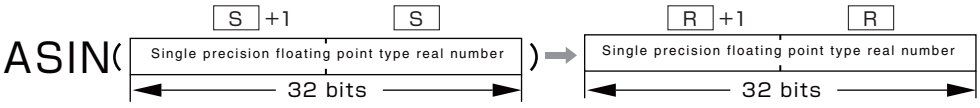
*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 If a type different from the function type specified in the return value, it will be automatically converted during conversion.
*3 The ".DF" suffix cannot be specified for device Z.

Point

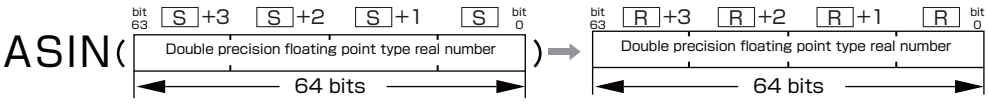
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **Refer to "INTG function", page 7-122 for details on the range of the single precision floating point type real number and double precision floating point type real number.**

Description of Operation

- S : When the suffix of the conversion destination is .U/.S/.D/.L/.F
  The angle (in radians) is calculated from the sine (SIN) value of the single precision floating point type real number stored in [S +1/S].
  The result is a 32-bit single-precision floating point type real number and stored in [R +1/R].



- S : When the suffix of the conversion destination is .DF
  The angle (in radians) is calculated from the sine (SIN) value of the double precision floating point type real number stored in [S +3/S +2/S +1/S].
  The result is a 64-bit double-precision floating point type real number and stored in [R +3/R +2/R +1/R].



"About operation flag (CR2009 to CR2012)", page 1-18
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

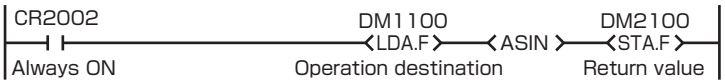
Reference

Angle (°) and radian (rad) are calculated as follows:

$$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745\ (\text{rad}),\ 1\ (\text{rad}) = \frac{180}{\pi} \fallingdotseq 57.23^\circ$$

● Example of format

DM2100.F = **ASIN** (DM1100.F)



Point

When the value to operate on is outside of the range -1.0 to 1.0, the value of [S] is stored in [R].

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000.F = DEG (**ASIN** (DM1000.F))

Operation Description

The angle (radian unit) is calculated from the sine value of the single precision floating point type real number stored in DM1000.F, and the result is converted to degrees (° unit) and stored in DM2000.F.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "DEG function", page 7-138
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ACOS Cosine value to angle (radians)

Return value*1 = ACOS (Operation destination)

Arc cosine

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Operation destination	Specifies the operation destination or the operation destination data.	.F	.F	.F	.F	.F	*3 .DF	—	—	○	○	○
R	Return value	The ACOS function uses the .F type if the argument is U/S/D/L/F, and the .DF type for DF. Argument types are automatically converted to function types.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 If a type different from the function type specified in the return value, it will be automatically converted during conversion.

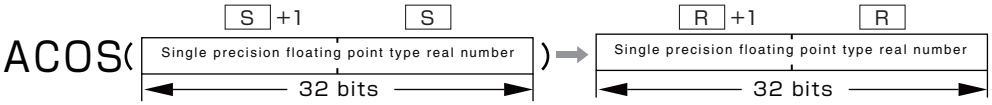
*3 The ".DF" suffix cannot be specified for device Z.

Point

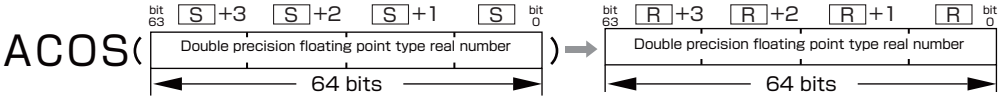
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **Refer to "INTG function", page 7-122 for details on the range of the single precision floating point type real number and double precision floating point type real number.**

Description of Operation

- S : When the suffix of the conversion destination is .U/.S/.D/.L/.F
  The angle (in radians) is calculated from the cosine (COS) value of the single precision floating point type real number stored in [S +1/S].
  The result is a 32-bit single-precision floating point type real number and stored in [R +1/R].



- S : When the suffix of the conversion destination is .DF
  The angle (in radians) is calculated from the cosine (COS) value of the double precision floating point type real number stored in [S +3/S +2/S +1/S].
  The result is a 64-bit double-precision floating point type real number and stored in [R +3/R +2/R +1/R].



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

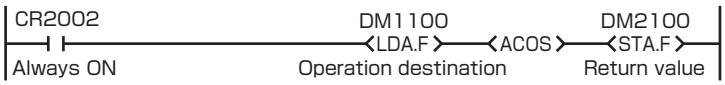
Reference

Angle (°) and radian (rad) are calculated as follows:

$$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745\ (\text{rad}),\ 1\ (\text{rad}) = \frac{180}{\pi} \fallingdotseq 57.23^\circ$$

● **Example of format**

DM2100.F = **ACOS** (DM1100.F)



Point

When the value to operate on is outside of the range -1.0 to 1.0, the value of [S] is stored in [R].

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

DM2000.F = DEG (**ACOS** (DM1000.F))

Operation Description

The angle (radian unit) is calculated from the cosine value of the single precision floating point type real number stored in DM1000.F, and the result is converted to degrees (° unit) and stored in DM2000.F.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "DEG function", page 7-138
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ATAN Tangent value to angle (radian)

Return value*1 = ATAN (Operation destination)

Arc tangent

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Operation destination	Specifies the operation destination or the operation destination data.	.F	.F	.F	.F	.F	*3 .DF	—	—	○	○	○
R	Return value	The ATAN function uses the .F type if the argument is U/S/D/L/F, and the .DF type for DF. Argument types are automatically converted to function types.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 If a type different from the function type specified in the return value, it will be automatically converted during conversion.

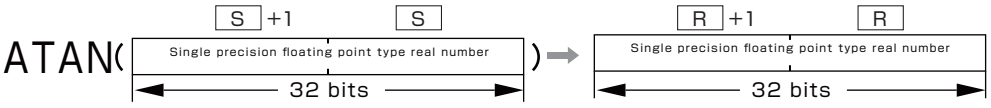
*3 The ".DF" suffix cannot be specified for device Z.

Point

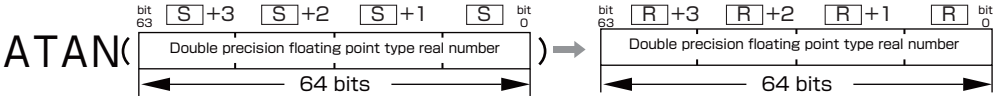
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **Refer to "INTG function", page 7-122 for details on the range of the single precision floating point type real number and double precision floating point type real number.**

Description of Operation

- S : When the suffix of the conversion destination is .U/.S/.D/.L/.F
  The angle (in radians) is calculated from the tangent (TAN) value of the single precision floating point type real number stored in [S +1/S].
  The result is a 32-bit single-precision floating point type real number and stored in [R +1/R].



- S : When the suffix of the conversion destination is .DF
  The angle (in radians) is calculated from the tangent (TAN) value of the double precision floating point type real number stored in [S+3/S+2/S+1/S].
  The result is a 64-bit double-precision floating point type real number and stored in [R+3/R+2/R+1/R].



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

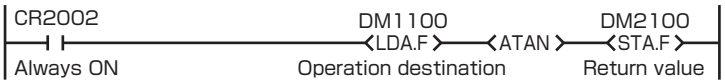
Reference

Angle (°) and radian (rad) are calculated as follows:

$$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745\ (\text{rad}),\ 1\ (\text{rad}) = \frac{180}{\pi} \fallingdotseq 57.23^\circ$$

● **Example of format**

DM2100.F = **ATAN** (DM1100.F)



Point

When the value to operate on is outside the single precision floating point type real number range, the value of [S] is stored in [R].

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

DM2000.F = DEG (**ATAN** (DM1000.F))

Operation Description

The angle (radian unit) is calculated from the tangent value of the single precision floating point type real number stored in DM1000.F, and the result is converted to degrees (° units) and stored in DM2000.F.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "DEG function", page 7-138
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ATAN2 Coordinates to angle (radian)

Return value*1 = ATAN2 (Operation destination 1, operation destination 2)

Arc tangent2

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S1	Operation destination 1	Specifies the operation destination 1 or the operation destination data 1.	.F	.F	.DF	.DF	.F	*3 .DF	—	—	○	○	○
S2	Operation destination 2	Specifies the operation destination 2 or the operation destination data 2.	.F	.F	.DF	.DF	.F	*3 .DF	—	—	○	○	○
R	Return value	The ATAN2 function uses the .F type if the argument is U/S/F, and the .DF type for D/L/DF. Argument types are automatically converted to function types.* 2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

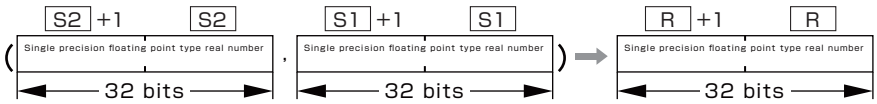
*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 If a type different from the function type specified in the return value, it will be automatically converted during conversion.
*3 The ".DF" suffix cannot be specified for device Z.

Point

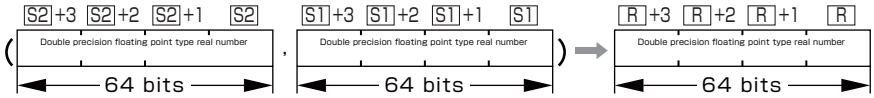
- **The ATAN2 function can be used only by KV STUDIO ver.9.0 or later.**
- **This function can be specified only with the KV-8000 series CPU units and KV-7000 Series CPU units that have CPU function version 2.0 or later.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **Refer to "INTG function", page 7-122 for details on the range of the single precision floating point type real number and double precision floating point type real number.**

Description of Operation

- When the suffix of S1 conversion destination 1 and S2 conversion destination 2 is .U/.S/.F
  The angle (in radians) relative to X axis of a single-precision floating point type real number is calculated from the Y coordinate obtained by converting the value specified by [S1+1/S1] to a single-precision floating point type real number and the X coordinate obtained by converting the value specified by [S2+1/S2] to a single-precision floating point type real number.
  The result is a 32-bit single-precision floating point type real number and stored in [R+1/R].



- When the suffix of S1 conversion destination 1 and S2 conversion destination 2 is .D/.L/.DF
  The angle (in radians) relative to X axis of a double-precision floating point type real number is calculated from the Y coordinate obtained by converting the value specified by [S1+3/S1+2/S1+1/S1] to a double-precision floating point type real number and the X coordinate obtained by converting the value specified by [S2+3/S2+2/S2+1/S2] to a double-precision floating point type real number.
  The result is a 64-bit double-precision floating point type real number and stored in [R+3/R+2/R+1/R].

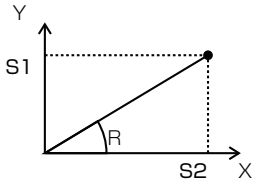


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Floating Point Instructions".

Reference

The ATAN instruction uses the TAN value to find the angle up to -π/2(rad) < R < π/2(rad). The ATAN2 instruction uses coordinates (S2, S1) to find the angle up to -π(rad) < R ≦ π(rad).



$$1^\circ = \frac{\pi}{180} \fallingdotseq 0.01745\ (\text{rad}),\ 1\ (\text{rad}) = \frac{180}{\pi} \fallingdotseq 57.23^\circ$$

When S1=0 and S2≧0, R (return value)=0°=0(rad).

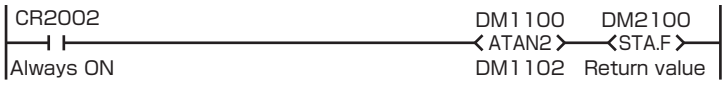
When S1=0 and S2<0, R (return value)=180°= π(rad).

When S1>0 and S2=0, R (return value)=90°=π/2(rad).

When S1<0 and S2=0, R (return value)=-90°=-π/2(rad).

● Example of format

DM2100.F = **ATAN2** (DM1100.F, DM1102.F)



Point

When the value to operate on is outside the single precision floating point type real number range, the value of R does not change.

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000.F = DEG (**ATAN2** (DM1000.F, DM1002.F))

Operation Description

The angle (in radians) relative to X axis of a single precision floating point type real number is calculated from the single precision floating point type real number X coordinate stored in DM1000.F and the single precision floating point type real number Y coordinate stored in DM1002.F, and the result is converted to degree (°) unit and stored in DM2000.F.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "DEG function", page 7-138
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ASC Convert BIN data to HEX ASCII code

HEX ASCII code conversion

Return value*1 = ASC (conversion destination)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Conversion destination*2	Specifies the destination that stores the HEX ASCII text code to be converted or the conversion destination data.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	.U type ASC function available.* 3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in internal registers.

*2 Even when a relay other than the leading relay (e.g. R1002 or R1005) of the channel for the bit data has been specified for the conversion destination, the function operates normally straddling the next channel.

*3 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.

*4 The ".DF" suffix cannot be specified for device Z.

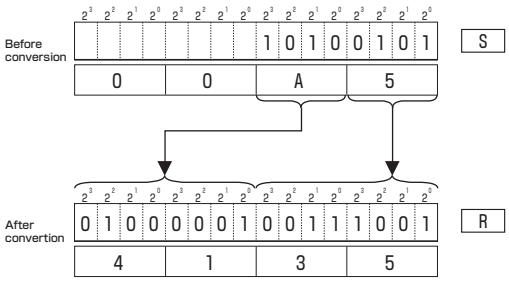
Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

The value of low-order 8 bits of the BIN data stored in S is converted to HEX ASCII text code, and the resulting 16-bit data is stored in R.



"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● Example of format

DM2000.U = **ASC** (DM1000.U)

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000.U = **ASC** (SRA (DM1000.U, 8))

Operation Description

The BIN data currently stored in DM1000 is shifted to the right by 8 bits.
The value of the low-order 8 bits (high-order 8 bits before the shift) of the shift result is converted to HEX ASCII code, and the result is stored in DM2000.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "SRA function", page 7-38
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ASC
RASC
DASC
STR
RDASC
HASC
RHASC
FASC
RFASC
VAL
LEN
SRGHT
RIGHT
SLEFT
LEFT
SMID
MID
SRPLC
REPLACE
SINS
INSERT
SDEL
DELETE
TRIM
LTRIM
RTRIM
STRIM
SPLIT
SFIND
SFINDN
INSTR
FIND
DISS
UNIS
CHR
CPSASC
RCPSASC

RASC Convert HEX ASCII code to BIN data

HEX ASCII code reverse conversion

Return value*1 = RASC (conversion destination)

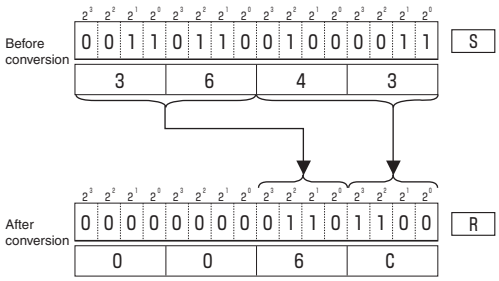
Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Conversion destination*2*3	Specifies the desitnation that stores the BIN data to be converted or the conversion destination data.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	.U type RASC function available.* 4	.U	.S	.D	.L	.F	*5 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in internal registers.
*2 Though 32-bit data can also be used for the conversion destination data, only the low-order 16 bits are targeted for conversion.
*3 Even when a relay other than the leading relay (e.g. R1002 or R1005) of the channel for the bit data has been specified for the conversion destination, the function operates normally straddling the next channel.
*4 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.
*5 The ".DF" suffix cannot be specified for device Z.

Point
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

The HEX ASCII text code (two characters) of the 16 bits stored in S are converted to BIN data, and the 16-bit data (low-order 8 bits are the conversion value, and the high-order 8 bits are 0) of the result is stored in R.



"About operation flag (CR2009 to CR2012)", page 1-18
"ASCII Code Table", page A-53
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

Point
- **S must store an ASCII code with both high-order 8 bits and lower-order 8 bits being 0 to 9 (30H to 39H) or A to F (41H to 46H), or a value (ASCII code) with high-order 8 bits being NUL (00H) and low-order 8 bits being 0 to 9 (30H to 39H) or A to F (41H to 46H). When either of the high-order 8 bits or low-order 8 bits is outside this range, the function is not executed.**
- **When there is an ASCII code that cannot be used, the value of S is stored in R.**

● **Example of format**

DM2000.U = **RASC** (DM1000.U)

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

DM2000.U = **RASC** (SWAP (DM1000))

Operation Description

The values of the upper byte (upper eight bits) and the lower byte (lower eight bits) of the HEX annotation ASCII text code currently stored in DM1000 are swapped.
As a result of the swap, the bytes are converted to 16-bit BIN data, and the result is stored in DM2000.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "SWAP function", page 7-100
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

DASC Convert BIN data to DEC ASCII text string

Decimal ASCII code conversion

Return value*1 = DASC (conversion destination, conversion parameter)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Conversion destination*2*3	Specifies the desitnation that stores the BIN data to be converted or the conversion destination data.	.U	.S	.D	.L	—	—	—	—	*4 ○	○	○
S2	Conversion parameter*5	Specifies the destination to which the conversion parameter is stored, or the parameter. (0 to 3)	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value*3*6	.T type DASC function available.	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.

*2 When bit data is specified, 16/32 continuous bits are processed.
When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

*3 For bit data, specify the start of the channel.

*4 Constants cannot be specified by "$".

*5 Specify the conversion parameter within the range 0 to 3.
S2 conversion parameter explanation

Value	Details	CR2814	CR2815
0	Zero suppress OFF, sign omission OFF	0	0
1	Zero suppress ON, sign omission OFF	1	0
2	Zero suppress OFF, sign omission ON	0	1
3	Zero suppress ON, sign omission ON	1	1

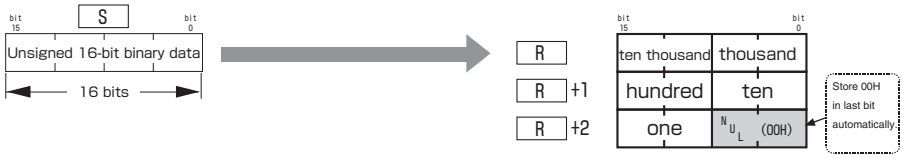
Point **The function might not operate correctly when a value outside of the range is specified to the conversion parameter.**

*6 T, C, CR, CTC, CTH, Z cannot be used.

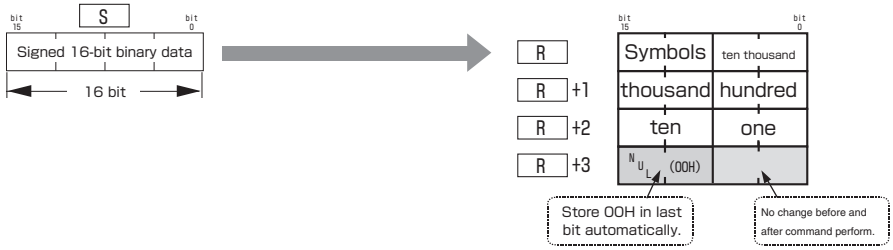
Description of Operation

The BIN data stored in S1 is converted to DEC ASCII code text string according to the conversion parameter specified with S2, and the result is stored in R.

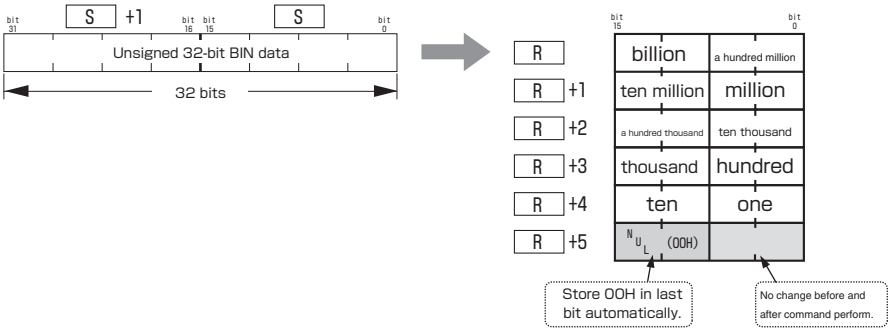
- S1 : When the conversion destination is unsigned 16-bit data (.U)



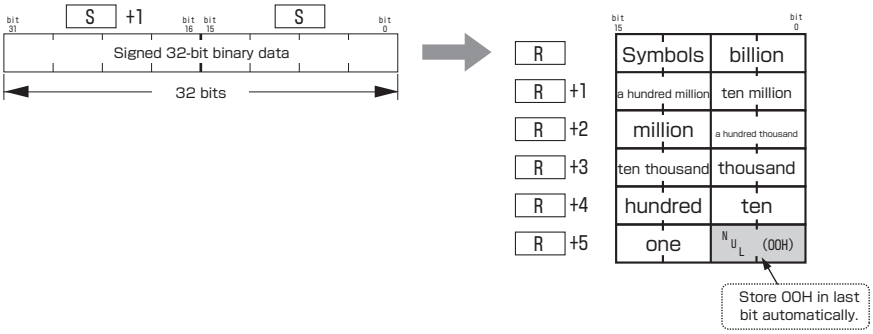
- S1 : When the conversion destination is signed 16-bit data (.S)



- S1 : When the conversion destination is unsigned 32-bit data (.D)



- S1 : When the conversion destination is signed 32-bit data (.L)

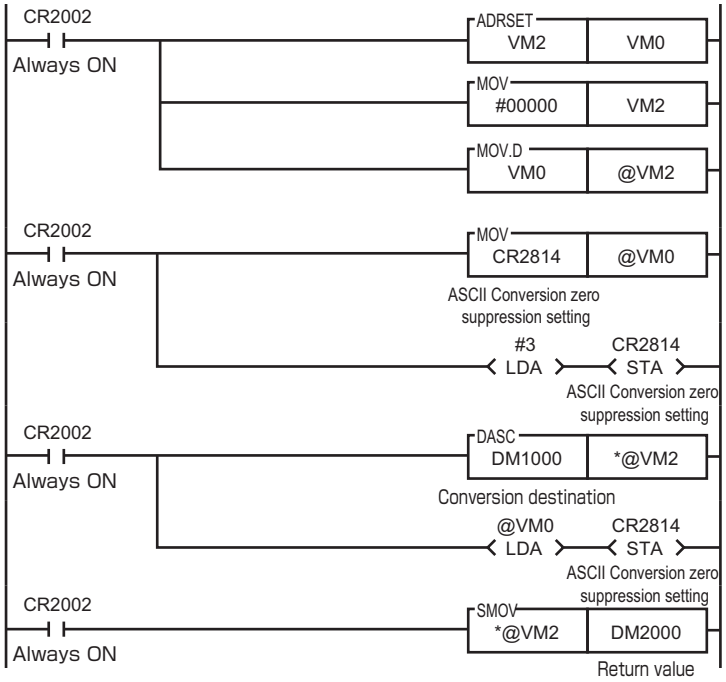


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

DM2000.T = **DASC** (DM1000 , 3)



Point

- **When a constant is used for the conversion destination data, it cannot be specified in HEX ($).**
- **Only 1-byte characters (ASCII code) can be specified for data conversion.**
- **Attention is required when changing or looking up CR2814/CR2815 by interrupt programs.**
- **When any of the following conditions is established, the end code (NUL(00H)) is stored in [R].**
  - **When a bit other than the start of the channel is specified when bit data is specified for [R].**
  - **When a timer/counter is specified to [R] by indirect specification.**

Sample Program

● When this function is used in combination with other functions

Programmed Script

EM2000.T = **DASC** (TBIN (DM1000.D), 0)

Operation Description

The 8-digit BCD data stored in DM1000.D is converted to BIN data.
The value resulting from conversion is converted to decimal ASCII code text string, and the result is stored in EM2000.T.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "TBIN function", page 7-82
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

STR Convert numerical value data to ASCII code

Decimal ASCII code conversion

Return value*1 = STR (conversion destination)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Conversion destination*2*4	Specifies the desitnation that stores the BIN data to be converted or the conversion destination data.	.U	.S	.D	.L	.F	—	—	—	○	○	○
R	Return value*3*5	.T type STR function available	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/ memory.

*2 When bit data is specified, 16/32 continuous bits are processed.
When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

*3 For bit data, specify the start of the channel.

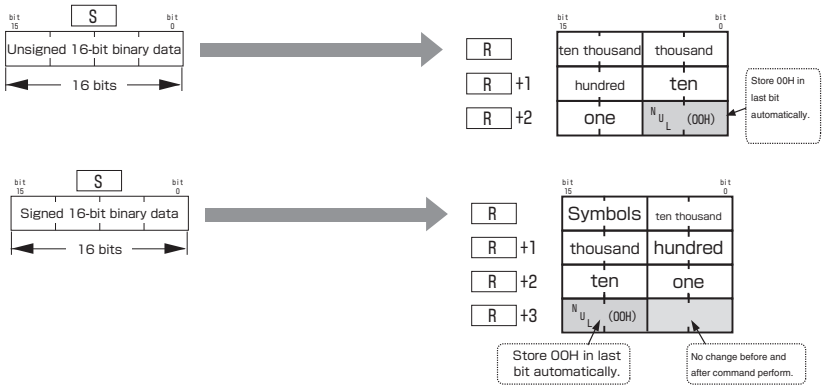
*4 Constants cannot be specified by "$".

*5 T, C, CR, CTC, CTH, Z cannot be used.

Description of Operation

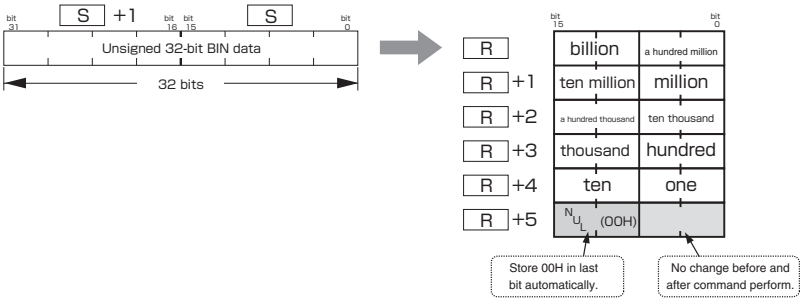
DEC ASCII text string conversion or single precision floating point type text string conversion is performed according to the argument type specified by S.

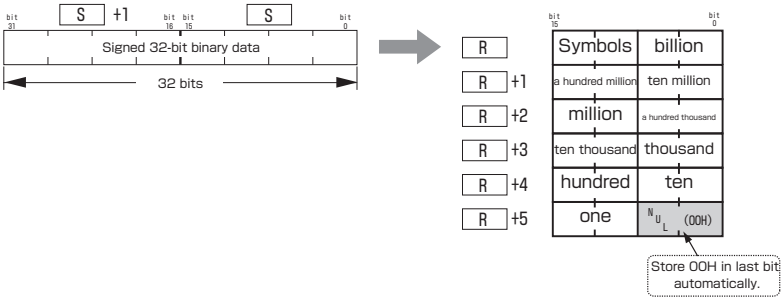
- When S conversion destination is 16 bits: The same operation as the DASC function is performed.



For details on operation, refer to "DASC function", page 7-158.

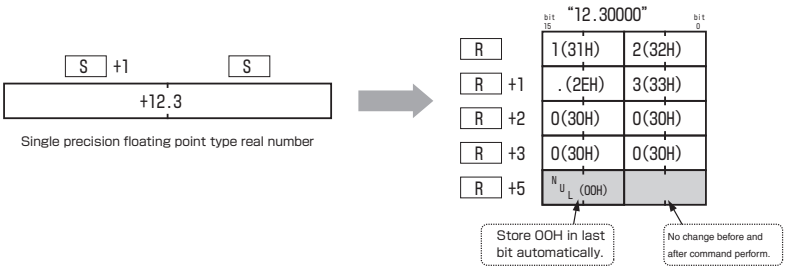
- When S conversion destination is 32 bits: The same operation as the DASC function is performed.





For details on operation, refer to "DASC function", page 7-158.

- When S conversion destination is single precision floating point type: The same operation as the FASC function* is performed.



*Conversion is performed with the conversion parameter=3 (CR2814=ON, CR2815=ON: zero suppress, symbol omitted).

For details on operation, refer to "FASC function", page 7-174.

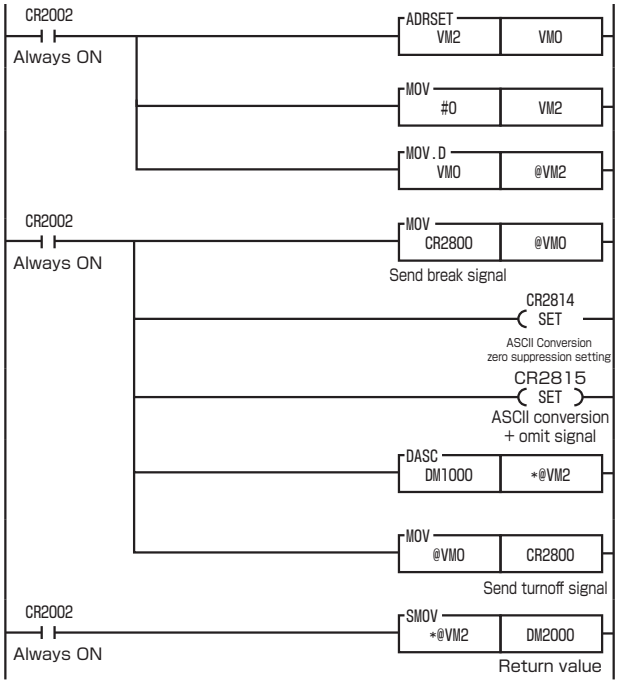
"About operation flag (CR2009 to CR2012)", page 1-18

Point

- **When a constant is used for the conversion destination data, it cannot be specified in HEX ($).**
- **When any of the following conditions is established, the end code (NUL(00H)) is stored in R.**
  - **When a bit other than the start of the channel is specified when bit data is specified for R.**
  - **When a timer/counter is specified to R by indirect specification.**

● **Example of format**

DM2000.T = **STR** (DM1000)



MEMO

RDASC Convert DEC ASCII text string to BIN data

Decimal ASCII code reverse conversion

Return value*1 = RDASC (conversion destination)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Conversion destination*2*3	Specifies the target that stores the decimal ASCII string to be converted or the decimal ASCII code string.	—	—	—	—	—	—	—	.T	○	○	○
R	Return value*4	.L type RDASC function available.* 5	.U	.S	.D	.L	.F	*6 .DF	—	—	—	○	○

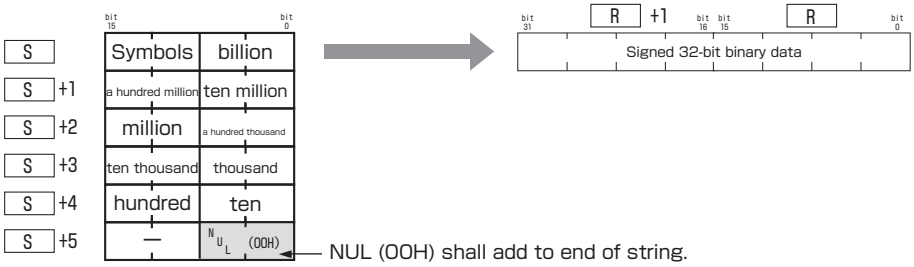
*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 T, C, CR, CTC, CTH, Z cannot be used.
*4 When bit data is specified, 32 continuous bits are processed.
When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.
*5 When the type other than .L is specified by return value, type conversion will be automatically conducted during conversion.
*6 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

The DEC ASCII text string currently stored starting from S is converted to signed 32-bit BIN data, and the result is stored in R.

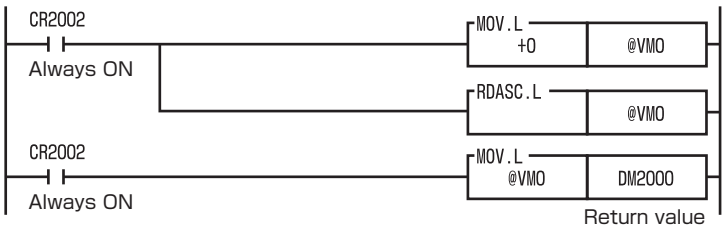


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● Example of format

DM2000.L = **RDASC** (DM1000.T)



Point

- **Specify a text string comprising $30 to $39 ("0" to "9") and $2B, $2D ("±": only in the case of signed conversion) to the decimal ASCII text string specified by the conversion destination data.**
- **When an ASCII text string other than the above has been specified to [S], the function is not executed.**
- **When the ASCII text string specified by [S] does not contain a NUL (00H), the function converts text strings up to 11 characters.**
- **When any of the following conditions is established, "0" is stored in [R].**
  - **When there is an invalid character in [S].**
  - **When the value after conversion is outside the signed 32-bit BIN data range**
  - **When a bit other than the start of the channel is specified when bit data is specified for [S].**
  - **When a timer/counter is specified to [S] by indirect specification.**

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000.L = **RDASC** (SLEFT (DM1000.T, 7))

Operation Description

The leading seven characters of the decimal ASCII text string currently stored starting from DM1000.T are cut, and converted to signed 32-bit BIN data, and the result is stored in DM2000.L.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "SLEFT function", page 7-188
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ASC
RASC
DASC
STR
RDASC
HASC
RHASC
FASC
RFASC
VAL
LEN
SRGHT
RIGHT
SLEFT
LEFT
SMID
MID
SRPLC
REPLACE
SINS
INSERT
SDEL
DELETE
TRIM
LTRIM
RTRIM
STRIM
SPLIT
SFIND
SFINDN
INSTR
FIND
DISS
UNIS
CHR
CPSASC
RCPSASC

HASC Convert BIN data to HEX ASCII text string

Return value*1 = HASC (conversion destination, number of conversion digits)

HEX ASCII code conversion

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Conversion destination*2	Specifies the desitnation that stores the BIN data to be converted or the conversion destination data. (unsigned BIN data)	.U	.U	.D	.D	—	—	—	—	○	○	○
n	Number of conversion digits*5	Specifies the number of digits of the numerical value (HEX) to store the conversion result to. (0 to 8)	.U	.U	.U	.U	—	—	—	—	*3 ○	○	○
R	Return value*4*6	.T type HASC function available.	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.

*2 When bit data is specified, 16/32 continuous bits are processed.
When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

*3 Constants must be specified by "$". They cannot be specified by "#".

*4 The .T suffix must be specified to the storage destination for the return value.
In addition, when bit data is specified for the return value, it is necessary to specify the start of the channel.

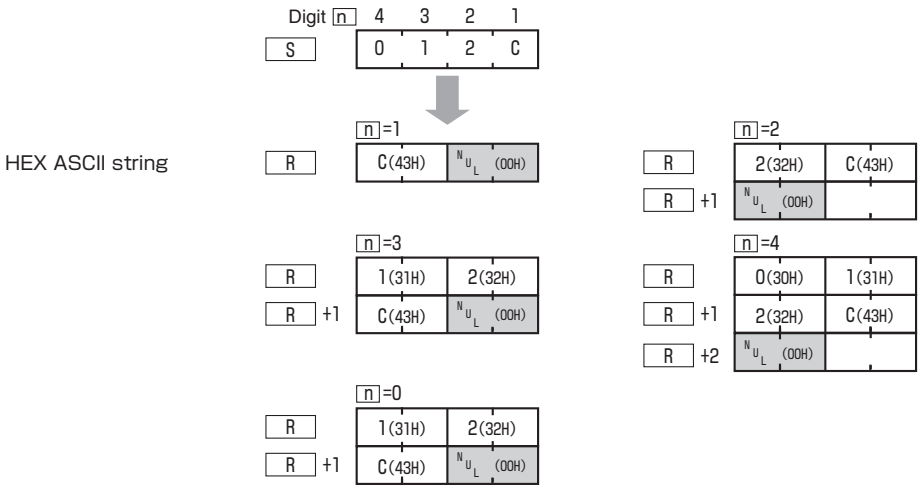
*5 Specify the number of digits (lower digit) to store the conversion result within the range 0 to 8 to the number of conversion digits.
(When "0" is specified, zero suppress is ON, and the number of effective digits is stored.)

*6 T, C, CR, CTC, CTH, Z cannot be used.

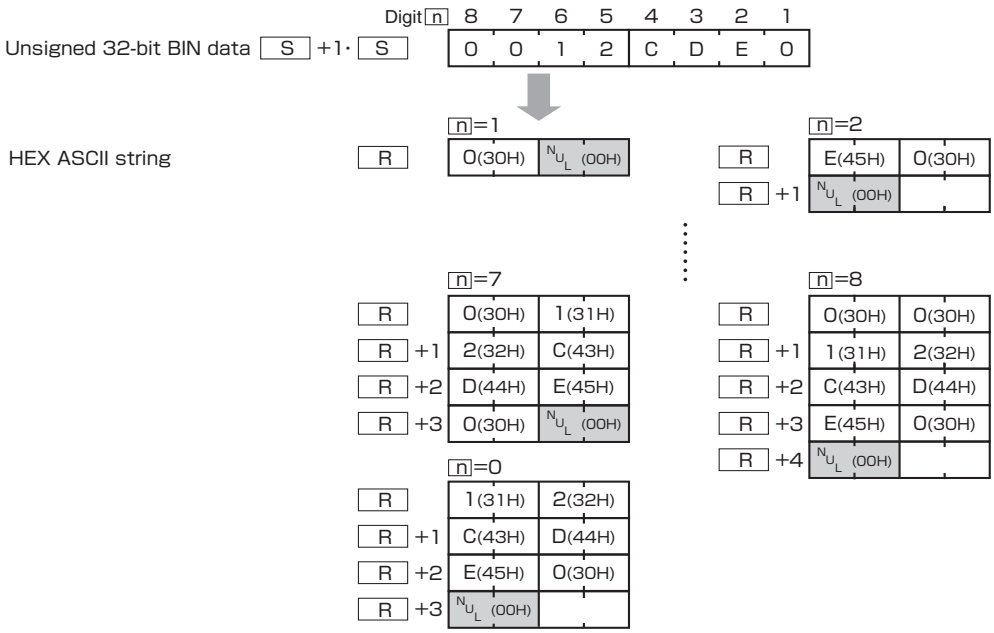
Description of Operation

The BIN data stored in S is converted to a HEX ASCII text string appended with the end code NUL (00H), and the result is stored in R by the specified n. (When n = 0, all the data of S is converted with zeros suppressed.)

- S : When the conversion destination is unsigned 16-bit data (.U)



- S : When the conversion destination is unsigned 32-bit data (.D)

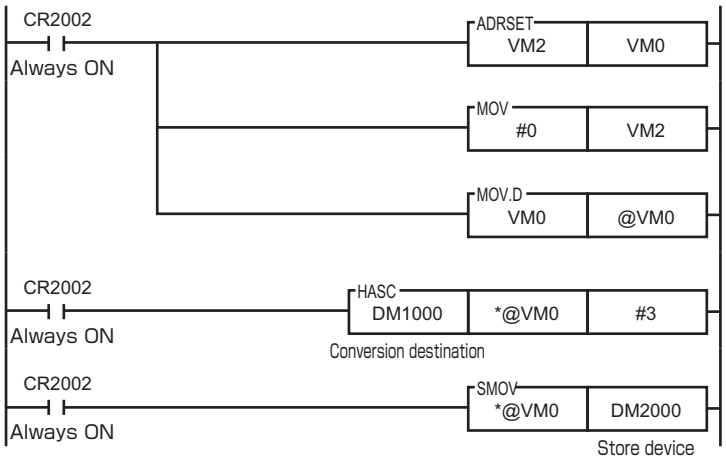


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● Example of format

DM2000.T = HASC (DM1000, 3)



Point

- **When a constant is used for the conversion destination data, it cannot be specified in decimal (#).**
- **When any of the following conditions is established, the end code (NUL(00H)) is stored in [R].**
  - **When a bit other than the start of the channel is specified when bit data is specified for [R].**
  - **When a timer/counter is specified to [R] by indirect specification.**
  - **When 5 or more is specified to [n] in the case of .U suffix instructions and 9 or more is specified in the case of .D suffix instructions.**

Sample Program

● When this function is used in combination with other functions

Programmed Script

EM2000.T = **HASC** (SRA (DM1000.D, 8), 0)

Operation Description

The BIN data currently stored in DM1000.D is shifted to the right by 8 bits.
The value of the shift result (upper 24 bits before the shift) is converted to a HEX ASCII text string, and is stored in EM2000.T for the effective number of digits with zero suppress ON.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "SRA function", page 7-38
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

MEMO

ASC
RASC
DASC
STR
RDASC
HASC
RHASC
FASC
RFASC
VAL
LEN
SRGHT
RIGHT
SLEFT
LEFT
SMID
MID
SRPLC
REPLACE
SINS
INSERT
SDEL
DELETE
TRIM
LTRIM
RTRIM
STRIM
SPLIT
SFIND
SFINDN
INSTR
FIND
DISS
UNIS
CHR
CPSASC
RCPSASC

RHASC Convert HEX ASCII text string to BIN data

HEX ASCII code reverse conversion

Return value*1 = RHASC (conversion destination)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Conversion destination*2*3	Specifies the destination that stores the HEX ASCII text string to be converted or the HEX ASCII code text string.	—	—	—	—	—	—	—	.T	○	○	○
R	Return value*4	.D type RHASC function available.*5	.U	.S	.D	.L	.F	*6 .DF	—	—	—	○	○

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 T, C, CR, CTC, CTH, Z cannot be used.
*4 When bit data is specified, 32 continuous bits are processed. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 32 bits straddling the next channel.
*5 When the type other than .D is specified by return value, type conversion will be automatically conducted during conversion.
*6 The ".DF" suffix cannot be specified for device Z.

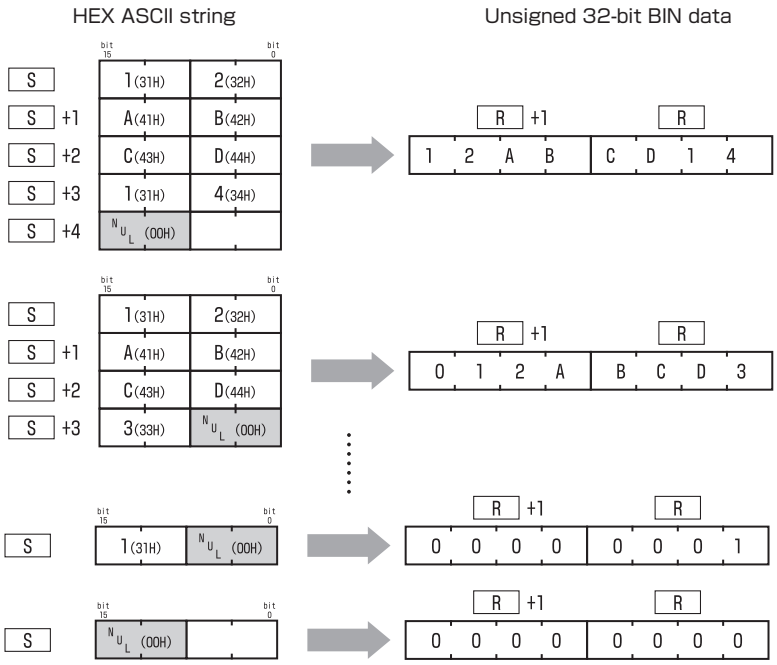
Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

The HEX ASCII code text string currently stored starting from S is converted to unsigned 32-bit BIN data, and the result is stored in R.

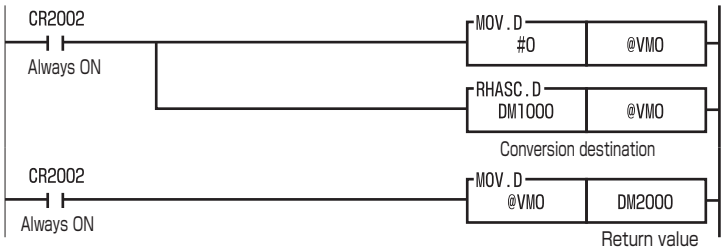


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● Example of format

DM2000.D = **RHASC** (DM1000.T)



Point

- **[S] A value in the range $30 to $39 ("0" to "9") or $41 to $46 ("A" to "F") must be stored in HEX ASCII text string specified by the conversion destination data.**
- **Use uppercase A to F (41H to 46H) in HEX ASCII text strings. Lowercase characters are not supported.**
- **When any of the following conditions is established, "0" is stored in [R].**
  - **When there is an invalid character in [S].**
  - **When the value after conversion is outside the unsigned 32-bit BIN data range**
  - **When a bit other than the start of the channel is specified when bit data is specified for [S].**
  - **When a timer/counter is specified to [S] by indirect specification.**

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000.D = **RHASC** (SLEFT (DM1000.T, 6))

Operation Description

The leading six characters of the HEX ASCII text string currently stored starting from DM1000.T are cut, and converted to unsigned 32-bit BIN data, and the result is stored in DM2000.D.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "SLEFT function", page 7-188
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

FASC Convert single precision floating point type real number to text string

Return value*1 = FASC (display format, total number of digits, number of digits past decimal point, conversion destination, conversion parameter)

Floating text string conversion

Argument/Return Value		Description	Supported Type								Const	Device/ Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Display specification format	Specifies the display specification format (decimal point format, exponential format).	.U	.U	.U	.U	—	—	—	.T	○	○	○
S2	Total number of digits	Specifies the total number of digits.	.U	.U	.U	.U	—	—	—	—	○	○	○
S3	Number of decimals	Specifies the number of decimals.	.U	.U	.U	.U	—	—	—	—	○	○	○
S4	Conversion destination*2*3*5	Specifies the destination that stores the single-precision floating point type real number to be converted or the single-precision floating point type real number data.	—	—	—	—	.F	—	—	—	○	○	○
S5	Conversion parameter	Specifies zero suppress and sign omission.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value*4*5	.T type FASC function available	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 When bit data is specified, it occupies 32 continuous bits. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, 32 bits are occupied straddling the next channel.
*3 Refer to "INTG function", page 7-122 for details on the range of the single precision floating point type real number.
*4 For bit data, specify the start of the channel.
*5 T, C, CR, CTC, CTH, Z cannot be used.

Description of Operation

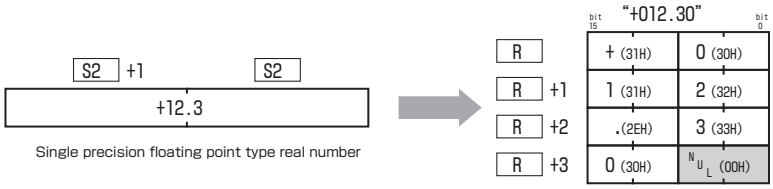
The single precision floating point type real number data stored in S4 is converted to a text string appended with the end code NUL (00H) according to the conversion parameters stored in S1 to S3 and S5, and the result is stored in R.

The conversion parameters are specified as follows:

S1	Display specification format	Specifies "0" when converting to a decimal point format text string, and "1" when converting to an exponent format text string.
S2	Total number of digits	Specify within the range 1 to 24 digit.
S3	Number of decimal	Specify within the range 0 to 7 digit.

The content to specify in the conversion parameter is as follows.

S5 Value	Details	CR2814	CR2815
0	Zero suppress OFF, sign omission OFF	0	0
1	Zero suppress ON, sign omission OFF	1	0
2	Zero suppress OFF, sign omission ON	0	1
3	Zero suppress ON, sign omission ON	1	1

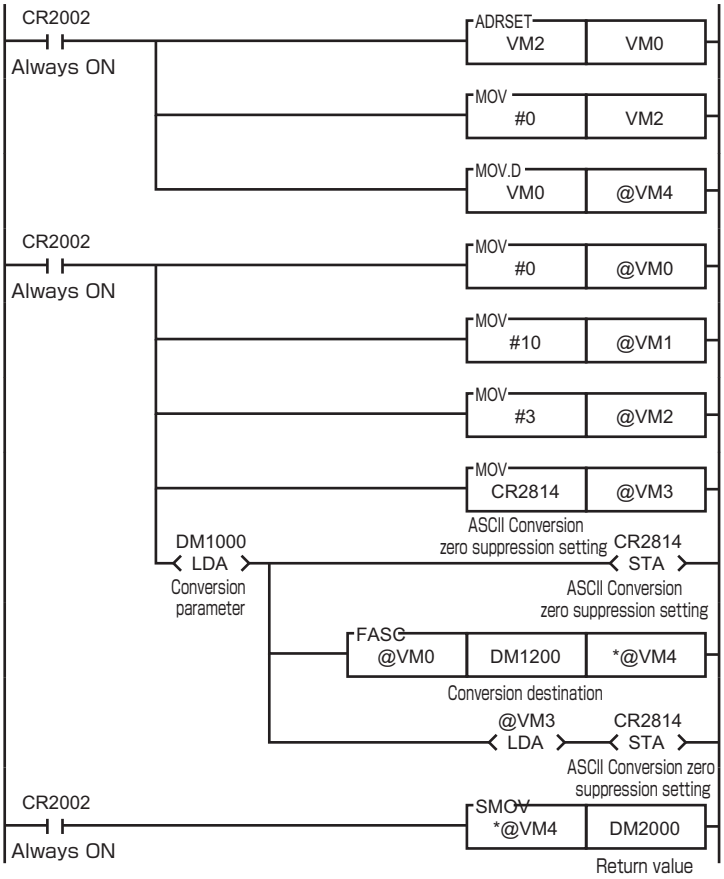


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● Example of format

DM2000.T = **FASC** (0, 10, 3, DM1200.F, DM1000)



ASC
RASC
DASC
STR
RDASC
HASC
RHASC
FASC
RFASC
VAL
LEN
SRGHT
RIGHT
SLEFT
LEFT
SMID
MID
SRPLC
REPLACE
SINS
INSERT
SDEL
DELETE
TRIM
LTRIM
RTRIM
STRIM
SPLIT
SFIND
SFINDN
INSTR
FIND
DISS
UNIS
CHR
CPSASC
RCPSASC

Point

- **When a constant is used for the conversion destination data, it cannot be specified in HEX ($).**
- **Attention is required when changing or looking up CR2814/CR2815 by interrupt programs.**
- **When any of the following conditions is established, the end code (NUL(00H)) is stored in [R].**
  - **When the argument value is out of range**
  - **When a single precision floating point type real number out of the range has been specified**
  - **When the number of digits of the text string to store has exceeded the total number of digits**
  - **When a bit other than the start of the channel is specified when bit data is specified for [R].**
  - **When a timer/counter is specified to [R] by indirect specification.**
  - **When the total number of digits and the number of digits past the decimal point do not satisfy the following conditions.**

	Decimal point format display	Exponent format display
[S3]=0	1≦[S2]≦24	5≦[S2]≦24
0<[S3]≦7	[S3]+2≦[S2] 3≦[S2]≦24	[S3]+6≦[S2] 7≦[S2]≦24

Sample Program

● When this function is used in combination with other functions

Programmed Script

EM2000.T = **FASC** (0, 10, 3, SIN (DM1000.F), 3)

Operation Description

The sine (SIN) value of the angle of the single precision floating point type real number stored in DM1000.F is calculated, and the result is converted to a decimal point format text string that takes three of the total of ten digits as the number of digits past the decimal point. (conversion with zero suppress and sign omission)
The converted text string is appended with the end code NUL (00H), and is stored in EM2000.T. (The conversion destination need not be converted to a single precision floating point type by using the FLOAT function, etc.)

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "SIN function", page 7-140
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

MEMO

ASC
RASC
DASC
STR
RDASC
HASC
RHASC
FASC
RFASC
VAL
LEN
SRGHT
RIGHT
SLEFT
LEFT
SMID
MID
SRPLC
REPLACE
SINS
INSERT
SDEL
DELETE
TRIM
LTRIM
RTRIM
STRIM
SPLIT
SFIND
SFINDN
INSTR
FIND
DISS
UNIS
CHR
CPSASC
RCPSASC

RFASC Text string to single precision floating point type real number conversion

Single precision floating point text string reverse conversion

Return value*1 = RFASC (conversion destination)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Conversion destination*2*4*5	Specifies the destination that stores the single precision floating point text string to be converted or the single precision floating point type text string.	—	—	—	—	—	—	—	.T	○	○	○
R	Return value*3*5	.F type RFASC function available.* 6	.U	.S	.D	.L	.F	*7 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 When bit data is specified, it occupies 32 continuous bits. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, 32 bits are occupied straddling the next channel.
*4 A NUL (00H) is required at the end of the single precision floating point type text string that is specified by the conversion destination.
*5 T, C, CR, CTC, CTH, Z cannot be used.
*6 When the type other than .F is specified by return value, type conversion will be automatically conducted during conversion.
*7 The ".DF" suffix cannot be specified for device Z.

Point

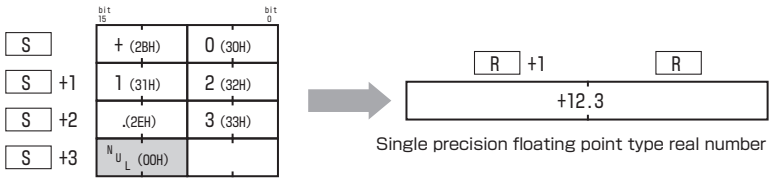
- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

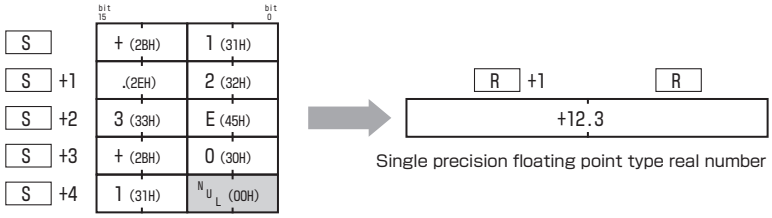
Description of Operation

The single precision floating point text string currently stored starting from S is converted to single precision floating point type real number data, and the result is stored in R.

when single precision floating point text string "+012.30"



when single precision floating point text string "+1.23E+01"

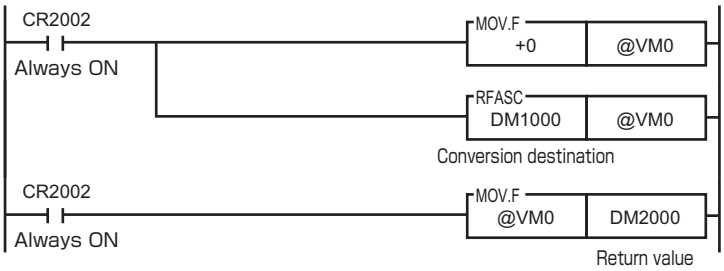


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● Example of format

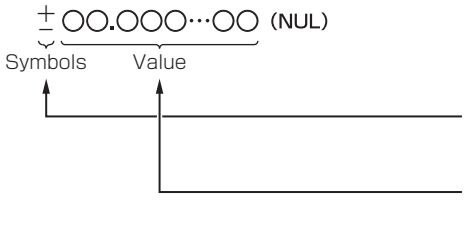
DM2000.F = **RFASC** (DM1000.T)

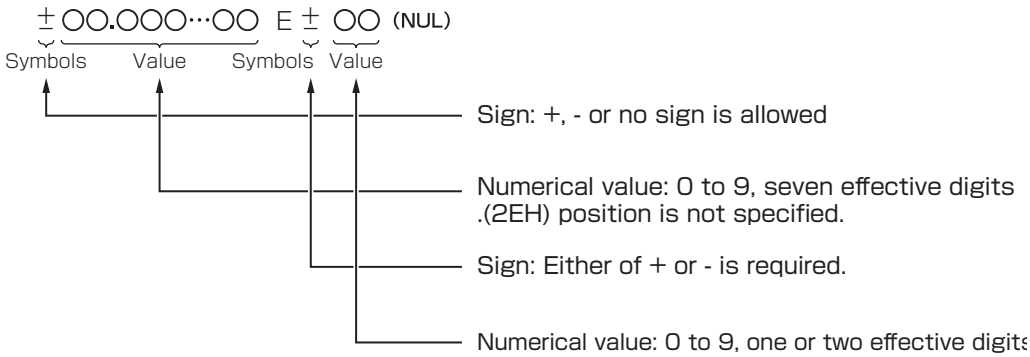


Point

- **Only the values of $30 to $39 ("0" to "9"), $2B ("+"), $2C ("-"), $45 ("E"), and $2E (".") must be currently stored in single precision floating point text string to be specified by the conversion destination data.**
- **When any of the following conditions is established, "0" is stored in [R].**
  - **When the length of a text string is 0**
  - **When the format of the text string is incorrect**
  - **When the numerical value after conversion exceeds the single precision floating point type real number range**
  - **When the number of digits of the text string to store has exceeded the total number of digits**
  - **When a bit other than the start of the channel is specified when bit data is specified for [S].**

● Format of text string





Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000.F = COS (**RFASC** (DM1000.T))

Operation Description

The single precision floating point text string of the angle (radian units) currently stored starting from DM1000.T is converted to single precision floating point type real number data, and the cosine (COS) value is calculated from the conversion result and stored in DM2000.F.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "COS function", page 7-142
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

VAL

Text string to single precision floating point type real number conversion

Single precision floating point text string reverse conversion

Return value*1 = VAL (conversion destination)

Argument/Return Value		Description	Supported Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Conversion destination*2*3*5	Specifies the destination that stores the single precision floating point text string to be converted or the single precision floating point type text string.	—	—	—	—	—	—	—	.T	○	○	○
R	Return value*4	.F type VAL function available.*6	.U	.S	.D	.L	.F	*7 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 A NUL (00H) is required at the end of the single precision floating point type text string that is specified by the conversion destination.
*4 When bit data is specified, it occupies 32 continuous bits. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, 32 bits are occupied straddling the next channel.
*5 T, C, CR, CTC, CTH, Z cannot be used.
*6 When the type other than .F is specified by return value, type conversion will be automatically conducted during conversion.
*7 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

This function differs from the "RFASC function" in name only, and operation, etc. are the same.
For details on operation, refer to "RFASC function", page 7-178.

LEN Detect text string length

Return value*1 = LEN (Destination text string)

Detect text string length

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Destination text string*3*4	Specifies the destination to store the text string for calculating the text string length, or the destination text string.	—	—	—	—	—	—	—	.T	*2 ○	○	○
R	Return value	.U type LEN function available.*5	.U	.S	.D	.L	.F	*6 .DF	—	—	—	○	—

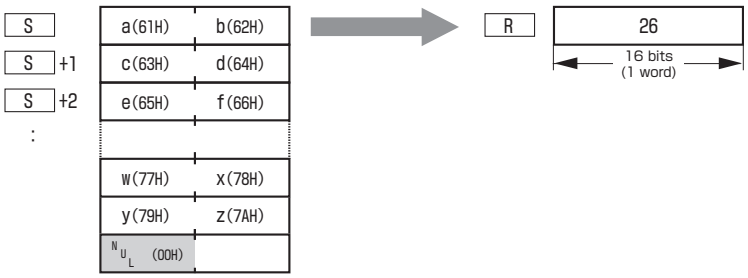
*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 To specify a text string using constants, enclose with " ".
*3 For bit data, specify the start of the channel.
*4 T, C, CR, CTC, CTH, Z cannot be used.
*5 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.
*6 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

The number of characters of the text string currently stored starting from S is counted, and the result is stored in R as 16-bit BIN data.

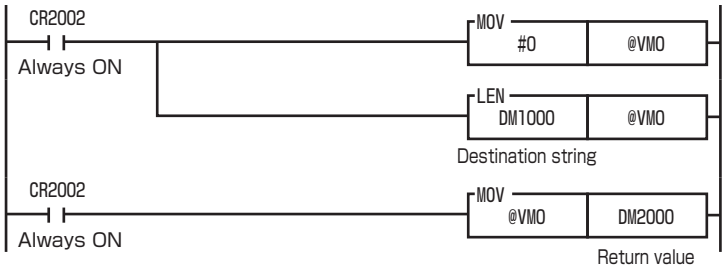


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

DM2000 = LEN (DM1000.T)



Point

- **2-byte characters are handled as two characters.**
- **End codes (NUL (00H)) are not counted.**
- **When any of the following conditions is established, "0" is stored in [R].**
  - **When there is no end code (NUL (00H))**
  - **When the text string including the end code (NUL (00H)) exceeds 2000 characters**
  - **When a bit other than the start of the channel is specified when bit data is specified for [S] by indirect specification.**
  - **When a timer/counter is specified to [S] by indirect specification.**

Sample Program

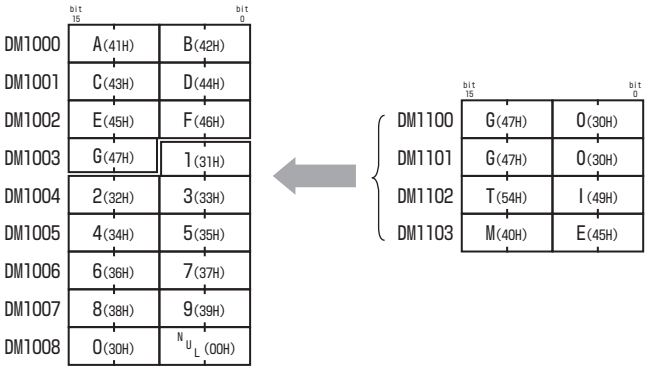
● **When this function is used in combination with other functions**

Programmed Script

DM2000.T = SINS (DM1000.T, DM1100.T, **LEN** (DM1000.T) **-** 10)

Operation Description

The length (number of characters) of the text string currently stored starting from DM1000 is calculated, and the text string currently stored starting from DM1100 is inserted ten characters in front (LEN (DM1000.T) -10).
The text string resulting from the insertion is appended with the end code NUL (00H) and is stored in DM2000.



Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "SINS function", page 7-200
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

SRGHT Cut text string on right

Return value*1 = SRGHT (cut destination, number of cut characters)

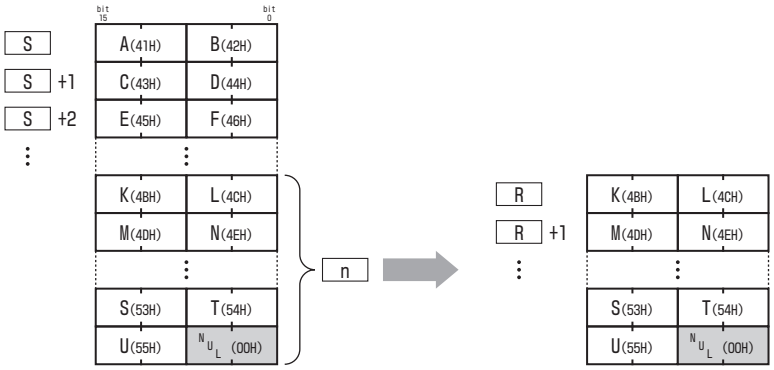
Cut text string on right

Argument/Return Value	Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S Cut destination*2*6	Specifies the destination to store the text string to be cut, or the cut destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
n Number of cut characters*5	Specifies the number of characters (number of bytes) to cut. (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
R Return value*2 *6	.T type SRGHT function available	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 $ cannot be used.
*5 Specify the number of characters (number of bytes) from the right end to the number of cut characters within the range 0 to 1999.
When "0" has been specified, one character's worth of NUL (00H) is stored in return value.
*6 T, C, CR, CTC, CTH, Z cannot be used.

Description of Operation

The text string currently stored starting from S is cut from the right end by the number of characters (number of bytes) specified by n, appended with the end code NUL (00H), and the result is stored in R.



Not including NUL(00H) in the end of string.

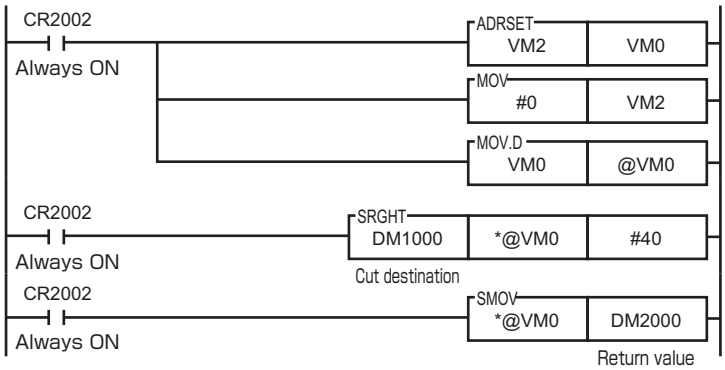
Save NUL (00H) in the end of string after cutting.

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

DM2000.T = SRGHT (DM1000.T, 40)



Point

- **When any of the following conditions is established, the end code (NUL (00H)) is stored.**
  - **When the text string specified by S does not contain the end code (NUL(00H))**
  - **When the text string including the end code (NUL (00H)) exceeds 2000 characters**
  - **When the text string to be stored in R onwards exceeds the device / variable range**
  - **When a bit other than the start of the channel is specified when bit data is specified for S and R.**
  - **When a timer/counter is specified to S and R by indirect specification.**

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

EM2000.L = RDASC (**SRGHT** (DM1000.T, 8))

Operation Description

The final eight characters (lower eight digits) of the DEC ASCII text string currently stored starting from DM1000 are cut, and converted to signed 32-bit BIN data, and the result is stored in EM2000.L.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "RDASC function", page 7-166
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

RIGHT Cut text string on right

Return value*1 = RIGHT (cut destination, number of cut characters)

Cut text string on right

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Cut destination*2*6	Specifies the destination to store the text string to be cut, or the cut destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
n	Number of cut characters*5	Specifies the number of characters (number of bytes) to cut. (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
R	Return value*2 *6	.T type RIGHT function available	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.

*2 For bit data, specify the start of the channel.

*3 To specify a text string using constants, enclose with " ".

*4 $ cannot be used.

*5 Specify the number of characters (number of bytes) from the right end to the number of cut characters within the range 0 to 1999.
When "0" has been specified, one character's worth of NUL (00H) is stored in return value.

*6 T, C, CR, CTC, CTH, Z cannot be used.

Description of Operation

This function differs from the "SRGHT function" in name only, and operation, etc. are the same.
For details on operation, refer to "SRGHT function", page 7-184.

MEMO

ASC
RASC
DASC
STR
RDASC
HASC
RHASC
FASC
RFASC
VAL
LEN
SRGHT
RIGHT
SLEFT
LEFT
SMID
MID
SRPLC
REPLACE
SINS
INSERT
SDEL
DELETE
TRIM
LTRIM
RTRIM
STRIM
SPLIT
SFIND
SFINDN
INSTR
FIND
DISS
UNIS
CHR
CPSASC
RCPSASC

SLEFT Cut text string on left

Return value*1 = SLEFT (cut destination, number of cut characters)

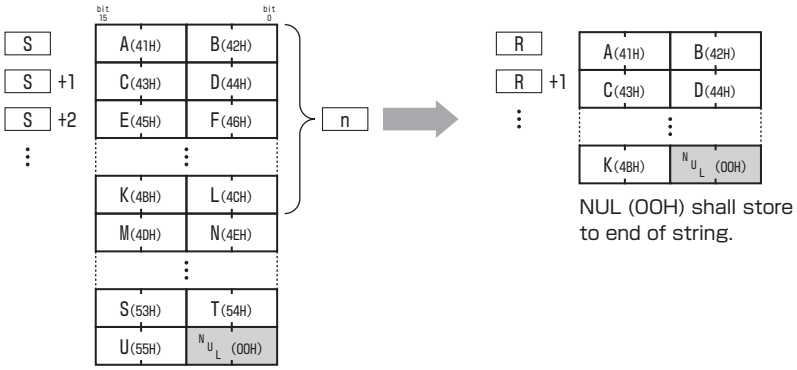
Cut text string on left

Argument/Return Value	Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S Cut destination*2*6	Specifies the destination to store the text string to be cut, or the cut destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
n Number of cut characters*5	Specifies the number of characters (number of bytes) to cut. (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
R Return value*2 *6	.T type SLEFT function available	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 $ cannot be used.
*5 Specify the number of characters (number of bytes) from the left end to the number of cut characters within the range 0 to 1999.
When "0" has been specified, one character's worth of NUL (00H) is stored in return value.
*6 T, C, CR, CTC, CTH, Z cannot be used.

Description of Operation

The text string currently stored starting from S is cut from the left end (start) by the number of characters (number of bytes) specified by n, appended with the end code NUL (00H), and the result is stored in R.

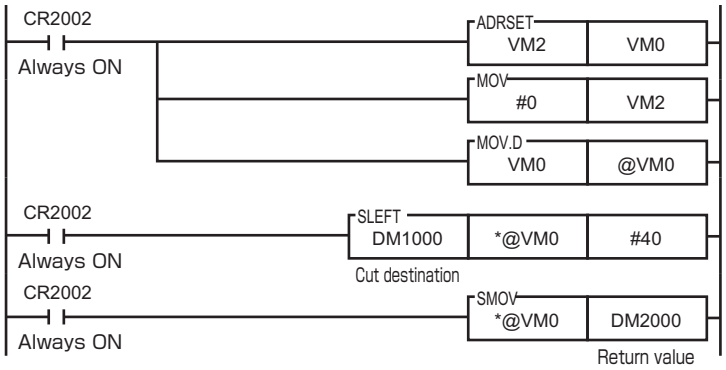


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

DM2000.T = **SLEFT** (DM1000.T, 40)



Point

- **When any of the following conditions is established, the end code (NUL (00H)) is stored.**
  - **When the text string specified by [S] does not contain the end code (NUL (00H))**
  - **When the text string including the end code (NUL (00H)) exceeds 2000 characters**
  - **When the text string to be stored in [R] onwards exceeds the device / variable range**
  - **When a bit other than the start of the channel is specified when bit data is specified for [S] and [R].**
  - **When a timer/counter is specified to [S] and [R] by indirect specification.**

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

EM2000.D = RHASC (**SLEFT** (DM1000.T, 4))

Operation Description

The leading four characters (upper four digits) of the HEX ASCII text string currently stored starting from DM1000.T are cut, and converted to unsigned 16-bit BIN data, and the result is stored in EM2000.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "RHASC function", page 7-172
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

LEFT Cut text string on left

Return value*1 = LEFT (cut destination, number of cut characters)

Cut text string on left

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Cut destination*2*6	Specifies the destination to store the text string to be cut, or the cut destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
n	Number of cut characters*5	Specifies the number of characters (number of bytes) to cut. (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
R	Return value*2 *6	.T type LEFT function available	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 $ cannot be used.
*5 Specify the number of characters (number of bytes) from the left end to the number of cut characters within the range 0 to 1999.
When "0" has been specified, one character's worth of NUL (00H) is stored in return value.
*6 T, C, CR, CTC, CTH, Z cannot be used.

Description of Operation

This function differs from the "SLEFT function" in name only, and operation, etc. are the same.
For details on operation, refer to "SLEFT function", page 7-188.

MEMO

SMID Cut text string

Return value*1 = SMID (cut destination, cut character position, number of cut characters)

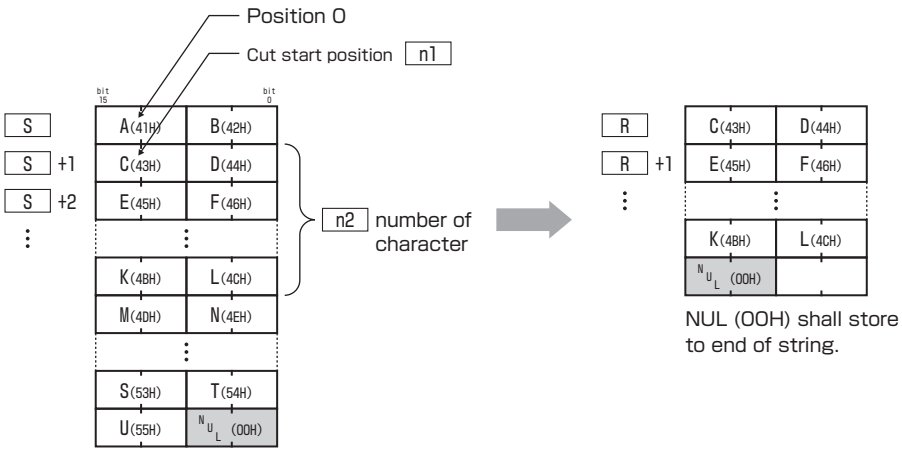
Cut text string

Argument/Return Value	Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S Cut destination*2*6	Specifies the destination to store the text string to be cut, or the cut destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
n1 Cut character position*7	Specifies the cut leading position (number of bytes). (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
n2 Number of cut characters*5	Specifies the number of characters (number of bytes) to cut. (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
R Return value*2*6	.T type SMID function available	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 $ cannot be used.
*5 Specify the number of characters (number of bytes) from the cut character to the number of cut characters within the range of 0 to 1999.
When "0" has been specified, one character's worth of NUL (00H) is stored in return value.
*6 T, C, CR, CTC, CTH, Z cannot be used.
*7 The leading character of the text string specified by S is taken to be "0".

Description of Operation

The text string currently stored starting from S is cut from the position (number of bytes) specified by n1 by n2 characters (number of bytes), appended with the end code NUL (00H), and the result is stored in R.

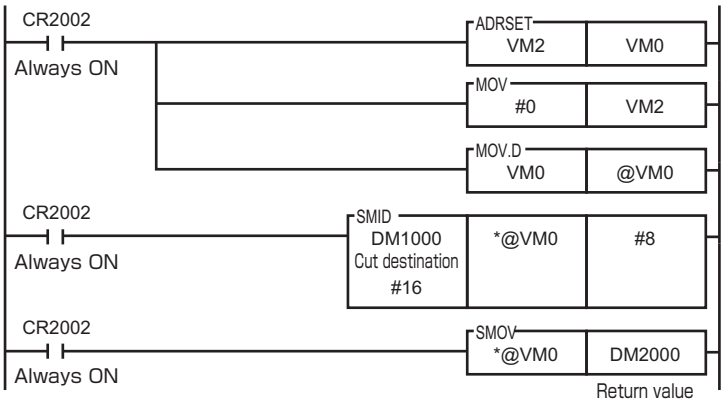


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

DM2000.T = **SMID** (DM1000.T, 8, 16)



Point

- **When any of the following conditions is established, the end code (NUL (00H)) is stored.**
  - **When the text string specified by [S] does not contain the end code (NUL (00H))**
  - **When the text string including the end code (NUL (00H)) exceeds 2000 characters**
  - **When [S] text string length < [n1].**
  - **When the text string to be stored in [R] onwards exceeds the device / variable range**
  - **When a bit other than the start of the channel is specified when bit data is specified for [S] and [R].**
  - **When a timer/counter is specified to [S] and [R] by indirect specification.**

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

EM2000.D = RHASC (**SMID** (DM1000.T, 2, 4))

Operation Description

The four characters (middle four digits) from the 2nd character of the HEX ASCII text string currently stored starting from DM1000.T are cut, and converted to unsigned 16-bit BIN data, and the result is stored in EM2000.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "RHASC function", page 7-172
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

MID Cut text string

Return value*1 = MID (cut destination, cut character position, number of cut characters)

Cut text string

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S	Cut destination*2*6	Specifies the destination to store the text string to be cut, or the cut destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
n1	Cut character position*7	Specifies the cut leading position (number of bytes). (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
n2	Number of cut characters*5	Specifies the number of characters (number of bytes) to cut. (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
R	Return value*2*6	.T type MID function available	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 $ cannot be used.
*5 Specify the number of characters (number of bytes) from the cut character to the number of cut characters within the range of 0 to 1999.
When "0" has been specified, one character's worth of NUL (00H) is stored in return value.
*6 T, C, CR, CTC, CTH, Z cannot be used.
*7 The leading character of the text string specified by S is taken to be "0".

Description of Operation

This function differs from the "SMID function" in name only, and operation, etc. are the same.
For details on operation, refer to "SMID function", page 7-192.

MEMO

ASC
RASC
DASC
STR
RDASC
HASC
RHASC
FASC
RFASC
VAL
LEN
SRGHT
RIGHT
SLEFT
LEFT
SMID
MID
SRPLC
REPLACE
SINS
INSERT
SDEL
DELETE
TRIM
LTRIM
RTRIM
STRIM
SPLIT
SFIND
SFINDN
INSTR
FIND
DISS
UNIS
CHR
CPSASC
RCPSASC

SRPLC Replace text string section

Return value*1 = SRPLC (replace destination, replace text string destination, replace leading postion, number of replaced characters)

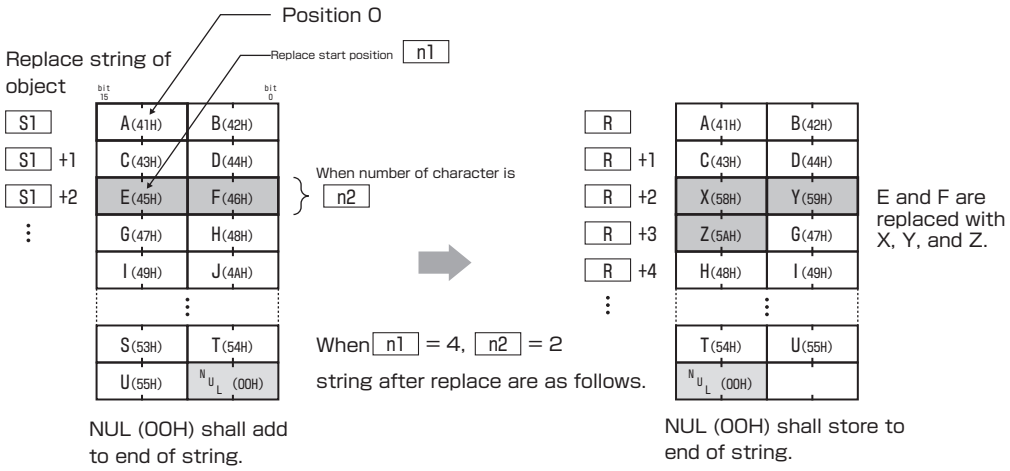
Replace text string section

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Replace destination*2*6	Specifies the destination to store the text string to be replaced, or the replace destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
S2	Replace text string destination*2*6	Specifies the destination to store the text string to replace, or the replace text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
n1	Replace leading position*7	Specifies the replace leading position (number of bytes). (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
n2	Number of replaced characters*5	Specifies the number of characters (number of bytes) to replace. (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
R	Return value*2*6	Stores the value of the function processed result.	—	—	—	—	—	—	—	.T	—	○	—

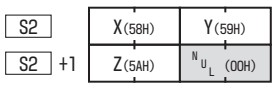
*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 $ cannot be used.
*5 Specify the number of characters (number of bytes) from the replace leading position to the number of replace characters within the range of 0 to 1999.
*6 T, C, CR, CTC, CTH, Z cannot be used.
*7 The leading character of the text string specified by S1 is taken to be "0".

Description of Operation

The text string currently stored starting from S1 is replaced with the text string currently stored starting from S2 from the position (number of bytes) specified by n1 by the text string with the number of characters (number of bytes) specified by n2, appended with the end code NUL (00H), and the result is stored in R.



String shall replace

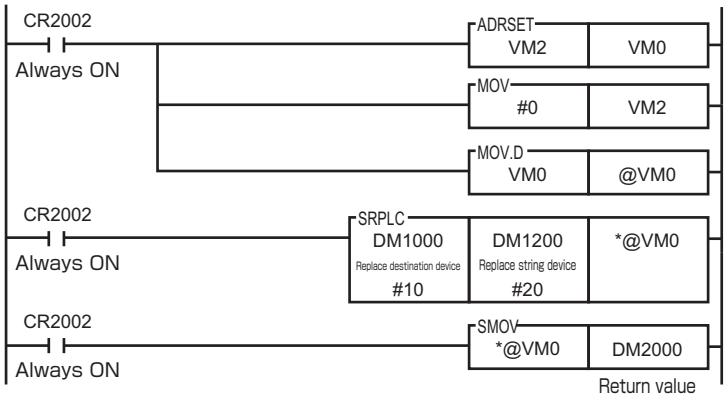


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● Example of format

DM2000.T = **SRPLC** (DM1000.T, DM1200.T, 10, 20)



Point

- **When any of the following conditions is established, the end code (NUL (00H)) is stored.**
  - **When the text string specified by [S1] and [S2] does not contain the end code (NUL (00H))**
  - **When the text string including the end code (NUL (00H)) exceeds 2000 characters**
  - **When [S1] text string length < [n1].**
  - **When the length of the text string after the replace exceeds 2000 characters including (NUL (00H))**
  - **When a bit other than the start of the channel is specified when bit data is specified for [S1], [S2] and [R].**
  - **When a timer/counter is specified to [S1], [S2] and [R] by indirect specification.**

Sample Program

● When this function is used in combination with other functions

Programmed Script

EM2000.L = RDASC (**SRPLC** (DM1000.T, EM1000.T, 0, 4))

Operation Description

The text string four characters long (number of bytes) from the start of the decimal ASCII text string currently stored from the start of DM1000.T is replaced with the decimal ASCII text string currently stored starting from EM1000.T.
The decimal ASCII text string resulting from the replace is converted to signed 32-bit BIN data, and the result is stored in EM2000.L.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "RDASC function", page 7-166
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

Hint

Operation of the SRPLC function changes as follows according to the content specified to the argument:

Example 1)Operation is the same as the SINS function when "0" is specified to the number of replace characters.

The replace text string is inserted (no replace destination range) to the 4th character of the replace destination text string.

Programmed Script

DM2000.T = **SRPLC** (DM1000.T, DM1200.T, 4, 0)

Example 2)Operation is the same as the SDEL function when an empty text string (NUL only) is specified to the replace text string.

The four characters from the 2nd character of the replace destination text string are deleted (no replace text string).

Programmed Script

DM2000.T = **SRPLC** (DM1000.T, "", 2, 4)

REPLACE Replace text string section

Return value*1 = REPLACE (replace destination, replace text string destination, number of replaced characters, replace leading position)

Replace text string section

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S1	Replace destination*2*6	Specifies the destination to store the text string to be replaced, or the replace destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
S2	Replace text string destination*2*6	Specifies the destination to store the text string to replace, or the replace text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
n1	Number of replaced characters*5	Specifies the number of characters (number of bytes) to replace. (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
n2	Replace leading position*7	Specifies the replace leading position (number of bytes). (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
R	Return value*2*6	Stores the value of the function processed result.	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 $ cannot be used.
*5 Specify the number of characters (number of bytes) from the replace leading position to the number of replace characters within the range of 0 to 1999.
*6 T, C, CR, CTC, CTH, Z cannot be used.
*7 The leading character of the text string specified by S1 is taken to be "0".

Description of Operation

This function differs from the "SRPLC function" in name only, and operation, etc. are the same.
For details on operation, refer to "SRPLC function", page 7-196.

ASC
RASC
DASC
STR
RDASC
HASC
RHASC
FASC
RFASC
VAL
LEN
SRGHT
RIGHT
SLEFT
LEFT
SMID
MID
SRPLC
REPLACE
SINS
INSERT
SDEL
DELETE
TRIM
LTRIM
RTRIM
STRIM
SPLIT
SFIND
SFINDN
INSTR
FIND
DISS
UNIS
CHR
CPSASC
RCPSASC

SINS Insert text string

Return value*1 = SINS (insert destination, insert text string destination, insert leading position)

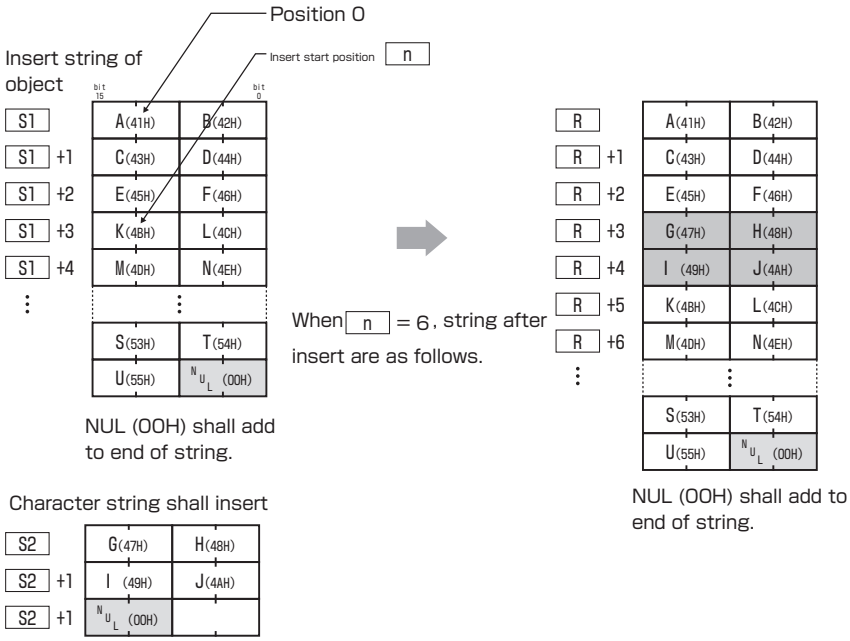
Insert text string

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Insert destination*2*6	Specifies the destination to store the text string to be inserted, or the insert destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
S2	Insert text string destination*2*6	Specifies the destination to store the text string to insert, or the insert text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
n	Insert leading position*7	Specifies the insert leading position (number of bytes). (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
R	Return value*4*6	.T type SINS function available	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 $ cannot be used.
*5 Specify the number of characters (number of bytes) from the left end (start) to the number of insert characters within the range 0 to 1999.
*6 T, C, CR, CTC, CTH, Z cannot be used.
*7 The leading character of the text string specified by S1 is taken to be "0".

Description of Operation

The text string currently stored starting from S2 is inserted to the position (number of bytes) specified by n of the text string currently stored starting from S1, appended with the end code NUL (00H), and the result is stored in R.

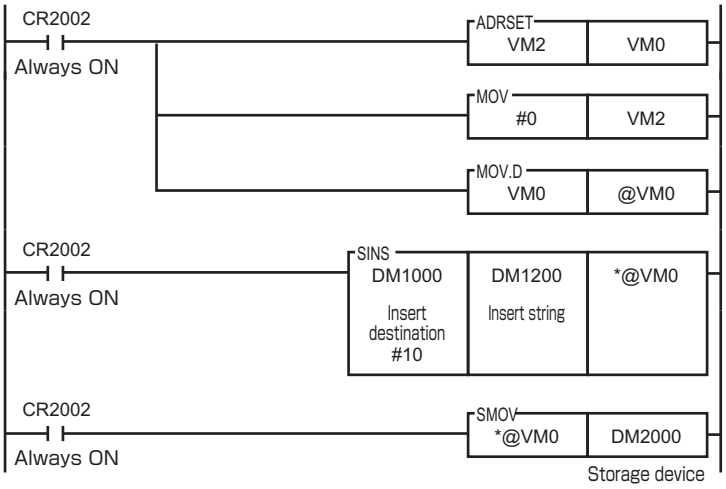


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● Example of format

DM2000.T = **SINS** (DM1000.T, DM1200.T, 10)



Point

- **When any of the following conditions is established, the end code (NUL (00H)) is stored.**
  - **When the text string specified by [S1] and [S2] does not contain the end code (NUL (00H))**
  - **When the text string including the end code (NUL (00H)) exceeds 2000 characters**
  - **When [S1] text string length < [n].**
  - **When the length of the text string after the insert exceeds 2000 characters including (NUL (00H))**
  - **When a bit other than the start of the channel is specified when bit data is specified for [S1], [S2] and [R].**
  - **When a timer/counter is specified to [S1], [S2] and [R] by indirect specification.**

Sample Program

● When this function is used in combination with other functions

Programmed Script

EM2000.D = RHASC (**SINS** (DM1000.T, EM1000.T, 2))

Operation Description

The HEX ASCII text string currently stored starting from EM1000.Tis inserted at the 2nd character (number of bytes) from the start of the HEX ASCII text string currently stored in DM1000.T.
The HEX ASCII text string resulting from the insert is converted to unsigned 32-bit BIN data, and the result is stored in EM2000.D.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "RDASC function", page 7-166
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

INSERT Insert text string

Return value*1 = INSERT (insert destination, insert text string destination, insert leading position)

Insert text string

Argument/Return Value		Description	Supported Type								Const	Device/ Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Insert destination*2*6	Specifies the destination to store the text string to be inserted, or the insert destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
S2	Insert text string destination*2*6	Specifies the destination to store the text string to insert, or the insert text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
n	Insert leading position*7	Specifies the insert leading position (number of bytes). (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
R	Return value*2*6	.T type INSERT function available	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 $ cannot be used.
*5 Specify the number of characters (number of bytes) from the left end (start) to the number of insert characters within the range 0 to 1999.
*6 T, C, CR, CTC, CTH, Z cannot be used.
*7 The leading character of the text string specified by S1 is taken to be "0".

Description of Operation

This function differs from the "SINS function" in name only, and operation, etc. are the same.
For details on operation, refer to "SINS function", page 7-200.

MEMO

ASC
RASC
DASC
STR
RDASC
HASC
RHASC
FASC
RFASC
VAL
LEN
SRGHT
RIGHT
SLEFT
LEFT
SMID
MID
SRPLC
REPLACE
SINS
INSERT
SDEL
DELETE
TRIM
LTRIM
RTRIM
STRIM
SPLIT
SFIND
SFINDN
INSTR
FIND
DISS
UNIS
CHR
CPSASC
RCPSASC

SDEL Delete text string section

Return value*1 = SDEL (delete destination, delete leading position, number of delete characters)

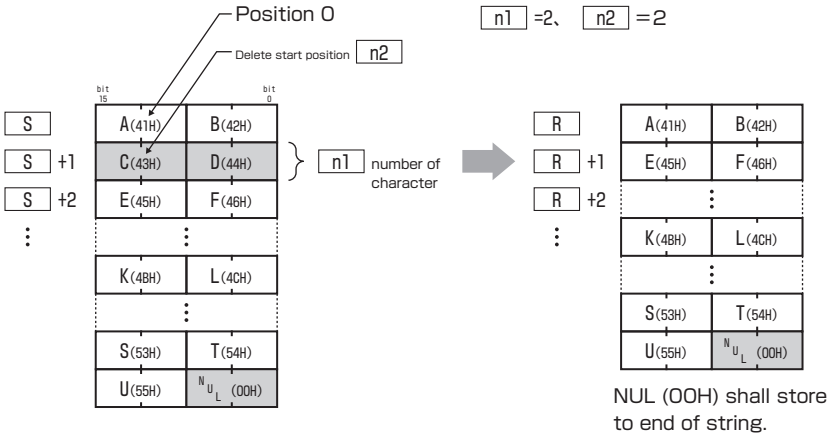
Delete text string section

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Delete destination*2*7	Specifies the destination to store the text string to be deleted, or the delete destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
n1	Delete leading position*4*5	Specifies the delete leading position (number of bytes). (0 to 1999)	.U	.U	.U	.U	—	—	—	—	○	○	○
n2	Number of delete characters*4*6	Specifies the number of characters (number of bytes) to delete. (0 to 1999)	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value*2*7	.T type SDEL function available	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 $ cannot be used.
*5 Specify the number of characters (number of bytes) from the left end (start) to the number of deleted characters within the range 0 to 1999.
*6 Specify the number of characters (number of bytes) from the delete leading position to the number of deleted characters within the range 0 to 1999.
*7 T, C, CR, CTC, CTH, Z cannot be used.

Description of Operation

The text string currently stored starting from S is deleted from the position (number of bytes) specified by n1 by the text string with number of characters (number of bytes) specified by n2, appended with the end code NUL (00H), and the result is stored in R.

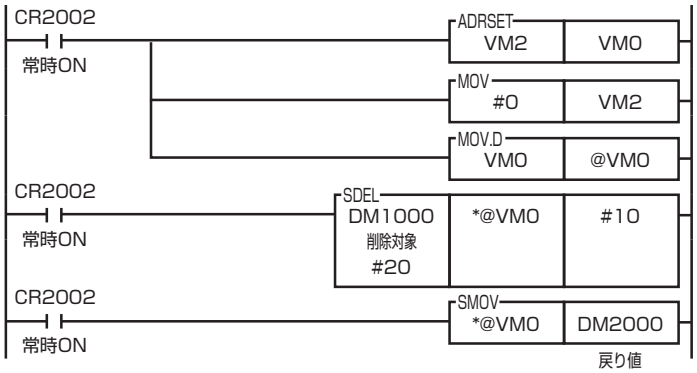


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

DM2000.T = **SDEL** (DM1000.T, 10, 20)



Point

- **When any of the following conditions is established, the end code (NUL (00H)) is stored.**
  - **When the text string specified by [S] does not contain the end code (NUL (00H))**
  - **When the text string including the end code (NUL (00H)) exceeds 2000 characters**
  - **When [S] text string length < [n1].**
  - **When the text string to be stored in [R] onwards exceeds the device / variable range**
  - **When a bit other than the start of the channel is specified when bit data is specified for [S] and [R].**
  - **When a timer/counter is specified to [S] and [R] by indirect specification.**

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

EM2000.D = RHASC (**SDEL** (DM1000.T, 2, 4))

Operation Description

A text string of four characters (number of bytes) is deleted from the 2nd character (number of bytes) of the HEX ASCII text string currently stored starting from DM1000.T. The HEX ASCII text string resulting from the delete is converted to unsigned 16-bit BIN data, and the result is stored in EM2000.D.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "RHASC function", page 7-172
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

DELETE Delete text string section

Return value*1 = DELETE (delete destination, number of delete characters, delete leading position)

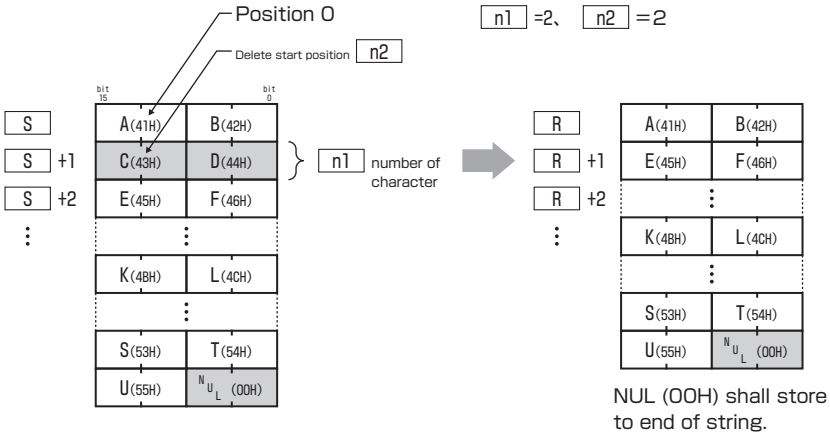
Delete text string section

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Delete destination*2*7	Specifies the destination to store the text string to be deleted, or the delete destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
n1	Number of delete characters*4*6	Specifies the number of characters (number of bytes) to delete. (0 to 1999)	.U	.U	.U	.U	—	—	—	—	○	○	○
n2	Delete leading position*4*5	Specifies the delete leading position (number of bytes). (0 to 1999)	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value*2*7	.T type DELETE function available	—	—	—	—	—	—	—	.T	○	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 $ cannot be used.
*5 Specify the number of characters (number of bytes) from the left end (start) to the number of deleted characters within the range 0 to 1999.
*6 Specify the number of characters (number of bytes) from the delete leading position to the number of deleted characters within the range 0 to 1999.
*7 T, C, CTC, CTH, Z cannot be used.

Description of Operation

The text string currently stored starting from S is deleted from the position (number of bytes) specified by n2 by the text string with number of characters (number of bytes) specified by n1, appended with the end code NUL (00H), and the result is stored in R.

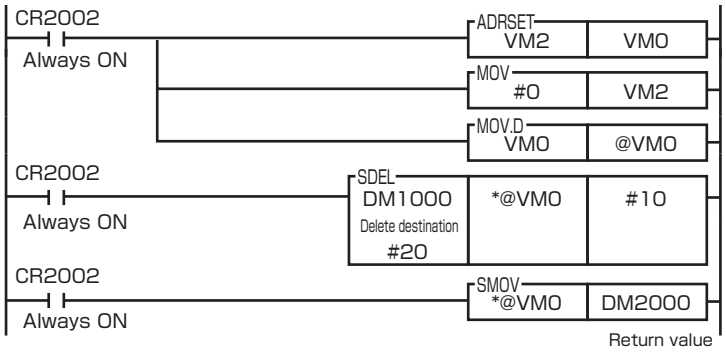


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

DM2000.T = **DELETE** (DM1000.T, 20, 10)



Point

- **When any of the following conditions is established, the end code (NUL (00H)) is stored.**
  - **When the text string specified by [S] does not contain the end code (NUL (00H))**
  - **When the text string including the end code (NUL (00H)) exceeds 2000 characters**
  - **When [S] text string length < [n2].**
  - **When the text string to be stored in [R] onwards exceeds the device / variable range**
  - **When a bit other than the start of the channel is specified when bit data is specified for [S] and [R].**
  - **When a timer/counter is specified to [S] and [R] by indirect specification.**

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

EM2000.D = RHASC (**DELETE** (DM1000.T, 4, 2))

Operation Description

A text string of four characters (number of bytes) is deleted from the 2nd character (number of bytes) of the HEX ASCII text string currently stored starting from DM1000.T. The HEX ASCII text string resulting from the delete is converted to unsigned 16-bit BIN data, and the result is stored in EM2000.D.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "RHASC function", page 7-172
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

TRIM

Trim 1-byte space and tab at ends of text string

Return value*1 = TRIM (Destination text string)

Trim

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Destination text string*2	Specifies the destination to store the text string to be trimmed, or the trim destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
R	Return value*2*4	Stores the trimmed text string.	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.

*2 For bit data, specify the start of the channel.

*3 To specify a text string using constants, enclose with " ".

*4 T, C, CTC, CTH, Z cannot be used.

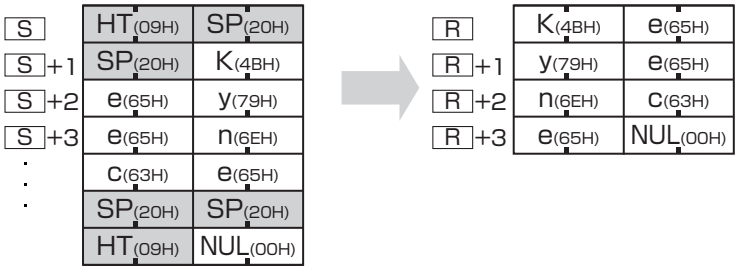
Point

- **The TRIM function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

All 1-byte spaces (20H) and labels (09H) at both ends (start and end) of the text string stored starting from S are deleted, the end code NUL (00H) is appended, and the result is stored in R.

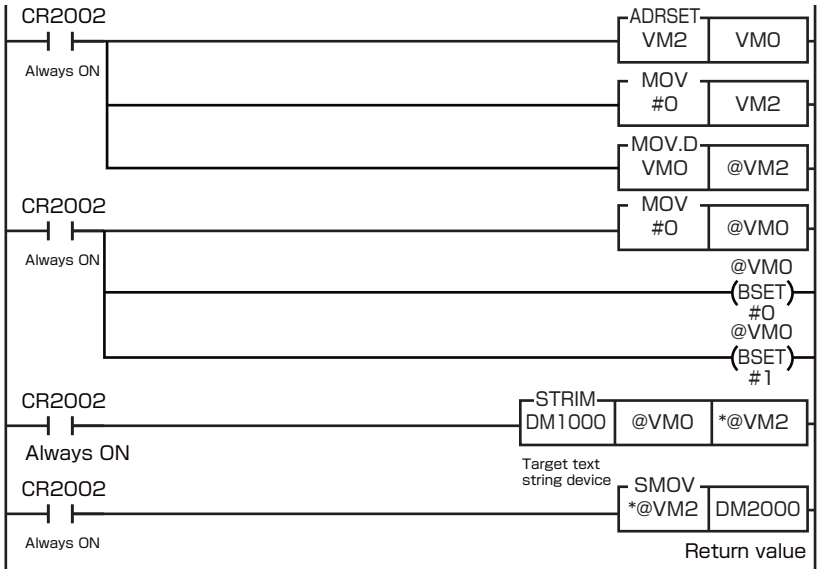


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

DM2000.T = **TRIM** (DM1000.T)



Sample Program

● **When this function is used in combination with other functions**

Programmed Script

EM2500.D = RHASC (**TRIM** (DM1000.T))

Operation Description

The 1-byte spaces and tabs are removed from both ends (leading and end) of a hexadecimal ASCII text string stored starting from DM1000.T.
The HEX ASCII text string resulting from the delete is converted to unsigned 16-bit BIN data, and the result is stored in EM2500.D.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ASC
RASC
DASC
STR
RDASC
HASC
RHASC
FASC
RFASC
VAL
LEN
SRGHT
RIGHT
SLEFT
LEFT
SMID
MID
SRPLC
REPLACE
SINS
INSERT
SDEL
DELETE
TRIM
LTRIM
RTRIM
STRIM
SPLIT
SFIND
SFINDN
INSTR
FIND
DISS
UNIS
CHR
CPSASC
RCPSASC

LTRIM

Trim 1-byte space and tab at left end of text string

Return value*1 = LTRIM (Destination text string)

Left trim

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Destination text string*2	Specifies the destination to store the text string to be trimmed, or the trim destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
R	Return value*2*4	Stores the trimmed text string.	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.

*2 For bit data, specify the start of the channel.

*3 To specify a text string using constants, enclose with " ".

*4 T, C, CTC, CTH, Z cannot be used.

Point

- **The LTRIM function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

All 1-byte spaces (20H) and labels (09H) at the left end (start) of the text string stored starting from S are deleted, the end code NUL (00H) is appended, and the result is stored in R.

S	HT(09H)	SP(20H)
S+1	SP(20H)	K(4BH)
S+2	e(65H)	y(79H)
S+3	e(65H)	n(6EH)
⋮	c(63H)	e(65H)
	SP(20H)	SP(20H)
	HT(09H)	NUL(00H)

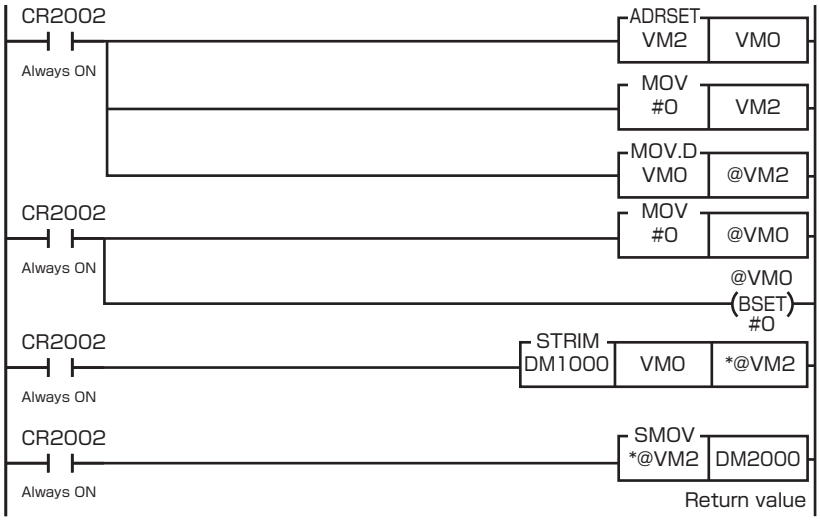
R	K(4BH)	e(65H)
R+1	y(79H)	e(65H)
R+2	n(6EH)	c(63H)
R+3	e(65H)	SP(20H)
R+4	SP(20H)	HT(09H)
R+5	NUL(00H)	

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

DM2000.T = **LTRIM** (DM1000.T)



Sample Program

● **When this function is used in combination with other functions**

Programmed Script

EM2500.D = RHASC (**LTRIM** (DM1000.T))

Operation Description

The 1-byte spaces and tabs are removed from the left end (leading) of a hexadecimal ASCII text string stored from DM1000.T.
The HEX ASCII text string resulting from the delete is converted to unsigned 16-bit BIN data, and the result is stored in EM2500.D.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ASC
RASC
DASC
STR
RDASC
HASC
RHASC
FASC
RFASC
VAL
LEN
SRGHT
RIGHT
SLEFT
LEFT
SMID
MID
SRPLC
REPLACE
SINS
INSERT
SDEL
DELETE
TRIM
LTRIM
RTRIM
STRIM
SPLIT
SFIND
SFINDN
INSTR
FIND
DISS
UNIS
CHR
CPSASC
RCPSASC

RTRIM

Trim 1-byte space and tab at right end of text string

Return value*1 = RTRIM (Destination text string)

Right trim

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Destination text string*2	Specifies the destination to store the text string to be trimmed, or the trim destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
R	Return value*2*4	Stores the trimmed text string.	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.

*2 For bit data, specify the start of the channel.

*3 To specify a text string using constants, enclose with " ".

*4 T, C, CTC, CTH, Z cannot be used.

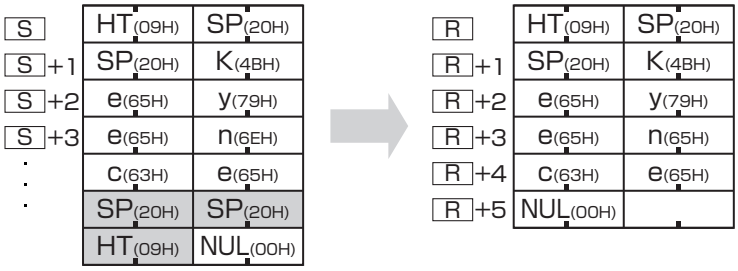
Point

- **The RTRIM function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

All 1-byte spaces (20H) and labels (09H) at the right end (end) of the text string stored starting from S are deleted, the end code NUL (00H) is appended, and the result is stored in R.

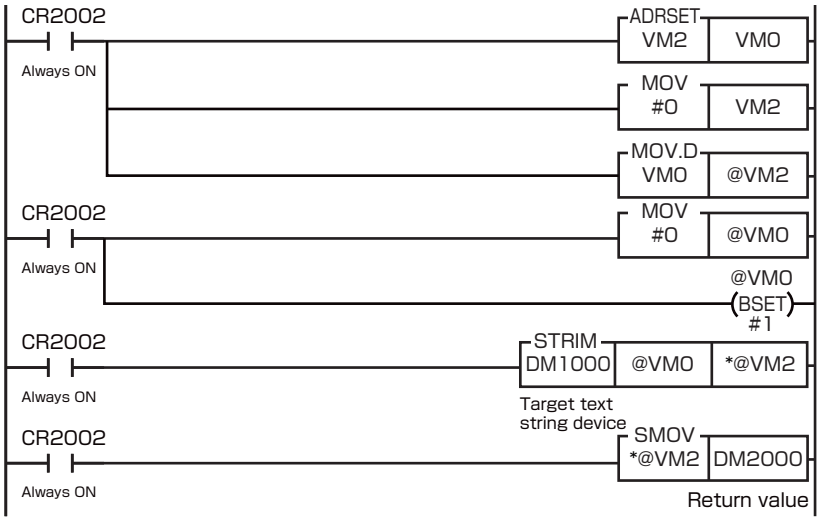


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

DM2000.T = **RTRIM** (DM1000.T)



Sample Program

● **When this function is used in combination with other functions**

Programmed Script

EM2500.D = RHASC (**RTRIM** (DM1000.T))

Operation Description

The 1-byte spaces and tabs are removed from the right end (leading) of a hexadecimal ASCII text string stored from DM1000.T.
The HEX ASCII text string resulting from the delete is converted to unsigned 16-bit BIN data, and the result is stored in EM2500.D.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ASC
RASC
DASC
STR
RDASC
HASC
RHASC
FASC
RFASC
VAL
LEN
SRGHT
RIGHT
SLEFT
LEFT
SMID
MID
SRPLC
REPLACE
SINS
INSERT
SDEL
DELETE
TRIM
LTRIM
RTRIM
STRIM
SPLIT
SFIND
SFINDN
INSTR
FIND
DISS
UNIS
CHR
CPSASC
RCPSASC

STRIM Trim 1-byte space and tab from specified text string position

Return value*1 = STRIM (Destination text string, delete position specification parameter)

Specified position trim

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	Formula
S1	Destination text string*2	Specifies the destination to store the text string to be trimmed, or the trim destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
S2	Delete position specifying parameter	Specifies the position to be trimmed (1 to 3).	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value*2*4	Stores the trimmed text string.	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 T, C, CTC, CTH, Z cannot be used.

Point

- **The STRIM function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

All 1-byte spaces and labels at the position specified by S2 are deleted from the text string stored starting from S1, the end code NUL (00H) is appended, and the result is stored in R.
The details specified with the delete position specifying parameter are given below.

S2 Value	Details
1	All 1-byte spaces and tabs at the left end of the text string are deleted. (Same action as LTRIM function)
2	All 1-byte spaces and tabs at the right end of the text string are deleted. (Same action as RTRIM function)
3	All 1-byte spaces and tabs at both ends of the text string are deleted. (Same action as TRIM function)

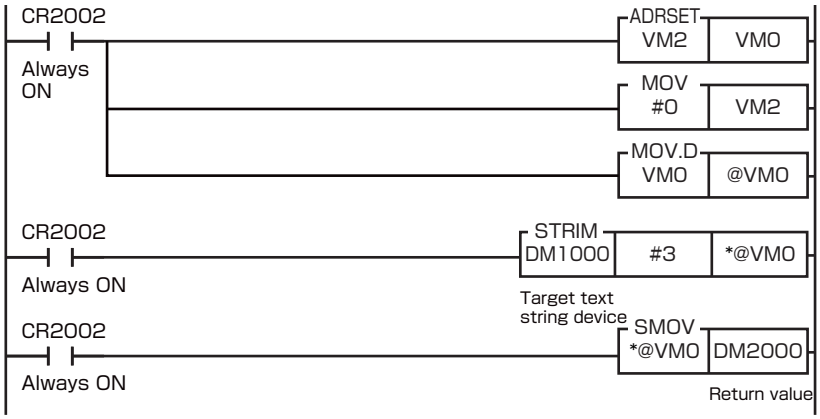
Point

If a parameter other than the above is specified, R is stored in S1 without executing the function.

"About operation flag (CR2009 to CR2012)", page 1-18
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

DM2000.T = **STRIM** (DM1000.T, 3)



Sample Program

● **When this function is used in combination with other functions**

Programmed Script

EM2500.D = RHASC (**STRIM** (DM1000.T, 3))

Operation Description

The 1-byte spaces and tabs are removed from both ends (leading and end) of a hexadecimal ASCII text string stored starting from DM1000.T.
The HEX ASCII text string resulting from the delete is converted to unsigned 16-bit BIN data, and the result is stored in EM2500.D.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ASC
RASC
DASC
STR
RDASC
HASC
RHASC
FASC
RFASC
VAL
LEN
SRGHT
RIGHT
SLEFT
LEFT
SMID
MID
SRPLC
REPLACE
SINS
INSERT
SDEL
DELETE
TRIM
LTRIM
RTRIM
STRIM
SPLIT
SFIND
SFINDN
INSTR
FIND
DISS
UNIS
CHR
CPSASC
RCPSASC

SPLIT Split text string

Return value*1 = SPLIT (split source text string, limit text string, split text string 1, split text string 2,• • •, split text string n)

Split

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Split source text string*2	Specifies the destination to store the split source text string, or the split target text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
S2	Limit text string*2	Specifies the destination to store the limit text string, or the limit target text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
D1	Split text string 1 *2	Specifies the destination to store the split text string 1.	—	—	—	—	—	—	—	.T	—	○	—
D2	Split text string 2 *2	Specifies the destination to store the split text string 2.	—	—	—	—	—	—	—	.T	—	○	—
⋮	⋮	⋮	—	—	—	—	—	—	—	.T	—	○	—
Dn	Split text string n *2*4	Specifies the destination to store the split text string n.	—	—	—	—	—	—	—	.T	—	○	—
R	Return value*2*5	Stores the number of splitting. .U type SPLIT function available.* 6	.U	.S	.D	.L	.F	.DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/ memory.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 Specify the argument number n in the range of 1 to 10.
*5 T, C, CTC, CTH, Z cannot be used.
*6 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.

Point

- **The SPLIT function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, KV-5500/ 5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

S1 is split with S2, and the results are stored in the D1 to Dn split text strings.
R stores the number of split text strings.

When S1 is "18,1A,35,E0,C3,EB" and S2 is "," (comma)

S	1(31H)	8(38H)
S+1	,(2CH)	1(31H)
S+2	A(41H)	,(2CH)
S+3	3(33H)	5(35H)
⋮	,(2CH)	E(45H)
	0(30H)	,(2CH)
	C(43H)	3(33H)
	,(2CH)	E(45H)
	B(42H)	NUL(00H)

D1	1(31H)	8(38H)
D2	1(31H)	A(41H)
D3	3(33H)	5(35H)
D4	E(45H)	0(30H)
D5	C(43H)	3(33H)
D6	E(45H)	B(42H)

"6" is stored in R

Reference

- When [R] nunber of splitting =0, the function is not executed.
- When 0 < [R] number of splitting < n, only the end code (NUL) is stored for the excessive argument.
- When [R] number of splitting = n, [Dn] may contain a limit character (splitting may still be possible).
  If only judging whether splitting is possible, the limit characters in [Dn] can be checked with the SFIND function.
  If there is a possibility of splitting, the text string can be split by specifying [S1] in [Dn] and executing the SPLIT function.

"About operation flag (CR2009 to CR2012)", page 1-18

● **Example of format**

DM5000.U = **SPLIT** (DM2000.T, ",", EM3000.T, EM3100.T, EM3200.T)

Sample Program

Programmed Script

DM2000.T = "Date, data, record"
DM5000.U = **SPLIT** (DM2000.T, ",", EM3000.T, EM3100.T, EM3200.T)

Operation Description

The text string "date, data, record" stored in DM2000.T is split by "," (comma) and stored in EM3000.T, EM3100.T, and EM3200.T.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

SFIND Search text string

Return value*1 = SFIND (search destination, search text string destination, search leading position)

Search text string

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Search destination*2*6	Specifies the destination to store the text string to be searched, or the search destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
S2	Search text string destination*2*6	Specifies the destination to store the text string to search, or the search text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
n	Search leading position*4*5	Specifies the search leading position (number of bytes). (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
R	Return value	.U type SFIND function available.*7	.U	.S	.D	.L	.F	*8 .DF	—	—	—	○	—

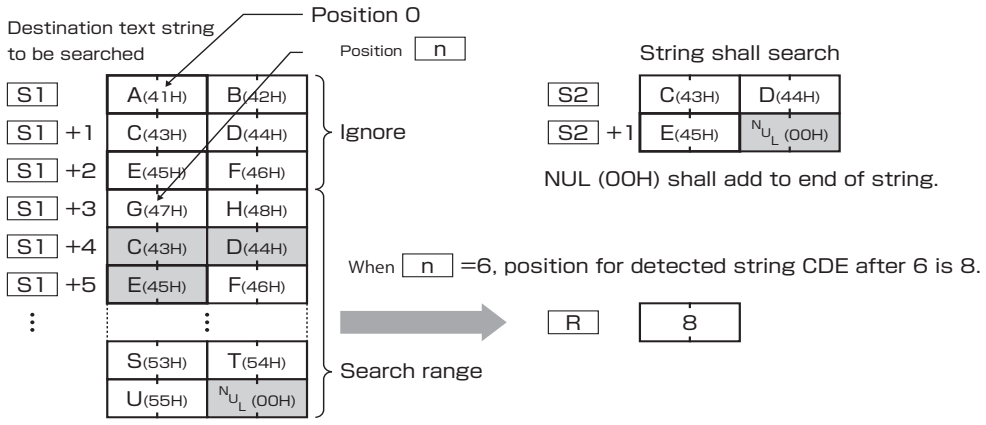
*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 $ cannot be used.
*5 Specify the number of characters (number of bytes) from the start of the search target string to the number of search characters within the range 0 to 1999.
*6 T, C, CTC, CTH, Z cannot be used.
*7 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.
*8 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

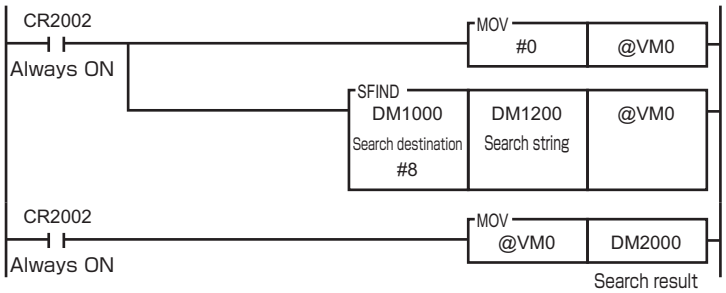
When the execution condition is established (TRUE), the search text string currently stored starting from S2 is searched from the position specified by n in the search destination text string currently stored starting from S1.
The leading character position (number of bytes starting from S1) of the search text string that is first found from n onwards as a result of the search is stored in R.
When the text string specified by S2 is not found, or the text string length of S2 is 0, $FFFF is stored in R.



"About operation flag (CR2009 to CR2012)", page 1-18
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● Example of format

DM2000.U = **SFIND** (DM1000.T, DM1200.T, 8)



Point

When any of the following conditions is established, "0" is stored in [R].

- **When the text string specified by [S1] and [S2] does not contain the end code (NUL (00H))**
- **When the text string including the end code (NUL (00H)) exceeds 2000 characters**
- **When (length of [S1] text string) < (length of [n]+[S2] text string). (Length of text strings do not include end code "NUL (00H)")**
- **When a bit other than the start of the channel is specified when bit data is specified for [S1] and [S2].**
- **When a timer/counter is specified to [S1] and [S2] by indirect specification.**

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000.U = **SFIND** (DM1000.T, SMID (DM1200.T, 4, 6), 0)

Operation Description

The text string currently stored starting from DM1000.T is searched from the start of the search destination text string currently stored starting from DM1200.T, and the text string resulting from having cut six characters (number of bytes) from the 4th character (number of bytes) is searched.
The leading character position of the search text string that was first found from the start of the search destination text string as a result of the search is stored in DM2000.U.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "SMID function", page 7-192
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

SFINDN Search text string

Return value*1 = SFINDN (search destination, search text string destination, number of search target bytes, search leading position)

Search text string N

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Search destination*2*4	Specifies the destination to store the text string to be searched, or the search destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
S2	Search text string destination*2*4	Specifies the destination to store the text string to search, or the search text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
S3	Number of search target bytes	Specifies the length (number of bytes) of the text string to be searched.	.U	.U	.U	.U	—	—	—	—	○	○	○
S4	Search leading position	Specifies the search leading position (number of bytes).	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	.U type SFINDN function available.*5	.U	.S	.D	.L	.F	*6 .DF	—	.T	—	○	—

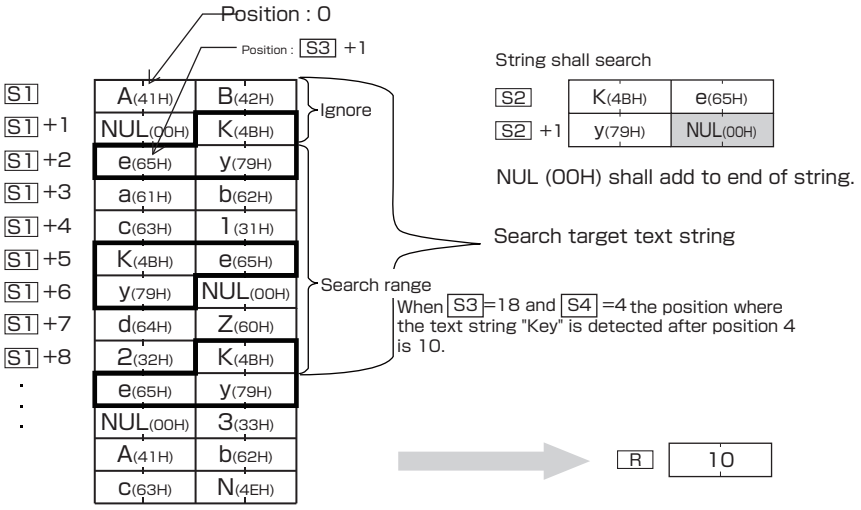
*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 T, C, CTC, CTH, Z cannot be used.
*5 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.
*6 The ".DF" suffix cannot be specified for device Z.

Point

- **The SFINDN function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

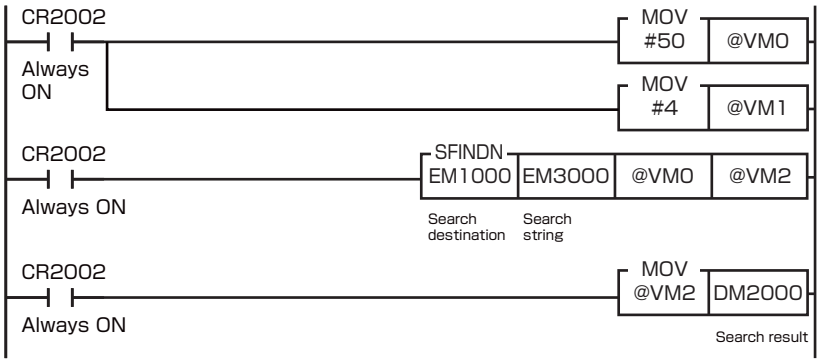
When the execution condition is established (TRUE), the text string stored starting from S2 is searched from S4 of the text string of the number of search target bytes starting from S1.
R stores the start position (number of bytes) of the text string specified by S2 found for the first time after the position specified by S4 in the text string specified by S3.
When the text string specified by S2 is not found, or the text string length of S2 is 0, "$FFFF" is stored in R.



"About operation flag (CR2009 to CR2012)", page 1-18
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

DM2000.U = SFINDN (EM1000.T, EM3000.T, 50, 4)



Sample Program

Programmed Script

EM1000.T = "abcdefghijklmnabcdefghijklmn"

EM3000.T = "abc"

DM2000.U = **SFINDN** (EM1000.T, EM3000.T, 50, 4)

Operation Description

The EM3000.T text string (abc) is searched for from the 50 characters in the text string stored in EM1000.T. The search starts from the fourth character of EM1000.T.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

INSTR Search text string

Return value*1 = INSTR (search leading position, search destination, text string search destination)

Search text string N

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
n	Search leading position*5	Specifies the search leading position (number of bytes). (0 to 1999)	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
S1	Search destination*2*6	Specifies the destination to store the text string to be searched, or the search destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
S2	Search text string destination*2*6	Specifies the destination to store the text string to search, or the search text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
R	Return value	.U type INSTR function available.*7	.U	.S	.D	.L	.F	*8 .DF	—	—	—	○	—

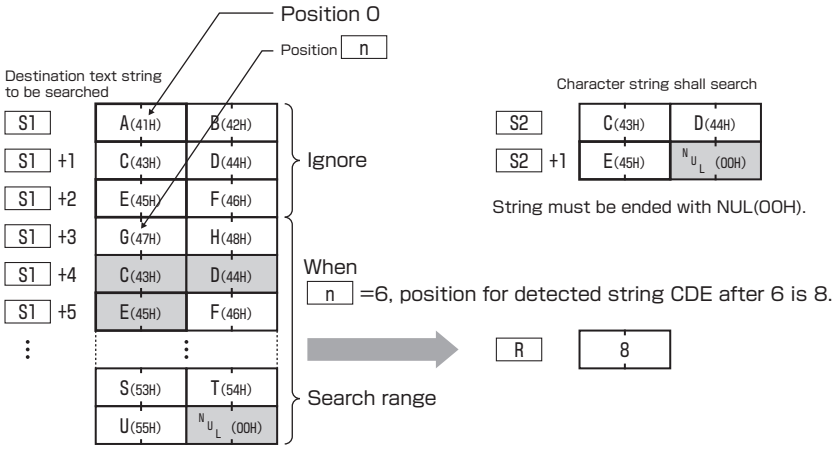
*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 $ cannot be used.
*5 Specify the number of characters (number of bytes) from the start of the search target string to the number of search characters within the range 0 to 1999.
*6 T, C, CTC, CTH, Z cannot be used.
*7 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.
*8 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

When the execution condition is established (TRUE), the search text string currently stored starting from [S2] is searched from the position (number of bytes) specified by [n] in the search destination text string currently stored starting from [S1].
The leading character position (number of bytes starting from [S1]) of the search text string that is first found from [n] onwards as a result of the search is stored in [R].
When the text string specified by [S2] is not found, or the text string length of [S2] is 0, "$FFFF" is stored in [R].



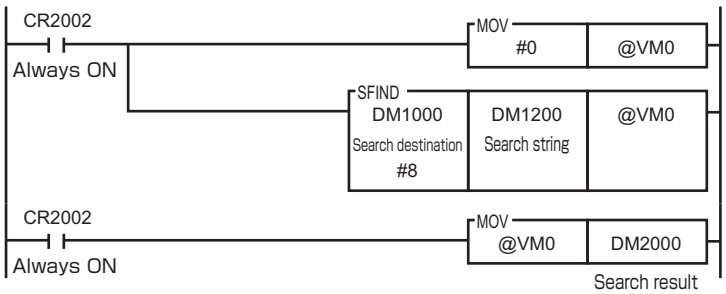
String must be ended with NUL(00H).

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● Example of format

DM2000.U = **INSTR** (8, DM1000.T, DM1200.T)



Point

When any of the following conditions is established, "0" is stored in [R].

- **When the text string specified by [S1] and [S2] does not contain the end code (NUL (00H))**
- **When the text string including the end code (NUL (00H)) exceeds 2000 characters**
- **When (length of [S1] text string) < (length of [n]+[S2] text string). (Length of text strings do not include end code "NUL (00H)")**
- **When a bit other than the start of the channel is specified when bit data is specified for [S1] and [S2].**
- **When a timer/counter is specified to [S1] and [S2] by indirect specification.**

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000.U = **INSTR** (0, DM1000.T, SMID (DM1200.T, 4, 6))

Operation Description

The text string currently stored starting from DM1000.T is searched from the start of the search destination text string currently stored starting from DM1200.T, and the text string resulting from having cut six characters (number of bytes) from the 4th character (number of bytes) is searched.
The leading character position of the search text string that was first found from the start of the search destination text string as a result of the search is stored in DM2000.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "SMID function", page 7-192
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

MEMO

FIND Search text string

Return value*1 = FIND (search destination, text string search destination)

Search text string

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S1	Search destination*2*4	Specifies the destination to store the text string to be searched, or the search destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
S2	Search text string destination*2*4	Specifies the destination to store the text string to search, or the search text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
R	Return value	.U type FIND function available.* 5	.U	.S	.D	.L	.F	*6 .DF	—	—	—	○	—

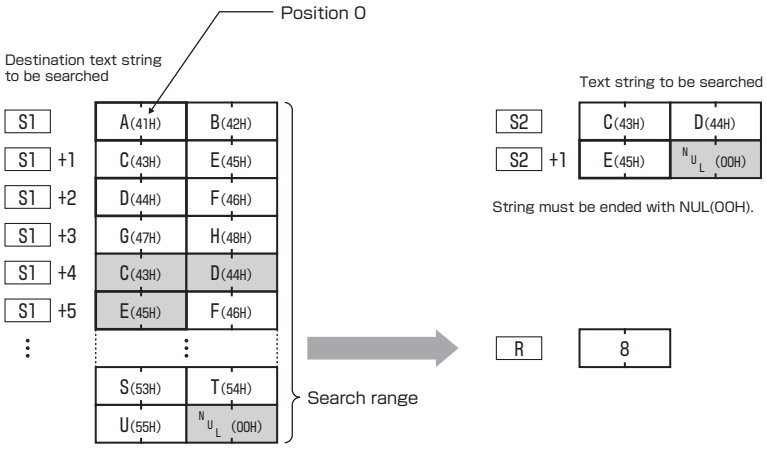
*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " " (up to 31 1-byte characters).
*4 T, C, CTC, CTH, Z cannot be used.
*5 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.
*6 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument ".DF" can be used only with KV STUDIO Ver.6.00 or later.**
- **The argument ".DF" can be specified only for KV-8000/7000 series CPU units, KV-5500/5000/3000 series CPU units with CPU function version 2.0 or later, and KV Nano series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

When the execution condition is established (TRUE), the search text string currently stored starting from S2 is searched from the search destination text string currently stored starting from S1.
The leading character position (number of bytes starting from S1) of the search text string that is first found as a result of the search is stored in R.



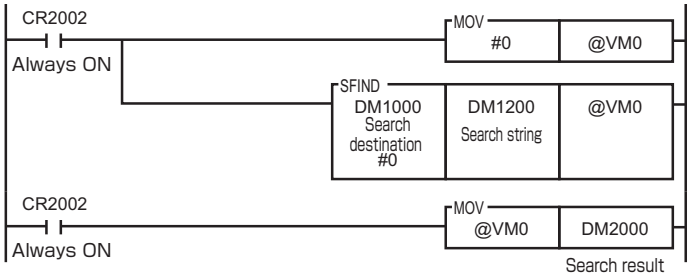
String must be ended with NUL(00H).

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

DM2000 = **FIND** (DM1000.T, DM1200.T)



Point

When any of the following conditions is established, "0" is stored in [R].

- **When the text string specified by [S1] and [S2] does not contain the end code (NUL (00H))**
- **When the text string including the end code (NUL (00H)) exceeds 2000 characters**
- **When (length of [S1] text string) < (length of [S2] text string).**
- **When a bit other than the start of the channel is specified when bit data is specified for [S1] and [S2].**
- **When a timer/counter is specified to [S1] and [S2] by indirect specification.**
- **When [S2] text string length is 0, or no text string is found, 65535 is input to [R].**

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

DM2000.U = **FIND** (DM1000.T, SMID (DM1200.T, 4, 6))

Operation Description

The text string currently stored starting from DM1000.T is searched from the start of the search destination text string currently stored starting from DM1200.T, and the text string resulting from having cut six characters (number of bytes) from the 4th character (number of bytes) is searched.
The leading character position of the search text string that was first found from the start of the search destination text string as a result of the search is stored in DM2000.U.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "SMID function", page 7-192
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

DISS Disperse text string

Disperse text string

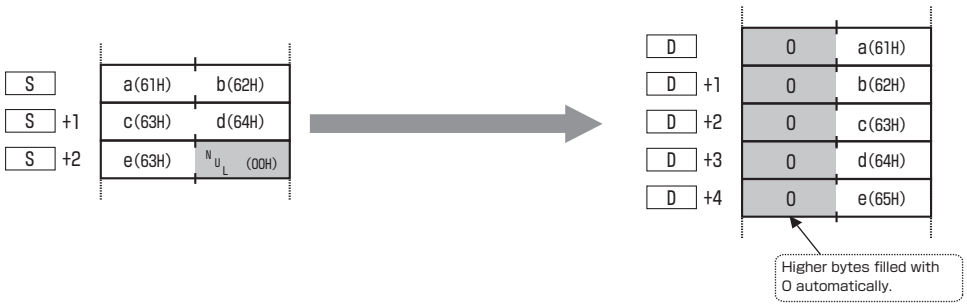
DISS ([execution condition,][*1] disperse destination, storage destination)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Disperse destination*2*4	Specifies the destination to store the text string to be dispersed, or the disperse destination text string.	—	—	—	—	—	—	—	.T	*3 ○	○	○
D	Storage destination*2 *5	Specifies the destination to store the dispersed text string (in words).	.U	.U	.U	.U	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)
*2 For bit data, specify the start of the channel.
*3 To specify a text string using constants, enclose with " ".
*4 T, C, CR, CTC, CTH, Z cannot be used.
*5 T, C, CTC, CTH, Z cannot be used.

Description of Operation

When the execution condition is established (TRUE), the text string in bytes currently stored starting from S is dispersed in words and stored starting from D.

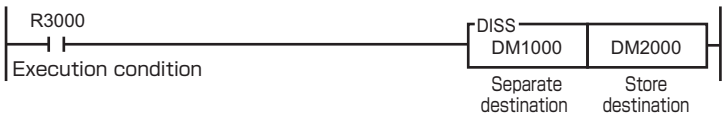


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● Example of format

DISS (R3000, DM1000.T, DM2000)



Point

When any of the following conditions is established, "0" is stored in D.

- **When the text string specified by S does not contain the end code (NUL (00H))**
- **When the text string including the end code (NUL (00H)) exceeds 2000 characters**
- **When a bit other than the start of the channel is specified when bit data is specified for S and D.**
- **When a timer/counter is specified to S and D by indirect specification.**

Sample Program

● When this function is used in combination with other functions

Programmed Script

DISS (SMID (DM1000.T, EM1000, EM1001), DM2000)

Operation Description

The text string currently stored starting from DM1000.T is cut from the character position (number of bytes) specified by EM1000 by the number of characters (number of bytes) specified by EM1001 at every scan (no execution condition).
The text string in byte units resulting from the cut is dispersed to word units starting from DM2000 and is stored.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "SMID function", page 7-192
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

ASC
RASC
DASC
STR
RDASC
HASC
RHASC
FASC
RFASC
VAL
LEN
SRGHT
RIGHT
SLEFT
LEFT
SMID
MID
SRPLC
REPLACE
SINS
INSERT
SDEL
DELETE
TRIM
LTRIM
RTRIM
STRIM
SPLIT
SFIND
SFINDN
INSTR
FIND
DISS
UNIS
CHR
CPSASC
RCPSASC

UNIS Unite text string

Unite text string

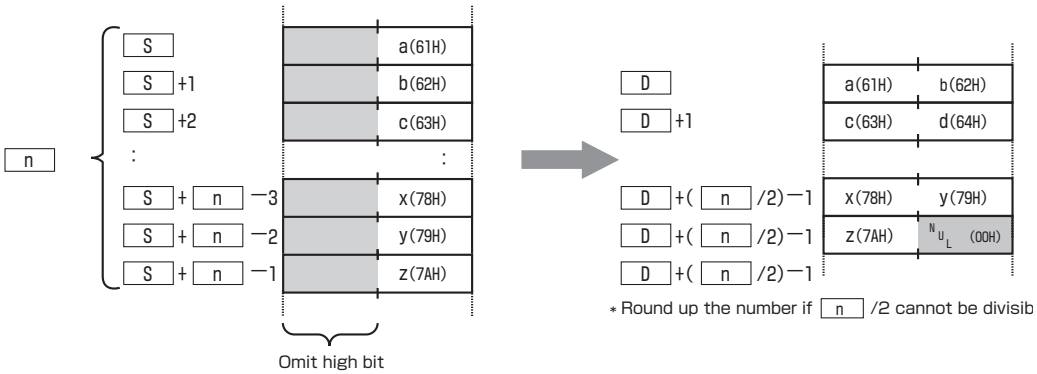
UNIS ([execution condition,]*1 unite destination, storage destination, number of unite characters)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Unite destination*2*5	Specifies the destination to store the text string (in words) to be united, or the unite destination text string (in words).	.U	.U	.U	.U	—	—	—	—	—	○	○
D	Storage destination*2 *6	Specifies the destination to store the united text string.	—	—	—	—	—	—	—	.T	—	○	—
n	Number of unite characters*4	Specifies the number of characters (number of bytes or number of words) to unite. (1 to 1999)	.U	.U	.U	.U	—	—	—	—	*3 ○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)
*2 For bit data, specify the start of the channel.
*3 $ cannot be used.
*4 Specify the number of unite characters within the range 1 to 1999.
*5 T, C, CTC, CTH, Z cannot be used.
*6 T, C, CR, CTC, CTH, Z cannot be used.

Description of Operation

When the execution condition is established (TRUE), the text string in words currently stored starting from [S] is united into byte unit by the number of characters (number of words) specified by [n], appended with the end code NUL (00H) and stored starting from [D].

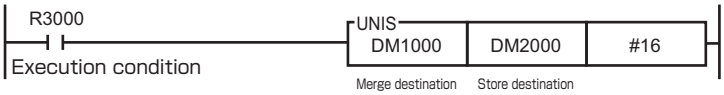


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

UNIS (R3000, DM1000, DM2000.T, 16)



Point

The function is not executed when any of the following conditions is established:

- **When the value of [n] is 0 or 2000 or more.**
- **When the text string including the end code (NUL (00H)) exceeds 2000 characters**
- **When a bit other than the start of the channel is specified when bit data is specified for [S] and [D].**
- **When a timer/counter is specified to [S] and [D] by indirect specification.**

Sample Program

● **To specify the bit data**

Programmed Script

UNIS (MR3000.U < 10, MR1000.U, MR2000.T, 10)

Operation Description

When MR3000.U is smaller than 10, the text string in word units currently stored starting from MR1000.U is united to byte units for ten characters (number of word units), and appended with the end code NUL (00H) and stored starting from MR2000.T.

● **When this function is used in combination with other functions**

Programmed Script

UNIS (DM1000, DM2000.T, ANDA (EM1000, $FF))

Operation Description

The text string in word units currently stored starting from DM1000 is united to byte units for the number of characters (number of word units) specified by the lower eight bits (extracted) of EM1000, and is appended with the end code NUL (00H) and stored starting from DM2000.T at every scan (no execution condition).

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "ANDA function", page 7-22
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

CHR Convert HEX ASCII code to characters

Return value*1 = CHR (conversion destination)

Convert characters

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Conversion destination	Specifies the destination to store the ASCII code to be converted or the ASCII code data. (16-bit data)	.U	.U	.U	.U	—	—	—	—	*2 ○	○	○
R	Return value*3	.T type CHR function available	—	—	—	—	—	—	—	.T	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 The value of the constant is within the range 0 to 255. When a destination is specified and the value of the destination is outside 0 to 255, the value obtained by discarding the high-order eight bits is used.

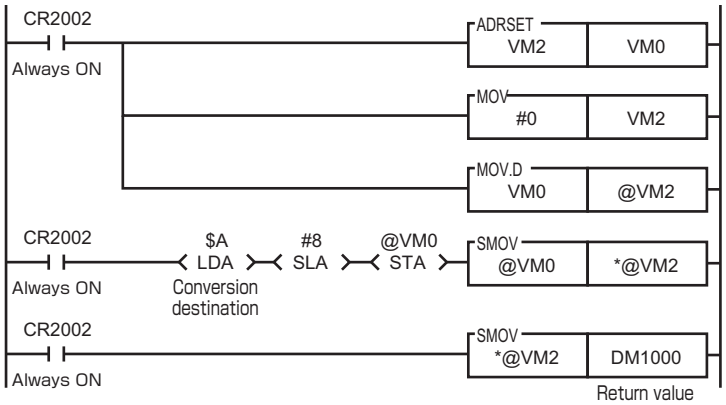
*3 T, C, CR, CTC, CTH, Z cannot be used.

Description of Operation

The ASCII code data stored in S is converted to a 1-byte text string appended with the end code NUL (00H), and the result is stored in R.

● Example of format

DM1000.T = CHR ($0A)



Sample Program

Programmed Script

DM2000.T = DM2000.T + **CHR** ($0D) + **CHR** ($0A)

Operation Description

The CR code (0DH) and LF code (0AH) are appended at the end of the text string DM2000.T.

Reference

- Specify ASCII code (half-width) to the conversion destination data.
  "ASCII Code Table", page A-53
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "Create a box script Step 5"(page 2-10).

MEMO

ASC
RASC
DASC
STR
RDASC
HASC
RHASC
FASC
RFASC
VAL
LEN
SRGHT
RIGHT
SLEFT
LEFT
SMID
MID
SRPLC
REPLACE
SINS
INSERT
SDEL
DELETE
TRIM
LTRIM
RTRIM
STRIM
SPLIT
SFIND
SFINDN
INSTR
FIND
DISS
UNIS
CHR
CPSASC
RCPSASC

CPSASC CIP text string conversion

Return value*1 = CPSASC (target data storage destination)

Argument/Return Value		Description	Supported Type								Const	Device/ Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Conversion target data storage destination*2	Specifies the start of the destination that stores the CIP text string data before conversion.	.U	.S	.D	.L	.F	.DF	.B	.T	—	○	—
R	Return value	.T type CPSASC function available	—	—	—	—	—	—	—	.T	—	—	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.

*2 When specifying bit data, specify the start of the channel.

Point

- **The CPSASC function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

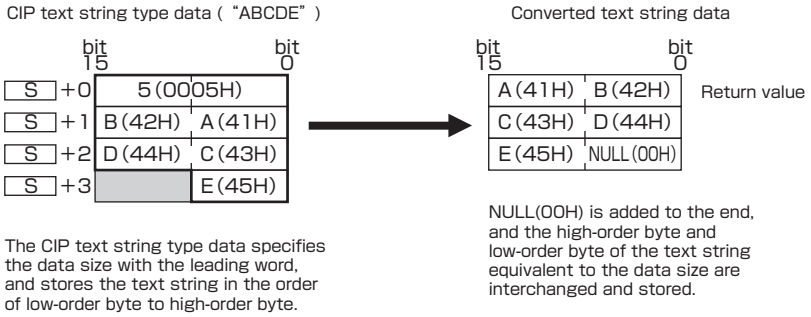
Description of Operation

The CIP text string type data stored starting from S is converted into a text string, and the result is stored in R.

The CPSASC function is used to convert the CIP text string type data, and process it as a text string.

Refer to "KV EtherNet/IP Function User's Manual" for details on the CIP data types.

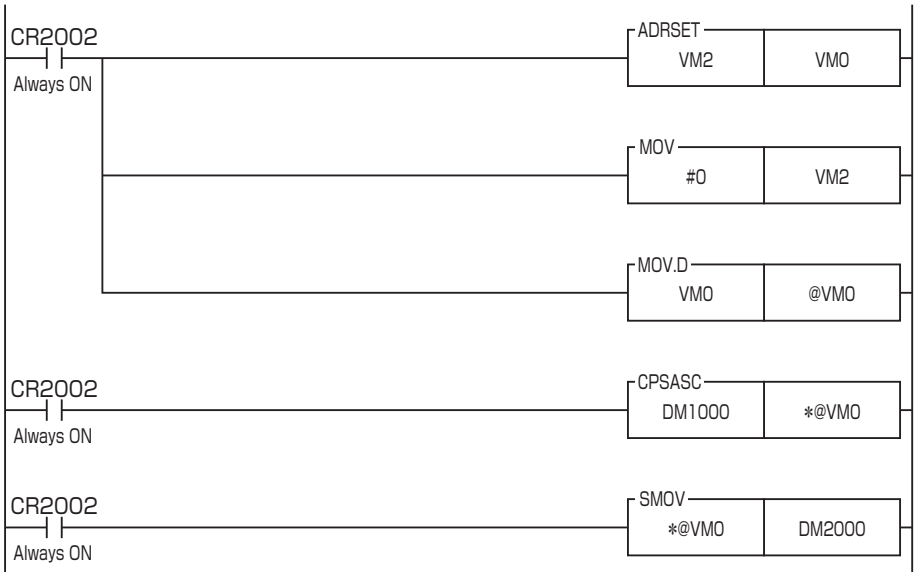
Refer to "EtherNet/IP Function User's Manual" for details on the CIP data types.



For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

DM2000.T = **CPSASC** (DM1000)



RCPSASC CIP text string type data reverse conversion

Return value*1 = RCPSASC (target data storage destination, converted data storage destination)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device/ Variable	For mula
S	Conversion target data storage destination	Specifies the start of the destination that stores the text string before conversion.	—	—	—	—	—	—	—	.T	*3 ○	○	—
D	Converted data storage destination	Specifies the start of the destination that stores the converted CIP text string type data.	.U	.S	.D	.L	.F	.DF	.B	.T	—	○	
R	Return value*2	The number of characters (in bytes) stored in D is stored. The RCPSASC function uses the .U function.	.U	.S	.D	.L	.F	.DF	—	—	—	—	—

*1 The return value can be omitted. When omitted, the conversion result is stored in a work device/memory.

*2 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.

*3 To specify a text string using constants, enclose with " ".

Point

- **The RCPSASC function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

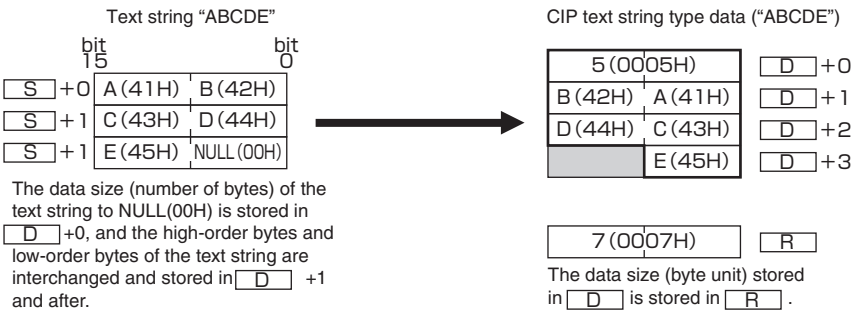
Description of Operation

The text string stored in S is converted into CIP text string type data, and the result is stored in D. After conversion, the number of characters (in byte) stored in D is stored in R.

The RCPSASC function is used to convert the CIP text string type data, and process it as a text string.

Refer to "KV EtherNet/IP Function User's Manual" for details on the CIP data types.

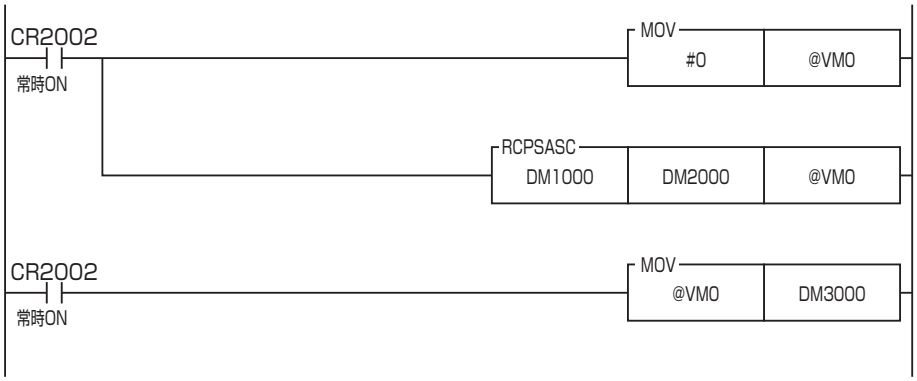
When the RCPSASC function is executed, the CIP text string type data is converted and stored in the return value.



For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Text Processing Instructions".

● **Example of format**

DM3000 = **RCPSASC** (DM1000.T, DM2000)



MEMO

8

EXTENDED FUNCTIONS

Data processing function 8-2
Table processing function 8-48
Clock processing function 8-70
High-speed processing function 8-78
PID Functions 8-84
Log Functions 8-86
Storage Functions 8-90
Access Window Functions 8-118
FREE RUN Counter Functions 8-124
Sensor Setting Function 8-126
Cyclic Communication Refresh Function 8-132

SEG Decode 4-bit data to 7-segment

Return value *1 = SEG (decode destination device, destination digit position)

7-segment

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Decode destination device*2	Specifies the device to which the decode destination data is stored or the decode destination data.	.U	.U	.D	.D	—	—	—	—	—	○	○
n	Destination digit position*3	Specifies the device to which the digit position to decode is stored, or the decode digit position. (0 to 3 for a .U suffix instruction, and 0 to 7 for a .D suffix instruction)	.U	.U	.U	.U	—	—	—	—	—	○	○
R	Return value	.U type SEG function available*4	.U	.S	.D	.L	.F	*5 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 S When the decoding target is specified as bit data, consecutive 16/32 bits are processed. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

*3 When a device is specified to the n destination digit position, and the value stored in device exceeds 0 to 3 or 0 to 7, operation is as follows.
When the decode destination device suffix is .U: The values of the lower two bits of the specified device are used.
When the decode destination device suffix is .D: The values of the lower three bits of the specified device are used.
For example, when the suffix of the S decode destination device is .U, DM01000 is specified to n, and that value is 23 (2BIN: 0000 0000 0001 0111), "3" is used as the value of n as the value of the lower two bits is "3".

*4 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.

*5 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument device ".DF" can be used only with the KV STUDIO Ver. 6.00 and higher versions.**
- **The argument device ".DF" can only be specified with KV-8000/7000 Series CPU units, with KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

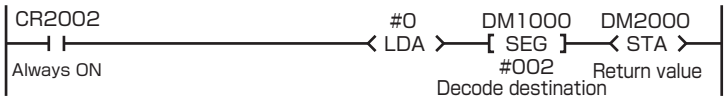
S The four bits (one digit) of digits of the 16-bit/32-bit data stored in n are decoded to data (8 bits) for 7-segment display, and the result is stored in R.

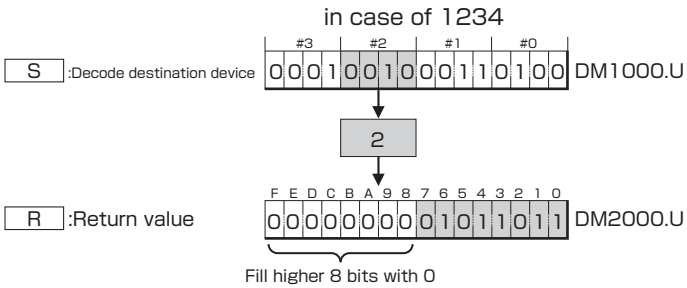
"About operation flag (CR2009 to CR2012)", page 1-18

● Example of format

- S : When decode destination device is 16 bits (.U/.S)

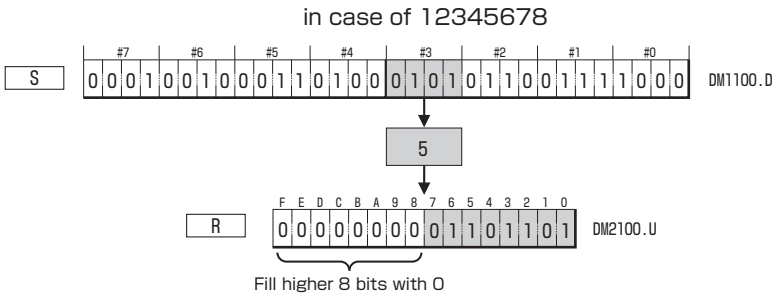
DM2000.U = **SEG** (DM1000.U, 2)





- S : When decode destination device is 32 bits (.D/.L)

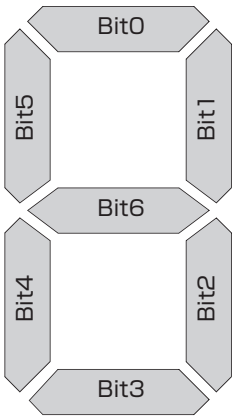
DM2100.U = **SEG** (DM1100.D, 3)



For details on execution timing, see 📖 KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions."

● Data table for 7-segment display

Value of Specified	16-bit internal register							
	Bit7	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**
$0	0	0	1	1	1	1	1	1
$1	0	0	0	0	0	1	1	0
$2	0	1	0	1	1	0	1	1
$3	0	1	0	0	1	1	1	1
$4	0	1	1	0	0	1	1	0
$5	0	1	1	0	1	1	0	1
$6	0	1	1	1	1	1	0	1
$7	0	0	1	0	0	1	1	1
$8	0	1	1	1	1	1	1	1
$9	0	1	1	0	1	1	1	1
$A	0	1	1	1	0	1	1	1
$B	0	1	1	1	1	1	0	0
$C	0	0	1	1	1	0	0	1
$D	0	1	0	1	1	1	1	0
$E	0	1	1	1	1	0	0	1
$F	0	1	1	1	0	0	0	1



SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

Sample Program

● When this function is used in combination with other functions

Programmed Script

R2000.U = **SEG** (TBCD (DM1000), 3)

Operation Description

The 16-bit BIN data currently stored in DM1000 is converted to 4-digit BCD (16-bit) data, and the value of the 1000's digit (3rd digit) of the conversion result is decoded to data for 7-segment display to turn ON the corresponding output relays (R2000 to R2015 (R2007)).

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "TBCD function", page 7-80
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

MEMO

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

BCNT ON bit count of specified device

Return value*1 = BCNT (operation destination)

Bit count

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S	Operation destination	Specifies the operation destination device or the operation destination data.	.U	.U	.D	.D	—	—	—	—	—	○	○
R	Return value	.U type SEG function available*2	.U	.S	.D	.L	.F	*3 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.

*3 The ".DF" suffix cannot be specified for device Z.

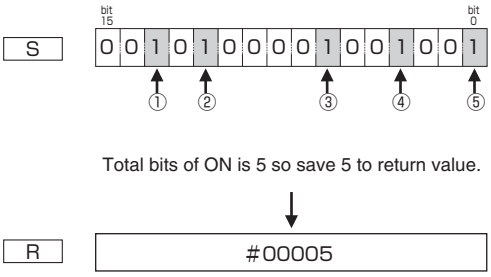
Point

- **The argument device ".DF" can be used only with the KV STUDIO Ver. 6.00 and higher versions.**
- **The argument device ".DF" can only be specified with KV-8000/7000 Series CPU units, with KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

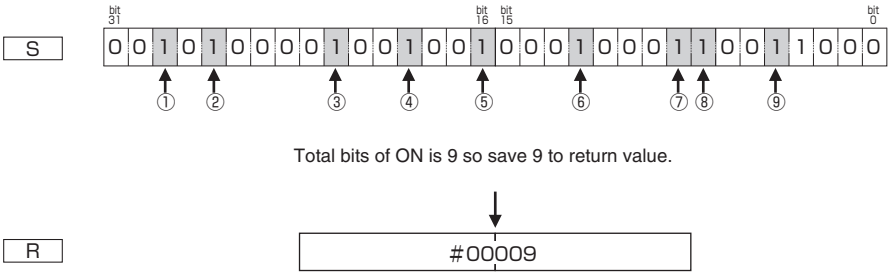
Description of Operation

The number of ON bits of the (16-bit/32-bit) BIN data stored in S are counted, and the result is stored in return value.

- S : When operation destination is 16 bits (.U/.S)



- S : When operation destination is 32 bits (.D/.L)



"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions."

● **Example of format**

DM2000.U = **BCNT** (DM1000.U)

Sample Program

● **To count a bit device**

Programmed Script

DM2000.U = **BCNT** (MR1000.D)

Operation Description

The ON bits of the 32-bit BIN data stored in MR1000.D are counted.
The result is stored in DM2000.U.

Ladder conversion

● **When this function is used in combination with other functions**

Programmed Script

DM2000.U = **BCNT** (ORA (DM1000.D, DM1100.D))

Operation Description

The data stored in DM1000.D is ORed by the data stored in DM1100.D, and the number of ON bits of the operation result is counted and stored in DM2000.U.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "ORA function", page 7-24
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, step 5, (page 2-10)

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

DCNT Count detected data in specified block

Return value *1 = DCNT (detection destination, detection range leading device, number of detected data)

Data count

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S1	Detection destination*2	Specifies the device to which the value to detect is stored, or the detection value.	.U	.U	.D	.D	.D	—	—	—	○	○	○
S2	Detection range leading device*2	Specifies the leading device of the data block to detect.	Handled as same type as S1.					—	—	—	—	○	—
n	Number of detected data	Specifies the device currently storing the number of data of the block to detect, or the number of detected data.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	.U type DCNT function available* 3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When a bit device has been specified, 16/32 continuous bits are handled. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

*3 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.

*4 The ".DF" suffix cannot be specified for device Z.

Point

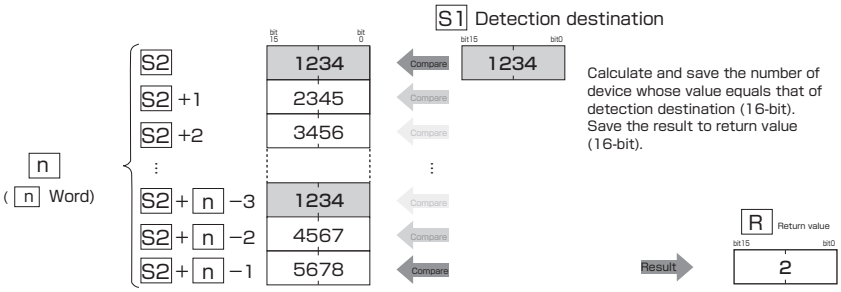
- **The argument device ".DF" can be used only with the KV STUDIO Ver. 6.00 and higher versions.**
- **The argument device ".DF" can only be specified with KV-8000/7000 Series CPU units, with KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.**

"KV-5500/5000/3000 CPU Function Version", page 4

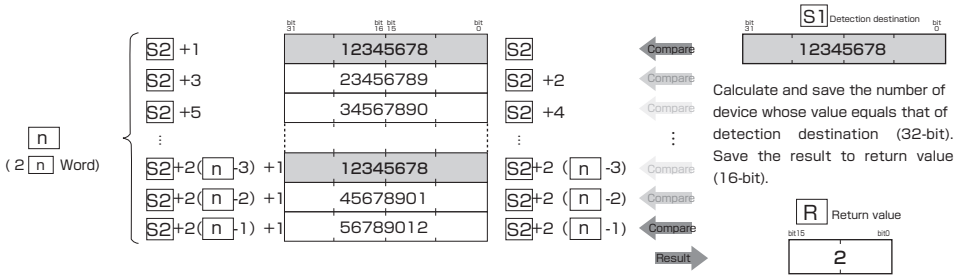
Description of Operation

S2 block data starting from n are each compared with the data in S1, and the number of matching data is stored in R.

- S1 : When detection destination is 16 bits (.U/.S)



- S1 : When detection destination is 32 bits (.D/.L)



"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions."

● Example of format

DM2000.U = **DCNT** (DM1000.U, DM1200.U, 4)

Point **n When the value of is "0", the value of the S1 is stored in R.**

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000.U = **DCNT** (COM (DM1000.D), MR1000.D, EM2000)

Operation Description

The 32-bit BIN data of number of detection data indicated by EM2000 starting from MR1000 (.D) are each compared with the value resulting from having inverted the bits of unsigned 32-bit BIN data currently stored in DM1000.D, and the number of matching data is stored in DM2000.U.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "COM function", page 7-30
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, step 5, (page 2-10)

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

SER Detect data in specified block

Return value *1 = SER (detection destination, detection range leading device, number of detected data)

Search

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S1	Detection destination*2	Specifies the device to which the value to detect is stored, or the detection value.	.U	.U	.D	.D	.D	—	—	—	○	○	○
S2	Detection range leading device*2	Specifies the leading device of the data block to detect.	Handled as same type as S1.					—	—	—	—	○	—
n	Number of detected data	Specifies the device currently storing the number of data of the block to detect, or the number of detected data.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	.UU type SER function available.*3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When a bit device has been specified, 16/32 continuous bits are handled. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

*3 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.

*4 The ".DF" suffix cannot be specified for device Z.

Point

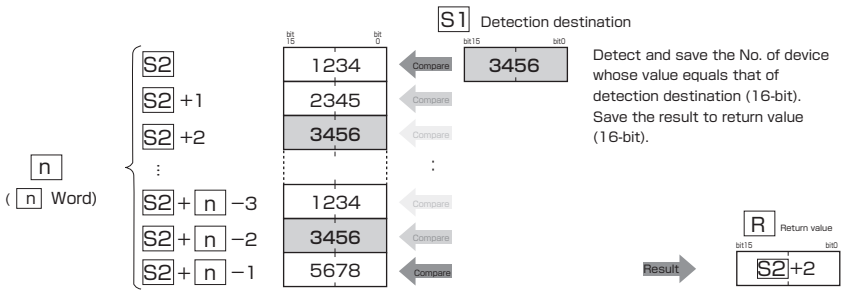
- **The argument device ".DF" can be used only with the KV STUDIO Ver. 6.00 and higher versions.**
- **The argument device ".DF" can only be specified with KV-8000/7000 Series CPU units, with KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

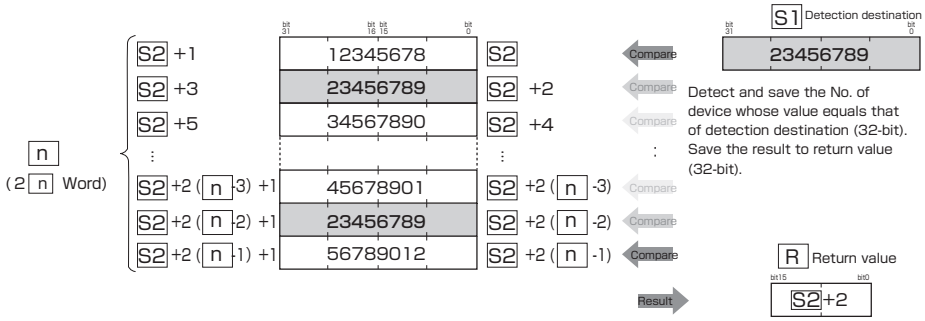
block data starting from S2 are each compared with the data in n, S1 and the number of matching data is stored in R .

When multiple matching data are detected, the smallest device No. is stored.

- S1 : When detection destination is 16 bits (.U/.S)



- S1 : When detection destination is 32 bits (.D/.L)



"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions."

● Example of format

DM2000.U = **SER** (DM1000.U, DM1200.U, 40)

Point

- **When no matching devices are found, or value of n is "0", the value of S1 is stored in R.**
- **When a bit device has been specified to the detection range leading device, the value obtained by converting not to 16-bit units but to decimal units taking R (MR, LR) 0000 as its reference point is stored in search result (return value). (example) When the search result is MR82312: 823x16+12=13180 is stored.**

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000.U = **SER** (NEG (DM1000.D), MR1000.D, EM2000.U)

Operation Description

The block of unsigned 32-bit BIN data of number of detection data indicated by EM2000 starting from MR1000.D are each compared with the values resulting from having inverted the sign of unsigned 32-bit BIN data currently stored in DM2000.U.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "NEG function", page 7-32
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, step 5, (page 2-10)

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

DSER Detect MAX value in a specified block

Return value *1 = DSER (Detection data, detection range leading device, number of detection device, number of detection store device)

Data retrieval

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Detection destination*2	Specify detection data or the device storing detection data.	.U	.U	.D	.D	.D	—	—	—	○	○	○
S2	Detection range leading device*2	Specifies the start of the range to be searched.	Handled as same type as S1.					—	—	—	—	○	—
n	Number of detection store device*2	Specifies the number of targets to be searched or the storage location for the number.	.U	.U	.U	.U	—	—	—	—	○	○	○
D	Number of detection store device*2	Specifies the target to store the detection result (number of detections).	.U	.U	.U	.U	—	—	—	—	—	○	—
R	Return value	Specifies the target to store the detection position (offset from the start of the detection range). .U type DSER function available.*3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in working memory.

*2 When a bit device has been specified, 16/32 continuous bits are handled. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

*3 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.

*4 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument device ".DF" can be used only with the KV STUDIO Ver. 6.00 and higher versions.**
- **The argument device ".DF" can only be specified with KV-8000/7000 Series CPU units, with KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.**

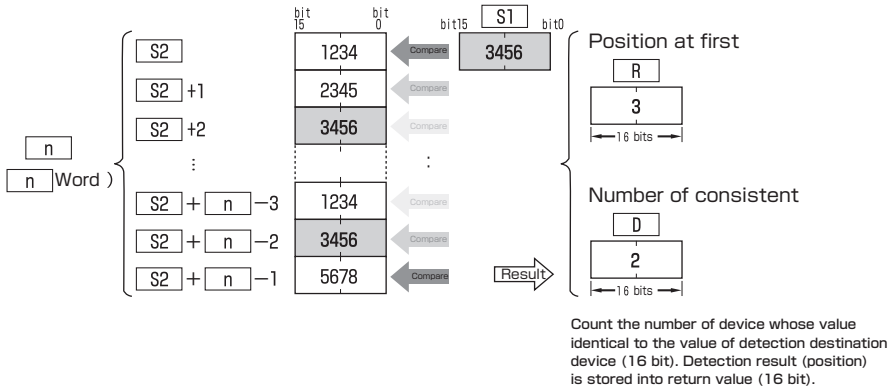
"KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

To search S1 devices starting from S2 by using n as keyword.

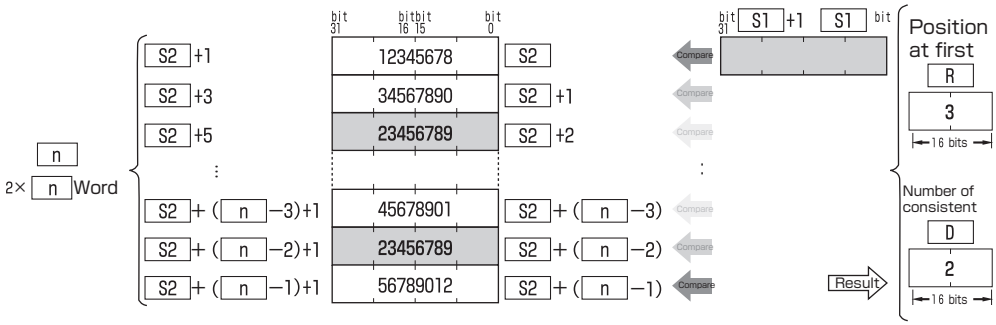
- The number of the device in conformity with S1 is stored in D.
- From the initial conformable device position S1 in relative position R.
- When conformable data not found, it is stored in D and R.

- S1 : When detection data is16 bits (.U)



Count the number of device whose value identical to the value of detection destination device (16 bit). Detection result (position) is stored into return value (16 bit).

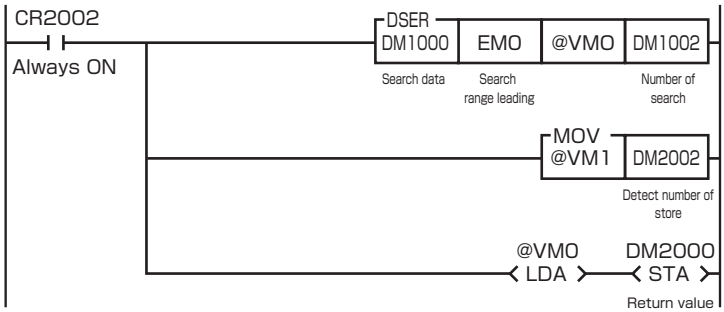
- S1 : When operation destination device is 32 bits



Count the number of device whose value identical to the value of detection destination device (32 bit). Detection result (position) is stored into return value (16 bit).

● Example of format

DM2000.U = **DSER** (DM1000.U, EM0, DM1002, DM2002)



SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

MAX

Detect maximum value in a specified block

Return value *1 = MAX (search destination device, number of search range data)

Search max. value

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Detection destination device*2*3*4	Specifies the start of the range to be searched.	.U	.S	.D	.L	.F	*6 .DF	—	—	—	○	—
n	Number of search range data*2	Specifies the device currently storing the number of data in the range to search, or the number of search range data.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	MAX function type is identical to the 1st argument.* 5	.U	.S	.D	.L	.F	*6 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 The range of numerical values that can be handled differs according to the instruction suffix.
*3 When a bit device has been specified, 16/32/64 continuous bits are handled. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16/32/64 bits straddling the next channel.
*4 When S type is .F or .DF, T and C cannot be specified.
*5 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.
*6 The ".DF" suffix cannot be specified for device Z.

Point

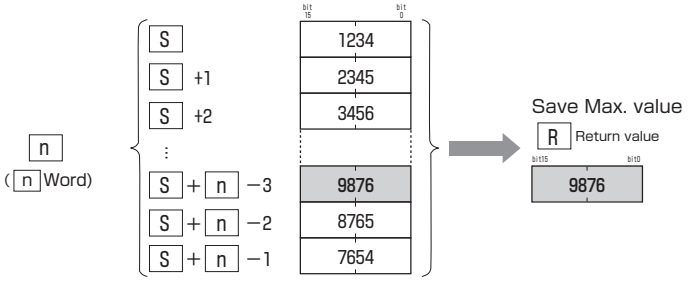
- **The argument devices ".F" and ".DF" can be used in KV STUDIO Ver. 9.0 or later only.**
- **The argument devices ".F" and ".DF" can be specified with the KV-8000 series CPU units and KV-7000 Series CPU units that have CPU function version 2.0 or later only.**
- **The return value device ".DF" can be used in KV STUDIO Ver. 6.00 or later only.**
- **The return value device ".DF" can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or later, and with KV Nano Series base units.**

"KV-7500/7300 CPU Function Version", page 3
"KV-5500/5000/3000 CPU Function Version", page 4

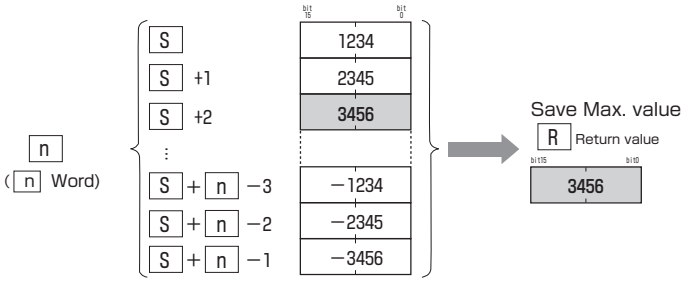
Description of Operation

The maximum value is searched in the (16-bit/32-bit/64-bit) BIN data stored in S words starting from n . (When the search destination is 32-bit data, the search range becomes the "specified number of data x 2 words" and when 64-bit data it becomes the "specified number of data x 4 words".) The maximum value as a result of the search is stored in R .

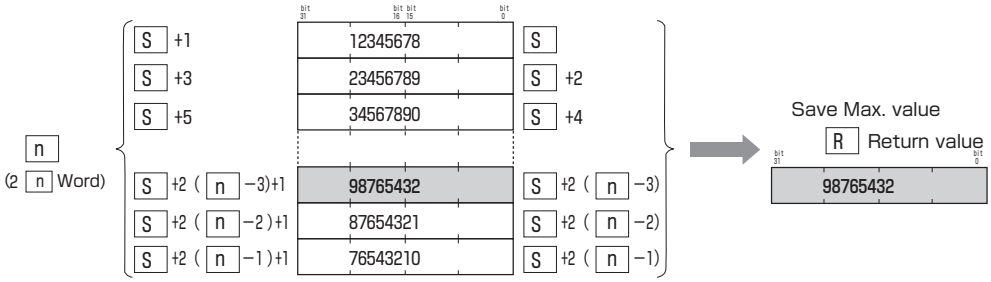
- S : When the search destination device is unsigned 16-bit data (.U)



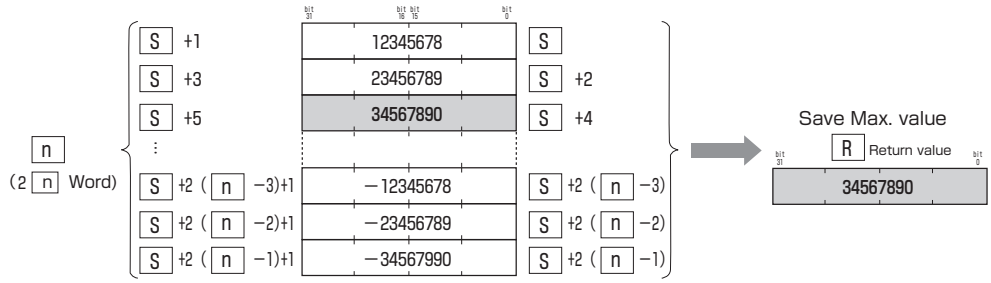
- [S] : When the search destination device is signed 16-bit data (.S)



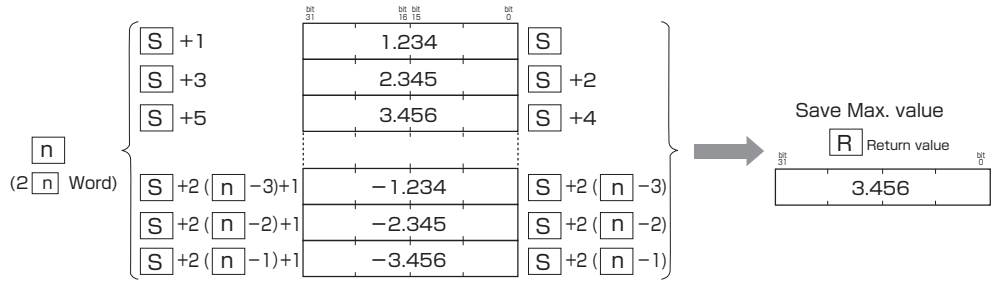
- [S] : When the search destination device is unsigned 32-bit data (.D)



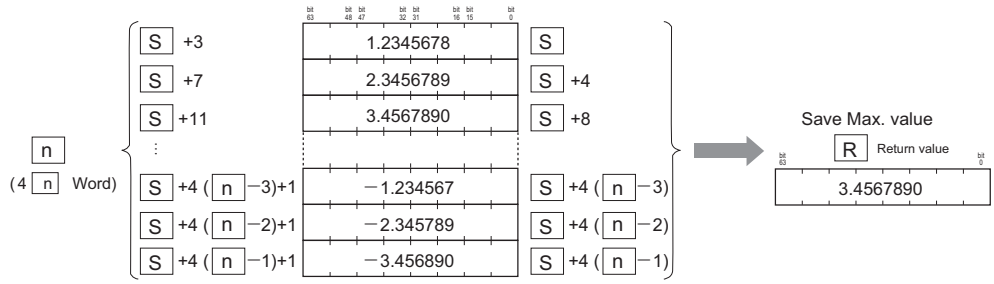
- [S] : When the search destination device is signed 32-bit data (.L)



- [S] : When the search destination device is a single precision floating point type (.F)



- [S] : When the search destination device is a double precision floating point type (.DF)



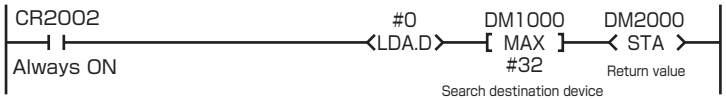
"About operation flag (CR2009 to CR2012)", page 1-18

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

For details on execution timing, see 📖 KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions."

● Example of format

DM2000.U = **MAX** (DM1000.U, 32)



Point n **When the value of** R **is 0, 0 is stored in :**

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000 = TBCD (**MAX** (DM1000.U, EM1100.U))

Operation Description

The maximum value is searched in the data stored in the range of EM1100 words starting from DM1000. The maximum value in the search result is converted to 4-digit BCD data, and stored in DM2000.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  📖 "TBCD function", page 7-80
- The KV script is converted to a ladder program before execution. To check the converted content, refer to 📖 "New a box script", step 5, step 5, (page 2-10)

MEMO

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

MIN

Detect minimum value in a specified block

Return value *1 = MIN (search destination device, number of search range data)

Minimum search

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S	Detection destination device *2*3*4	Specifies the start of the range to be searched.	.U	.U	.D	.D	.F	*6 .DF	—	—	—	○	—
n	Number of search range data *2	Specifies the device currently storing the number of data in the range to search, or the number of search range data.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	MIN function type is identical to the 1st argument *5	.U	.S	.D	.L	.F	*6 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 The range of numerical values that can be handled differs according to the instruction suffix.
*3 When a bit device has been specified, 16/32/64 continuous bits are handled. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16/32/64 bits straddling the next channel.
*4 When S type is .F or .DF, T and C cannot be specified.
*5 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.
*6 The ".DF" suffix cannot be specified for device Z.

Point

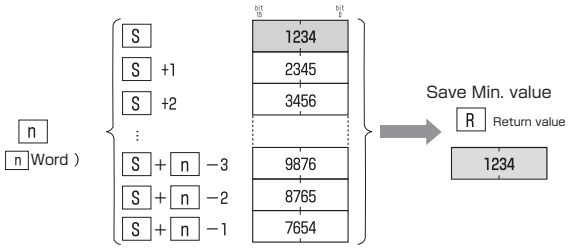
- **The argument devices ".F" and ".DF" can be used in KV STUDIO Ver. 9.0 or later only.**
- **The argument devices ".F" and ".DF" can be specified with the KV-8000 series CPU units and KV-7000 Series CPU units that have CPU function version 2.0 or later only.**
- **The return value device ".DF" can be used in KV STUDIO Ver. 6.00 or later only.**
- **The return value device ".DF" can only be specified with KV-7000 Series CPU units, KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or later, and with KV Nano Series base units.**

"KV-7500/7300 CPU Function Version", page 3
"KV-5500/5000/3000 CPU Function Version", page 4

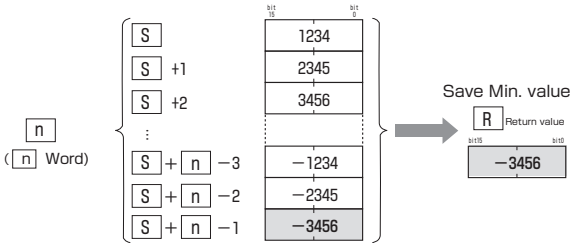
Description of Operation

The minimum value is searched in the (16-bit/32-bit/64-bit) BIN data stored in S words starting from n. (When the search destination is 32-bit data, the search range becomes the "specified number of data x 2 words" and when 64-bit data it becomes the "specified number of data x 4 words".) The minimum value as a result of the search is stored in R.

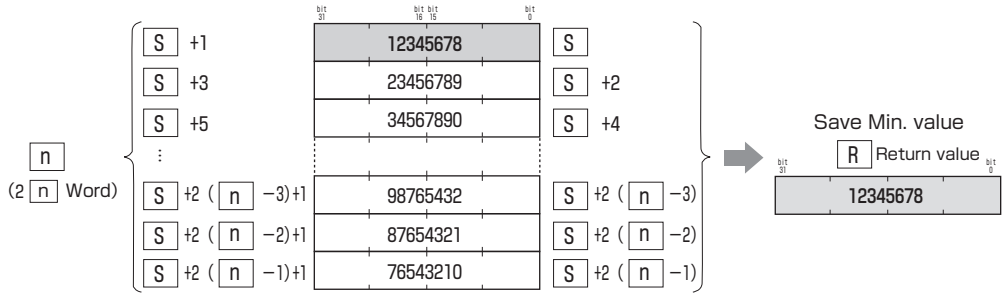
- S : When the search destination device is unsigned 16-bit data (.U)



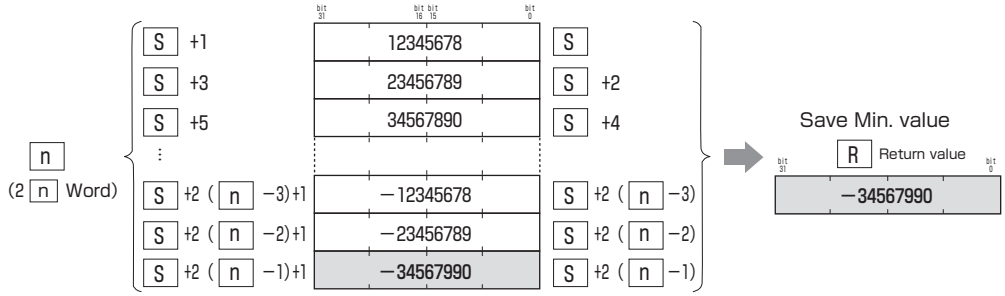
- S : When the search destination device is signed 16-bit data (.S)



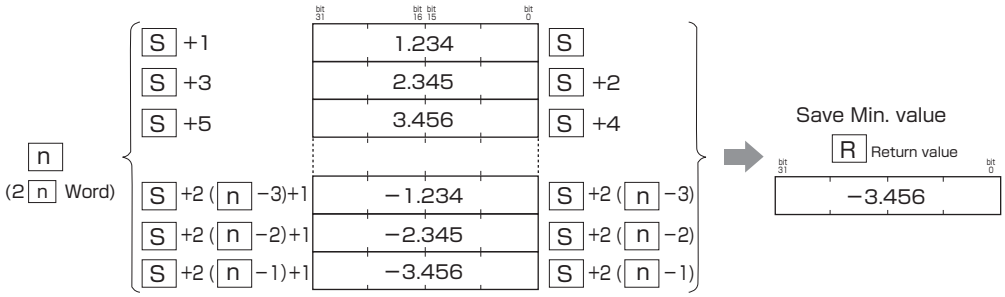
- S : When the search destination device is unsigned 32-bit data (.D)



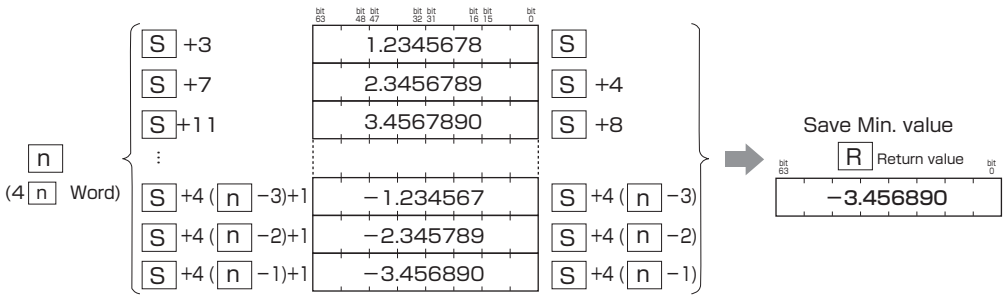
- S : When the search destination device is signed 32-bit data (.L)



- S : When the search destination device is a single precision floating point type (.F)



- S : When the search destination device is a double precision floating point type (.DF)



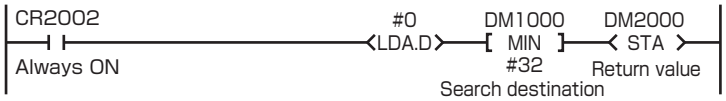
"About operation flag (CR2009 to CR2012)", page 1-18

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

For details on execution timing, see 📖 KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions."

● Example of format

DM2000.U = **MIN** (DM1000.U, 32)



Point n **When the value of** R **is 0, 0 is stored in :**

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000.S = NEG (**MIN** (DM1000.S, EM1100))

Operation Description

The minimum value is searched in the data stored in the range of EM1100 words starting from DM1000. The sign of the MIN value in the search result is inverted, and stored in DM2000.S.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  📖 "NEG function", page 7-32
- The KV script is converted to a ladder program before execution. To check the converted content, refer to 📖 "New a box script", step 5, (page 2-10)

MEMO

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

AVG Average value of specified block

Return value *1 = AVG (operation destination device, number of operation range data)

Average

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S	Operation destination device *2*3*4	Specifies the start of the range to be averaged.	.U	.U	.D	.D	.F	*6 .DF	—	—	—	○	—
n	Number of operation range data *2	Specifies the number of data in the range to average or the device to which that number is currently stored.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	AVG function type is identical to the 1st argument. *5	.U	.S	.D	.L	.F	*6 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 The range of numerical values that can be handled differs according to the instruction suffix.
*3 When a bit device has been specified, 16/32/64 continuous bits are handled. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16/32/64 bits straddling the next channel.
*4 When S type is .F or .DF, T and C cannot be specified.
*5 When the type different from the 1st argument is specified by return value, type conversion will be automatically conducted during conversion.
*6 The ".DF" suffix cannot be specified for device Z.

Point

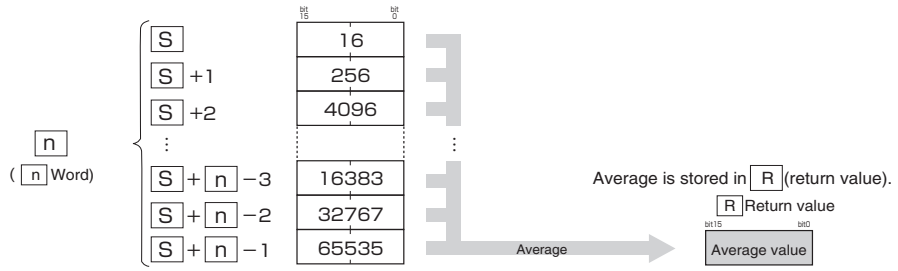
- **The argument devices ".F" and ".DF" can be used in KV STUDIO Ver. 9.0 or later only.**
- **The argument devices ".F" and ".DF" can be specified with the KV-8000 series CPU units and KV-7000 Series CPU units that have CPU function version 2.0 or later only.**
- **The return value device ".DF" can be used in KV STUDIO Ver. 6.00 or later only.**
- **The return value device ".DF" can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or later, and with KV Nano Series base units.**

"KV-7500/7300 CPU Function Version", page 3
"KV-5500/5000/3000 CPU Function Version", page 4

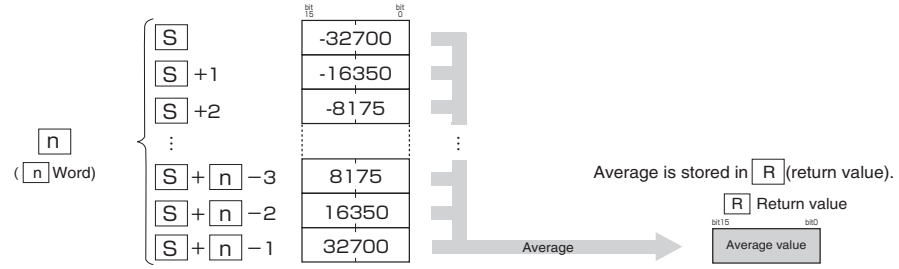
Description of Operation

The average value of all BIN data (16-bit/32-bit (unsigned/signed)/64-bit) stored in S words is calculated starting from n. (When the operation destination is 32-bit data, the operation range becomes the "specified number of data x 2 words" and when 64-bit data it becomes the "specified number of data x 4 words".)
The average value as a result of the search is stored in R.

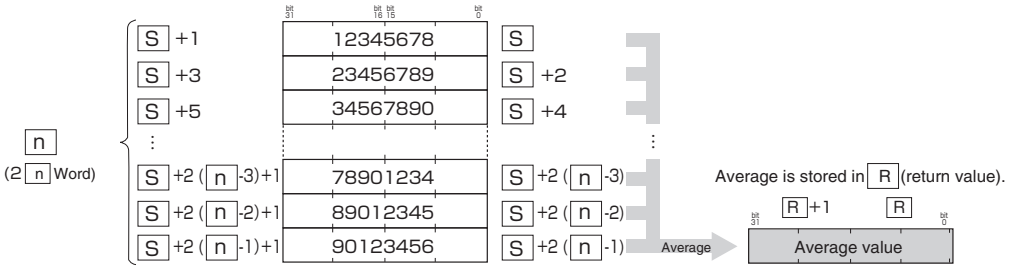
- S : When the operation destination device is unsigned 16-bit data (.U)



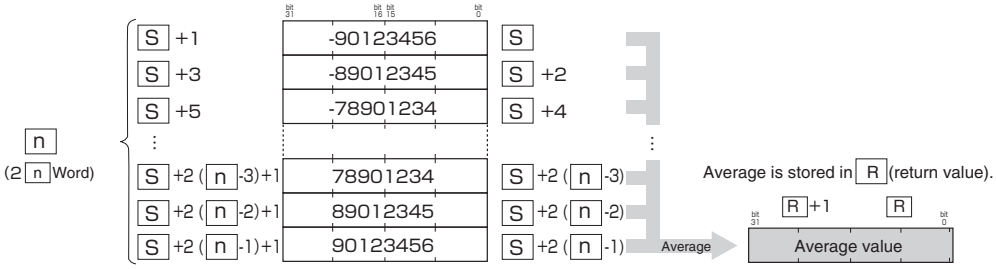
- S : When the operation destination device is signed 16-bit data (.S)



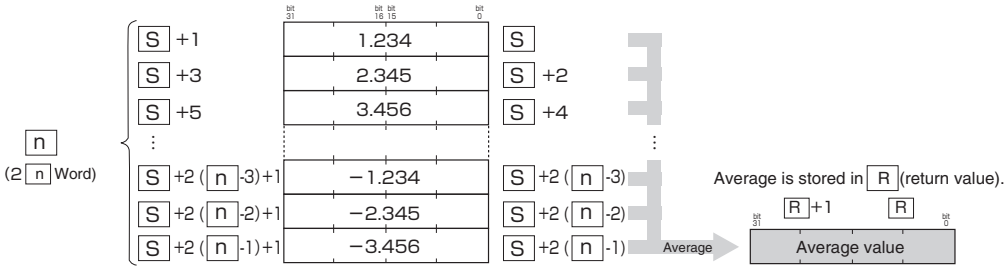
- S : When the operation destination device is unsigned 32-bit data (.D)



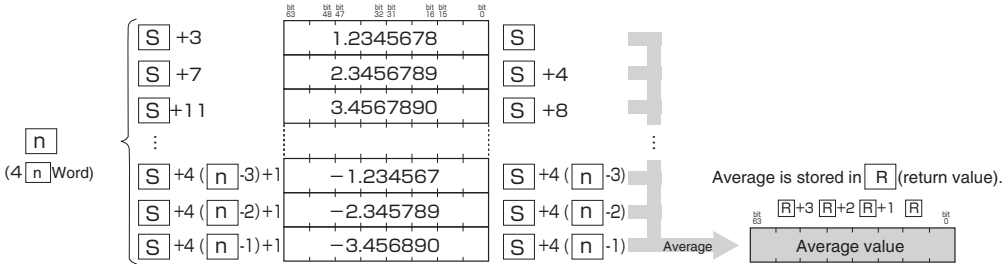
- S : When the operation destination device is signed 32-bit data (.L)



- S : When the search destination device is a single precision floating point type (.F)



- S : When the search destination device is a double precision floating point type (.DF)



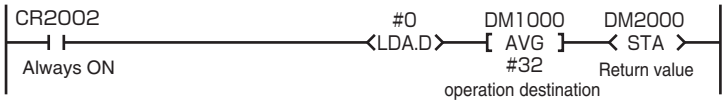
"About operation flag (CR2009 to CR2012)", page 1-18

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

For details on execution timing, see 📖 KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions."

● Example of format

DM2000.U = **AVG** (DM1000.U, 32)



Point **n When the value of R is 0, 0 is stored in :**

MEMO

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

WSUM Total value of specified block

Return value *1 = WSUM (operation destination device, number of operation range data)

Word sum

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S	Operation destination device*2*3*4	Specifies the start of the range to be summed.	.U	.U	.D	.D	.F	*6 .DF	—	—	—	○	—
n	Number of operation range data*2	Specifies the device currently storing the number of data in the range to total, or the total range data.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	.L type WSUM function available.*5	.U	.S	.D	.L	.F	*6 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 The range of numerical values that can be handled differs according to the instruction suffix.
*3 When a bit device has been specified, 16/32/64 continuous bits are handled. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16/32/64 bits straddling the next channel.
*4 When S type is .F or .DF, T and C cannot be specified.
*5 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.
*6 The ".DF" suffix cannot be specified for device Z.

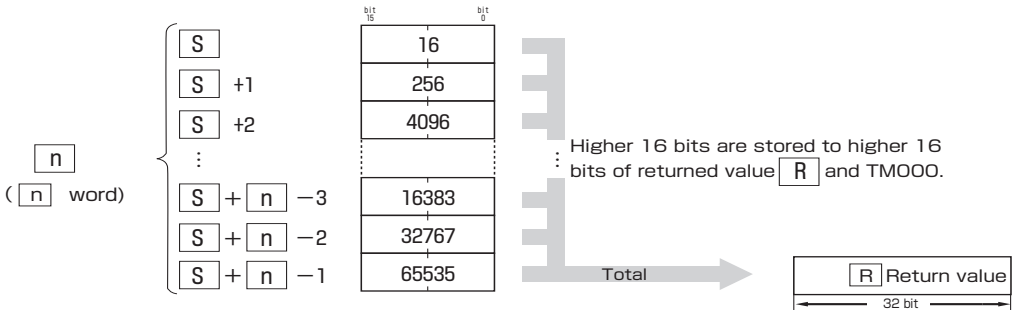
Point

- **The argument devices ".F" and ".DF" can be used in KV STUDIO Ver. 9.0 or later only.**
- **The argument devices ".F" and ".DF" can be specified with the KV-8000 series CPU units and KV-7000 Series CPU units that have CPU function version 2.0 or later only.**
- **The return value device ".DF" can be used in KV STUDIO Ver. 6.00 or later only.**
- **The return value device ".DF" can only be specified with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or later, and with KV Nano Series base units.**
  "KV-7500/7300 CPU Function Version", page 3
  "KV-5500/5000/3000 CPU Function Version", page 4

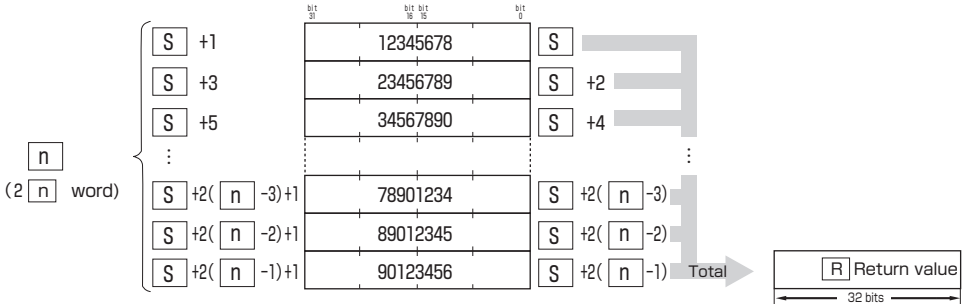
Description of Operation

S The total value of all BIN data (16-bit/32-bit (unsigned)/64-bit) stored in the range of words specified by is calculated with the device specified by n as the leading device. (When the operation destination is 32-bit data, the operation range becomes the "specified number of data x 2 words" and when 64-bit data it becomes the "specified number of data x 4 words") The operation results and total value are stored in R.

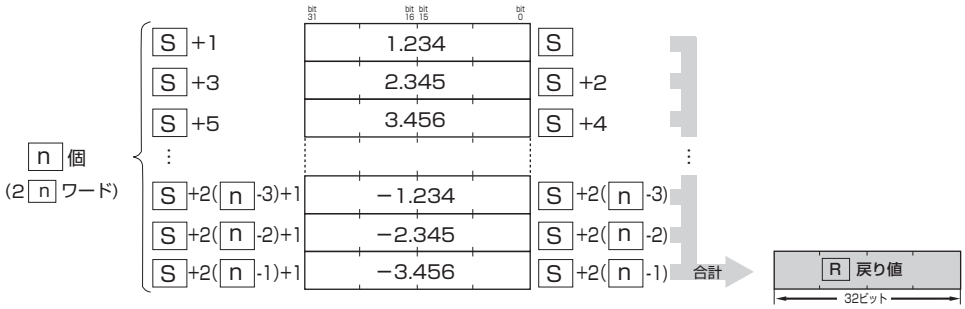
- S : When operation destination is 16 bits (.U/.S)



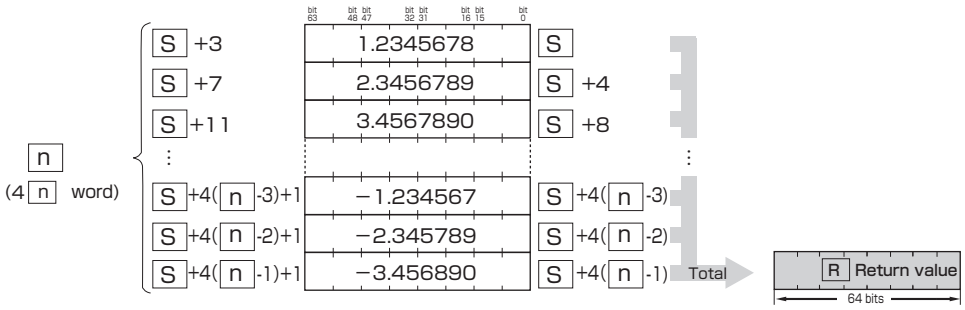
- S : When operation destination is 32 bits (.D/.L)



- S : When the search destination device is a single precision floating point type (.F)



- S : When the search destination device is a double precision floating point type (.DF)

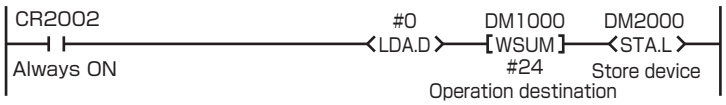


"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions."

● Example of format

DM2000.L = **WSUM** (DM1000.U, 24)



Reference

When the operation result exceeds the range (−2147483648 to +2147483647) of signed 32-bit data, the lower 32-bit data is stored in return value, and the upper 32-bit data is stored in TM001 and TM000.

Point

n When the value of R is 0, 0 is stored in :

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

Sample Program

● When this function is used in combination with other functions

Programmed Script

DM2000.F = FLOAT (**WSUM** (DM1000.U, EM1100.U))

Operation Description

Calculate the total of all data in consecutive EM1100.U devices with DM1000.U as the head device, and convert the operation result into a single-precision floating-point real number, and store it in DM2000.F.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "FLOAT function", page 7-114
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, step 5, (page 2-10)

Hint

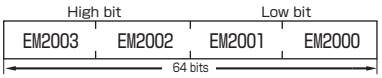
When the result of having performed an operation by the WSUM function exceeds the signed 32-bit range, the lower 32-bit data is stored in return value, and the upper 32-bit data is stored in temporary data memories TM0 and TM1.
The WSUM uses the upper 32-bit (TM0/TM1) data to handle data as 64-bit data.

Programmed Script

EM2000.L = **WSUM** (DM1000, 1024)

EM2002.L = TM0000.L

All of unsigned 16-bit BIN data stored in range of 1024 words starting from DM1000 is totaled. The lower 32 bits of the total value are stored in EM2000.L, and then the upper 16 bits (TM0.L) are immediately stored in EM2002.L to be handled as signed 64-bit BIN data from EM2000.L to EM2002.L.



Reference

The temporary data of TM0/TM1 is used also in other operation formulas. For this reason, when data is to be handled as 64-bit data, the data must be stored in a separate device immediately after the WSUM has been programmed.

MEMO

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

BSUM Total value (in byte units) of specified block

Return value *1 = BSUM (operation destination device, number of operation range data)

Byte sum

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	For mula
S	Operation destination device*2*3*4	Specifies the start of the range to be summed.	.U	.U	.U	.U	—	—	—	.U	—	○	—
n	Number of operation range data*2	Specifies the number of data (in byte units) to calculate the total value.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	.D type BSUM function available.*5	.U	.S	.D	.L	.F	*6 .DF	—	—	—	○	—

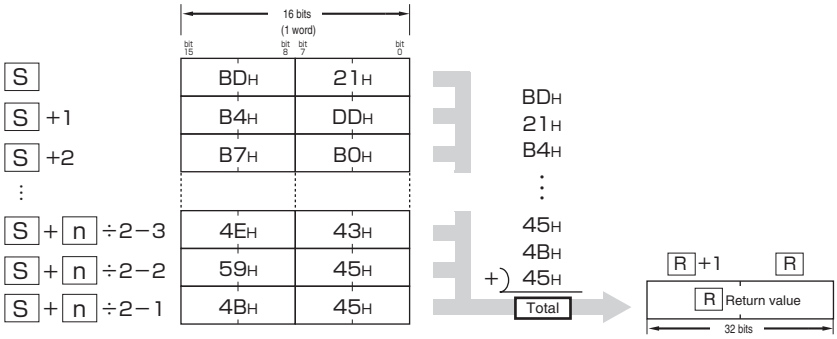
*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 The range of numerical values that can be handled differs according to the instruction suffix.
*3 When a bit device has been specified, 16/32 continuous bits are handled. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16/32 bits straddling the next channel.
*4 T, C, CTC, CTH, Z cannot be used.
*5 When the type other than .D is specified by return value, type conversion will be automatically conducted during conversion.
*6 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument device ".DF" can be used only with the KV STUDIO Ver. 6.00 and higher versions.**
- **The argument device ".DF" can only be specified with KV-8000/7000 Series CPU units, with KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

The total values of each of the lower and upper eight bits of the BIN data currently stored in the specified range starting from the S are calculated, and are stored in R. When n is an odd number, up to the upper eight bits of the last device are calculated. Calculation is performed in order upper to lower bits, and when n is an odd number, the lower eight bits of the last word data are ignored.

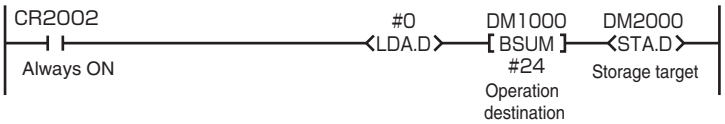


"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions."

● **Example of format**

DM2000.D = **BSUM** (DM1000, 24)



Reference When the operation result exceeds the signed 32-bit data range, the lower 32-bit data is stored in return value, and the upper 32-bit data is stored in TM001 and TM000.

Point

When either of the following conditions is established, 0 is stored in [R]:

- **When a timer/counter is specified to [S]**
- **[S] When a device other than the leading device of the channel is specified when a bit device is specified to**
- **When the value of [n] is "0"**

Sample Program

● **When this function is used in combination with other functions**

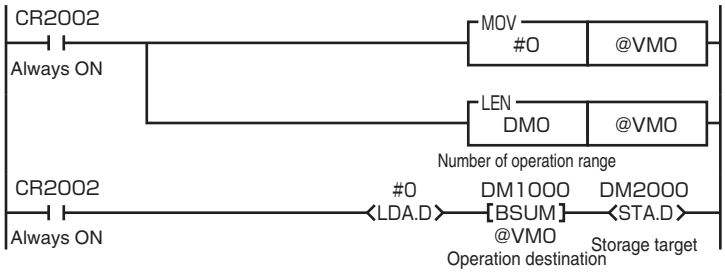
Programmed Script

DM2000.D = **BSUM** (DM1000.T, LEN (DM0000.T))

Operation Description

The ASCII text string currently stored in DM1000.T is totaled in byte units, and the result is stored in [DM2000,DM2001].

Ladder conversion



Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "LEN function", page 7-182
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

CRC

Calculate CRC value in specified block

Return value *1 = CRC (default, operation method, operation order, invert bit operation order, data block leading device, number of data)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	For mula
S1	Default*5	Specifies the operation mode.*2	.U	.U	.U	.U	—	—	—	—	○	○	○
S2	Operation method*5		.U	.U	.U	.U	—	—	—	—	○	○	○
S3	Operation order*5		.U	.U	.U	.U	—	—	—	—	○	○	○
S4	Invert bit order*5		.U	.U	.U	.U	—	—	—	—	○	○	○
S5	Data block leading device*3*5	Specifies the leading device of the data block to calculate.	.U	.U	.U	.U	—	—	—	—	—	○	—
n	Number of data*4	Specifies the number of data (1- 65535).	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	.U type CRC function available*6	.U	.S	.D	.L	.F	*7 .DF	—	—	—	○	—

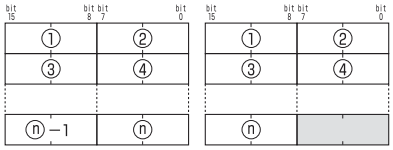
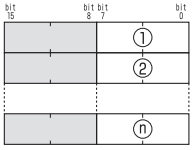
*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.
*2 When the specified device or the value of the formula is other than "0" or "1", the value is handled as "1".
*3 When specifying a bit device, specify the leading device of the channel.
*4 S3 When is "0", specify the number in 1-word (16-bit) units.
S3 When is "1 (other than "0")", specify the number of 1-byte (8-bit) units.
*5 T, C, CTC, CTH, Z cannot be used.
*6 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.
*7 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument device ".DF" can be used only with the KV STUDIO Ver. 6.00 and higher versions.**
- **The argument device ".DF" can only be specified with KV-8000/7000 Series CPU units, with KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

The CRC value of S1 S4 data starting from S5 is calculated in the operation mode specified by to n operation data bit order inversion, and is stored in R.
The operation mode is specified as follows.

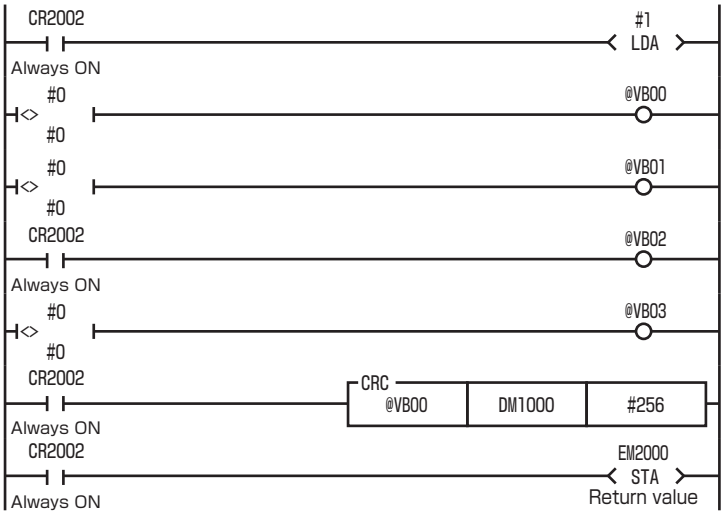
Item		CRC Operation Method: When "1" is specified	When "0" is specified
S1	CRC default	The value set to 16-bit internal register is used as the CRC default.	The CRC default is fixed at 0x0000.
S2	CRC Operation Method	CRC-ITU-T(CRC-CCITT)	CRC-16(CRC-ANSI)
S3	Order of operation	Operation is performed in order upper 8 bits -> lower 8 bits. n : Even  n : Odd number 	Operation is performed only on the lower 8 bits. (The upper 8 bits are ignored.) 
S4	Operation data bit order inversion	The bit alignment (order) of data to be operated on is inverted in byte units, and the CRC value is calculated.	The bit position is not inverted.

S4 The following example illustrates the bit order inversion of operation data (in bytes).

Operation Data Bit Order Inversion	CRC-ITU-T (S2 =1)	CRC-16 (S2 =0)
Not inverted S4 :「0」	bit 7 — Operating order → bit 0	bit 7 ← Operating order — bit 0
Inverted S4 :"1" (other than 0)	bit 7 ← Operating order — bit 0	bit 7 — Operating order → bit 0

● Example of format

EM2000.U = **CRC** (1, 0, 0, 0, DM1000.U, 256)



"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions", CRC Instruction.

Point

When either of the following conditions is established, the value of R is stored in S1:

- **When a word device or timer/counter has been specified to S1**
- **When a device other than the leading device of the channel is specified when a timer/counter or a bit device is specified to S2 by indirect specification**
- **When the value of n is "0"**

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

ZRES Reset specified block

Zone reset

ZRES ([execution condition,][*1] leading device, final device)

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D1	Leading device	Specifies the start of the range to be reset (0 is stored).	.U	.U	.U	.U	.U	*2 .U	.B	.T	—	○	—
D2	Final device	Specifies the final device of the range to reset.	Handled as same type as D1.								—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 [ ] can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 The ".DF" suffix cannot be specified for device Z.

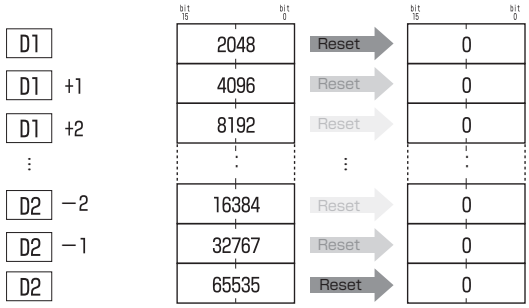
Point

- **The argument device ".DF" can be used only with the KV STUDIO Ver. 6.00 and higher versions.**
- **The argument device ".DF" can only be specified with KV-8000/7000 Series CPU units, with KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

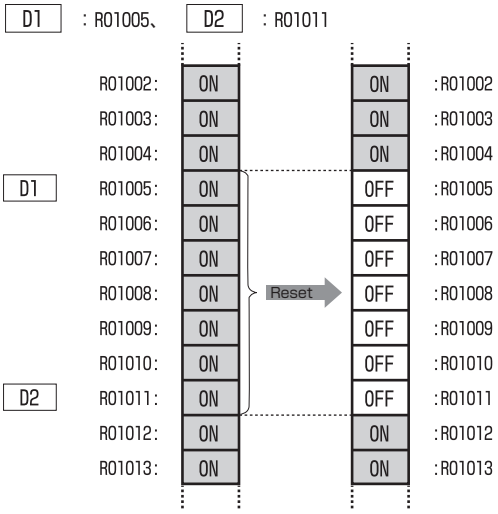
When the execution condition is established (TRUE), all of the values of D1 and D2 are reset.

- To specify the word device



* The suffix is handled as .U even if .D .L .F .DF are specified as the suffix. For this reason, "0" is stored for the specified one word as the final word.

- To specify the bit device



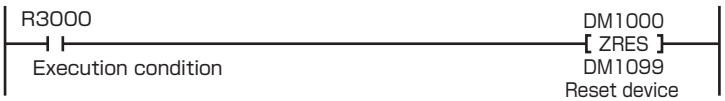
Reference When a word device has been specified, the word becomes "0" in word units, and when a bit device has been specified, the bit becomes OFF in bit units.

"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions."

● Example of format

ZRES (R3000, DM1000.U, DM1099.U)



Point **When the type of specified devices differ at D2 and D1, the ZRES function is not executed.**

Sample Program

● To specify the bit device

Programmed Script **ZRES** (R3000, MR1008, MR1207)

Operation Description When R3000 turns ON, all of the 32 bits from M1008 to MR1207 are reset (are turned OFF).

● When an index register is used

Programmed Script **ZRES** (DM2000:Z01, DM2000:Z02)

Operation Description All of the words in the range DM2000:Z01 to DM2000: Z02 are reset (cleared to "0") at every scan (no execution condition).

Reference The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

BCMP Compare data between specified blocks

Return value *1 = BCMP (compare source device, compare destination device, operation store device, number of compare data)

Compare data between blocks

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	For mula
S1	Compare source device*2	Specifies the leading device of the compare source data block.	.U	.U	.D	.D	.D	—	—	—	—	○	—
S2	Compare destination device*2	Specifies the leading device of the compare destination data block.	Handled as same type as S1.					—	—	—	—	○	—
S3	Result store device*2*3	Specifies the start to store the comparison result.	—	—	—	—	—	—	.B	—	—	○	—
n	Number of compare data	Specifies the device currently storing the number of data of the block to compare, or the number of compare data.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	ON when all data matched. Otherwise, OFF	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work device.

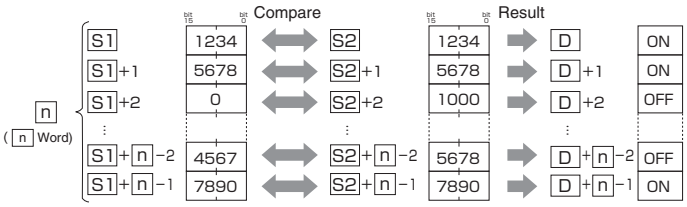
*2 When a bit device has been specified, 16/32 continuous bits are handled. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16/32 bits straddling the next channel.

*3 When a bit device has been specified, the program will function normally even if the bit device straddles channels.
When a word device has been specified to D, the LSB of n number of 16-bit word devices becomes the store destination of the result. Bits other than the lowermost bit do not change.

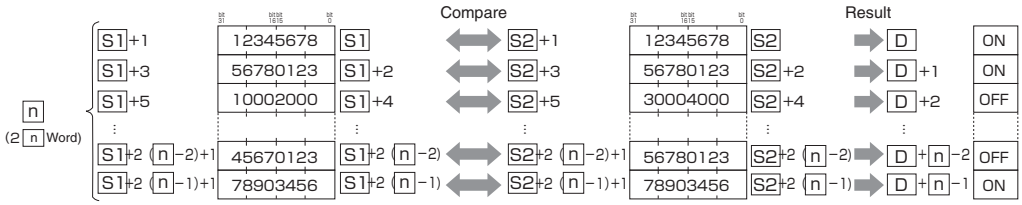
Description of Operation

When the execution condition is established (TRUE), S1 block data starting from n and S2 block data starting from n are each compared.
When the compare result matches, ON is stored in bit devices for the number of compare data starting from D . When the compare result does not match, OFF is stored.
D When the is a word device, storage of the compare result is handled in 16-bit word device units, and only the LSB of each word device is targeted. (Bits other than the LSB do not change.) (Bits other than the LSB do not change.)
If all data matches as a result of the compare, R are turned ON. Otherwise, the bit devices are turned OFF.

- S1 : When the compare source device is 16 bits (.U/.S)



- S1 : When the compare source device is 32 bits (.D/.L)

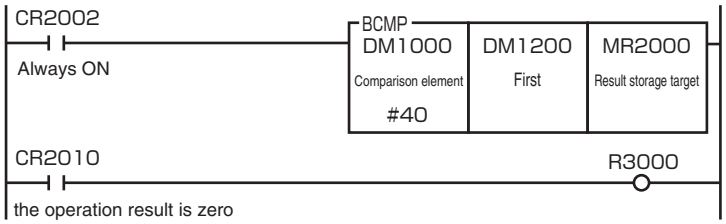


"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions."

● Example of format

R3000 = **BCMP** (DM1000.U, DM1200.U, MR2000, 40)



Point **When the value of n is 0, R turns ON.**

Sample Program

● When this function is used in combination with other functions

Programmed Script

MR2000 = **BCMP** (DM1000.L, EM2000.L, MR3000, ANDA (EM3000, $3F))

Operation Description

Of signed 32-bit BIN data block starting from DM1000.L and signed 32-bit BIN data block starting from EM2000 (.L), the data of the number of compare data (number of compare data x 2 words) indicated by the lower six bits (extracted) of EM3000 are each compared.
The compare result is stored in 16-bit data starting from MR3000.
If all of the data is matching as a result of the compare, MR2000 is turned ON. Otherwise, MR2000 is turned OFF.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "ANDA function", page 7-22
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

BCMPI Compare data of value with block

Return value *1 = BCMPI (compare source device, compare value device, result store device, number of compare data)

Compare data between blocks

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S1	Compare source device*2	Specifies the leading device of the data block to compare.	.U	.U	.D	.D	.D	—	—	—	—	○	○
S2	Compare value device	Specifies the target to store the value to be compared or the comparison.	Handled as same type as S1.					—	—	—	—	○	—
D	Result store device*3	Specifies the start to store the comparison result.	—	—	—	—	—	—	.B	—	—	○	—
n	Number of compare data	Specifies the device currently storing the number of data of the block to compare, or the number of compare data.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	ON when all data matched. Otherwise, OFF	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work device.

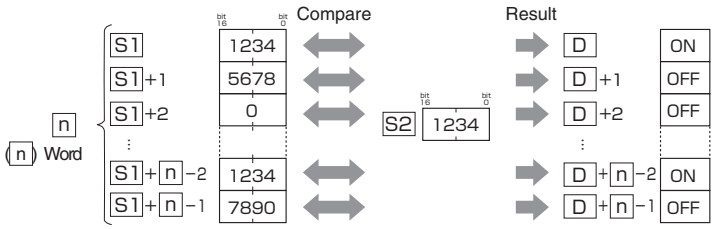
*2 When a bit device has been specified, 16/32 continuous bits are handled. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

*3 When a bit device has been specified, the program will function normally even if the bit device straddles channels.
When a word device has been specified to D, the LSB of n number of 16-bit word devices becomes the store destination of the result. Bits other than the lowermost bit do not change.

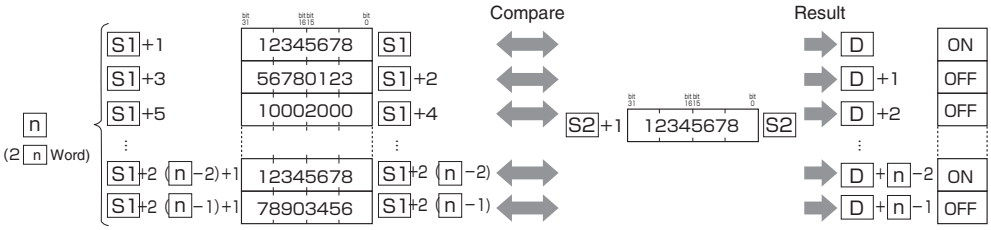
Description of Operation

When the execution condition is established (TRUE), S1 block data starting from n and the data in S2 are each compared.
When the compare result matches, ON is in D bit devices starting from n. When the compare result does not match, OFF is .
When the D is a word device, storage of the compare result is handled in 16-bit word device units, and only the LSB of each word device is targeted. (Bits other than the LSB do not change.) (Bits other than the LSB do not change.)
If all data matches as a result of the compare, R are turned ON. Otherwise, the bit devices are turned OFF.

- S1 : When the compare source device is 16 bits (.U/.S)



- S1 : When the compare source device is 32 bits (.D/.L)

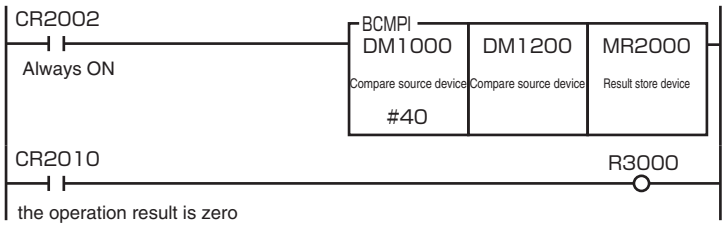


"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions."

● Example of format

R3000 = **BCMPI** (DM1000.U, DM1200, MR2000, 40)



Point **When the value of n is 0, R turns ON.**

Sample Program

● When this function is used in combination with other functions

Programmed Script

MR2000 = **BCMPI** (DM1000.D, RRNCA(MR1000.D, 16), MR3000, 24)

Operation Description

The unsigned 24*32-bit BIN data blocks starting from DM1000.D are each compared with the value resulting from having rotated (upper/lower words swapped) the unsigned 32-bit BIN data currently in MR1000.D 16 bits to the right without a carry.
The compare result is stored in 16-bit data starting from MR3000.
If all of the data is matching as a result of the compare, MR2000 is turned ON. Otherwise, MR2000 is turned OFF.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "RRNCA function", page 7-50
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

RND Generate random number

Return value *1 = RND (type of random number)

Random

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Type of random number*2	Specifies the operation destination device of generated random number or the operation destination data.	.U	.U	.U	.U	—	—	—	—	—	○	○
R	Return value	.U type RND function available.*3	.U	.S	.D	.L	.F	*4 .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in working memory.

*2 When (R1002, R1005, etc.) other than initial channel of the bit device is specified for category of random number (conversion destination device), jump the following channel can also exert normal function.

*3 When the type other than .U is specified by return value, type conversion will be automatically conducted during conversion.

*4 The ".DF" suffix cannot be specified for device Z.

Point

- **The argument device ".DF" can be used only with the KV STUDIO Ver. 6.00 and higher versions.**
- **The argument device ".DF" can only be specified with KV-8000/7000 Series CPU units, with KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

S generates the category of random number (operation destination device) to random number R.

The random number occurring is analog random number of 1- 65535. S When = 0, R stores 1 in return value.

"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions."

Point

- **Random number generation algorithm shall use "M series".**
- **Due to periodicity of value generated from RND function (for example, 2 generated after 1), so, when category of S random number is not changed, the value of random number generated will be not changed.**

● Example of format

DM1000.U = **RND** (DM1000.U)

MEMO

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

SORT Binary data sort

Sort

SORT ([execution condition,][*1] sorting order, data table leading device, number of data items to sort)

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Sorting order	Specifies the sorting order. (0: ascending, 1: descending)	.U	.U	.U	.U	—	—	—	—	○	○	○
D	Data table leading device[*2]	Specifies the leading device of the byte sequence to search for.	.U	.S	.D	.L	—	—	—	—	—	○	—
n	Number of data items to sort[*3]	Specifies the number of data items to sort.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 [ ] can be omitted. If the execution condition is omitted, the instruction is always executed (it is executed every scan).

*2 When a bit device is specified, D channel is occupied if the n type is .U/.S, and 2x n channels are occupied if the type is .D/.L.
When a word device is specified, D word is occupied if the n type is .U/.S, and 2x n words are occupied if the type is .D/.L.

*3 When a bit device is specified, 16 continuous bits are occupied. When a word device is specified, one word is occupied.

Point

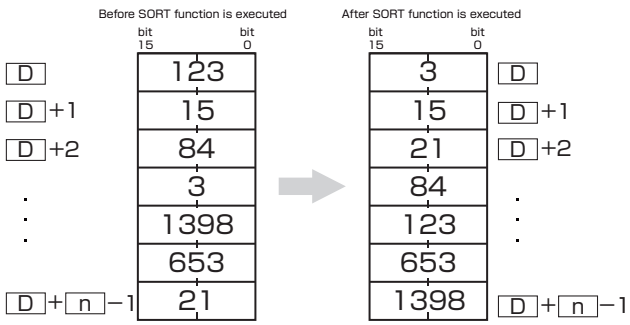
- **The SORT function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, with KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

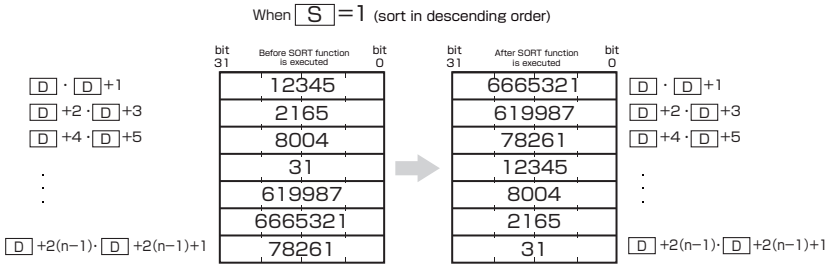
The D number of binary data items to sort in the data table specified with n data table leading device is sorted with the sorting order specified in S sorting order.

- D : data table leading device is 16 bits (.U/.S)

When S =0 (sort in ascending order)



- [D] : data table leading device is 32 bits (.D/.L)



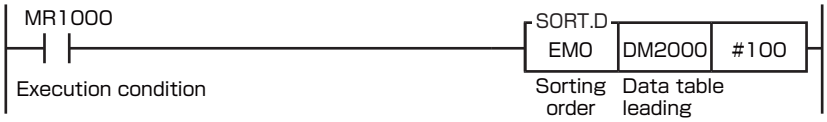
Point

The SORT function sorts all data when the function is executed. Therefore, if a large value is specified in [n], the scan time may increase.
If the effect onto the scan time needs to be adjusted, divide the data to be sorted into several scans using the SORTN function.
"SORTN function", page 8-44

"About operation flag (CR2009 to CR2012)", page 1-18
For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions."

● Example of format

SORT (MR1000, EM0, DM2000.D, 100)



SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

SORTN Binary data partition sort

Return value *1 = SORTN (sort control flag, sort order, number of comparison times, data table leading device, number of data items to sort)

Partition sort

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S1	Sort control flag	Specifies the sort processing.	—	—	—	—	—	—	.B	—	—	○	—
S2	Sorting order	Specifies the sorting order. (0: ascending, 1: descending)	.U	.U	.U	.U	—	—	—	—	○	○	○
S3	Number of comparison times*3	Specifies the number of comparisons to be made in one scan. (1 to 65535)	.U	.U	.U	.U	—	—	—	—	○	○	○
D	Data table leading device*2	Specifies the leading device of the byte sequence to search for.	.U	.S	.D	.L	—	—	—	—	—	○	—
n	Number of data items to sort*3	Specifies the number of data items to sort.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	This device turns OFF during sorting, and turns ON when sorting is completed.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the bit information is stored in a work device.

*2 When a bit device is specified, D channel is occupied if the n type is .U/.S, and 2x n channels are occupied if the type is .D/.L.
When a word device is specified, D word is occupied if the n type is .U/.S, and 2x n words are occupied if the type is .D/.L.

*3 When a bit device is specified, 16 continuous bits are occupied. When a word device is specified, one word is occupied.

Point

- **The SORTN function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can only be specified with KV-8000/7000 Series CPU units, with KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

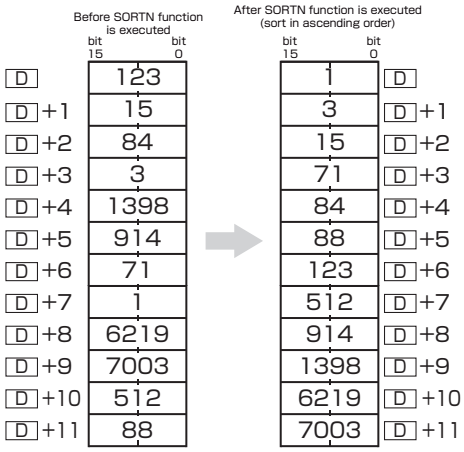
Description of Operation

The D number of binary data items to sort in the data table specified with n data table leading device is sorted with the sorting order specified in S2 sorting order.
The number of comparisons made per scan is specified with S3 number of comparison times.

Example

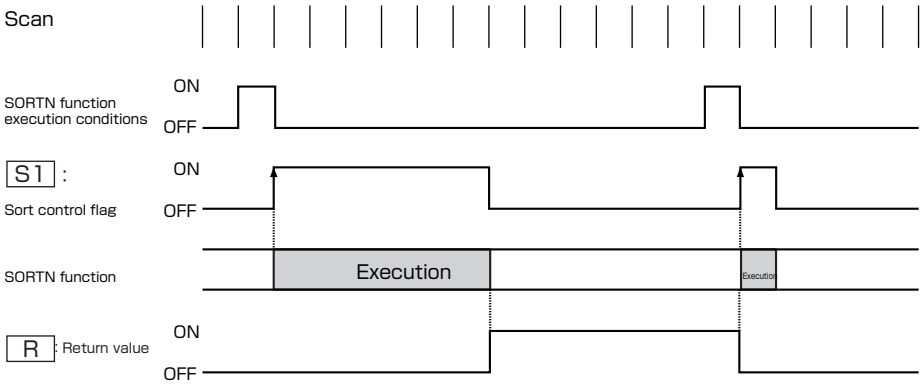
D : data table leading device is 16 bits (.U/.S)

When S2 =0、S3 =3、n data table leading device is 16 bits (.U/.S)



Three comparisons are made per scan, and the data is sorted over multiple scans.
R =ON is written when sorting is completed.

The SORTN function execution timing is shown below.



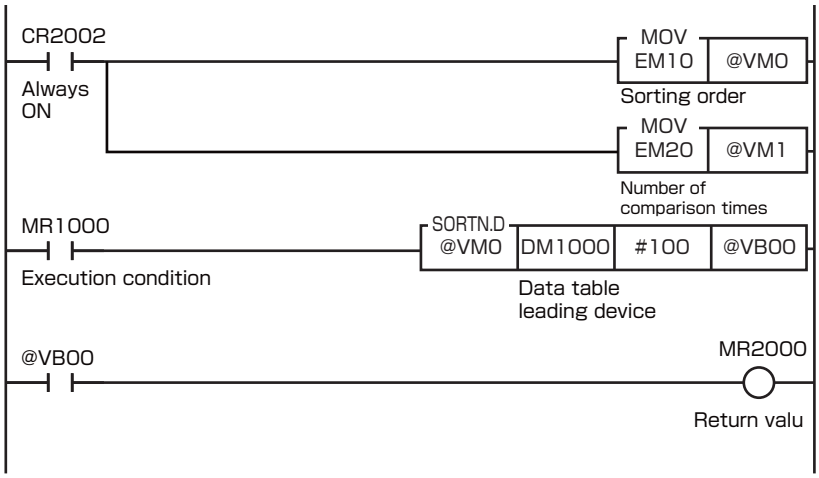
"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Data Processing Instructions."

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

● **Example of format**

MR2000 = **SORTN** (MR1000, EM10, EM20, DM1000.D, 100)



MEMO

SEG
BCNT
DCNT
SER
DSER
MAX
MIN
AVG
WSUM
BSUM
CRC
ZRES
BCMP
BCMPI
RND
SORT
SORTN

FIFOW Write to FIFO table

FIFO write

FIFOW ([execution condition,][*1] data store device, data table leading device)

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Data store device[*2]	Specifies the device to which the data to write to the FIFO data block is currently stored.	Handled as same type as D.					—	—	—	—	○	○
D	Data table leading device	Specifies the leading device of the FIFO data table.	.U	.U	.D	.D	.D	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 [ ] can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

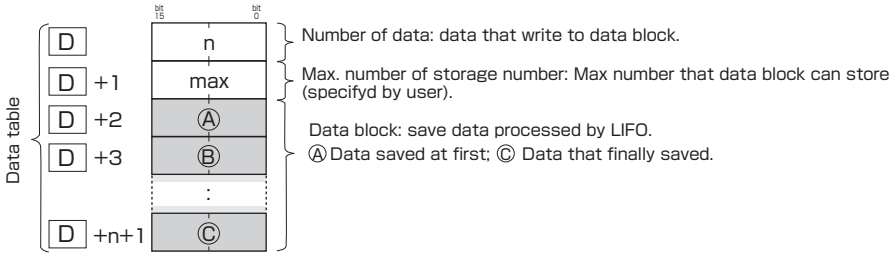
*2 When a bit device has been specified, 16/32 continuous bits are handled. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16/32 bits straddling the next channel.

Description of Operation

When the execution condition is established (TRUE), S the data (16-bit/32-bit) stored in D is written to the end of the data block specified by the .

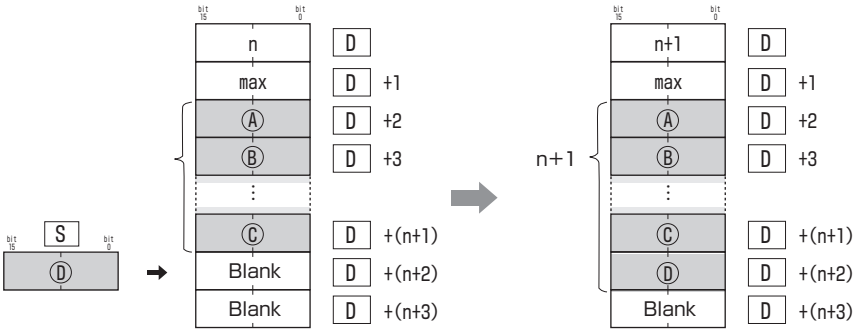
● About data table and data block

The FIFOW and FIFOR instructions FIFO (First In First Out) process using a group of devices called a "data table". The figure below shows the structure of a data table.

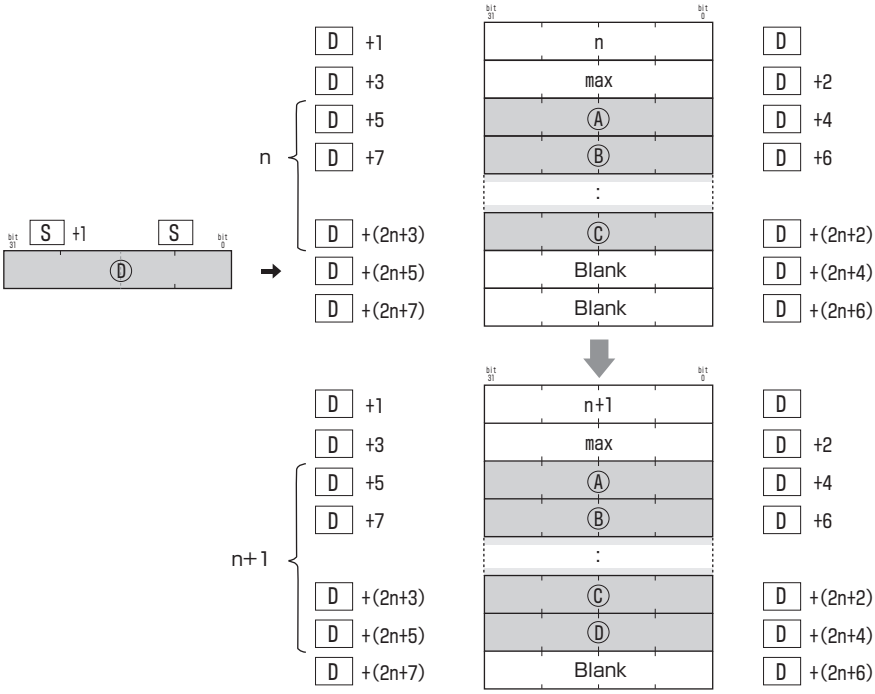


- Before executing this function, reset (store "0") the number of data (n) to be stored in D. Also, set the maximum number of stored data items to D+1.
- As the leading device D of the data table, specify the same device as leading device of the data table for the corresponding FIFOR function.

- S : When the data store device is 16 bits (.U/.S)
  - The data store destination becomes D +(n+2).
  - n is incremented by 1 after storage.



- S : When the data store device is 32 bits (.D/.L/.F)
  - The data store destination becomes D +(2n+4) • D +(2n+5).
  - n is incremented by 1 after storage.

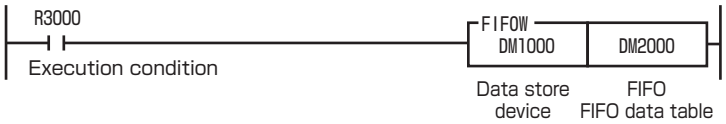


"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Table Processing Instructions."

● **Example of format**

FIFOW (R3000, DM1000, DM2000.U)



Point

The function is not executed when either of the following conditions is established:

- **When maximum number of stored data is "0"**
- **When the number of data n exceeds the maximum number of stored data.**
- **When the maximum number of stored data is a number that causes the data table to exceed the device range**

Sample Program

● **When an index register is used**

Programmed Script

FIFOW (R3000, DM1000, DM2000:Z1)

Operation Description

When R3000 turns ON, the 16-bit data currently stored in DM1000 is written to the end of the data block starting from DM (2000+Z1).

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

Hint

The "number of currently stored data" of the data table specified by the data table leading device is saved to the leading device. Also, the "maximum number of stored data" is set to the leading +1 device. Before the FIFOW function is used, the number of currently stored data must be reset ("0" stored), and the maximum number of stored data must be set. (default)

Programmed Script

```
IF LDP ( R3000 )  THEN
   DM1000 = 0
   DM1001 = 10
END IF

FIFOW ( LDP ( R3100 ), DM2000, DM1000 )
```

' The entire data table is cleared when R3000 turns ON.

' The maximum number of stored data is set to 10 when R3000 turns ON.

' Write to the data block is performed when R3100 turns ON.

Operation Description

When the rising edge of R3000 (change from OFF to ON state) has been detected, the data table having ten data blocks is initialized from the 16-bit data table leading device DM1000. (A data table comprises the number of stored data, maximum number of stored data, and data block, and is a total of 12 words.)
When the rising edge of R3100 (change from OFF to ON state) has been detected, the 16-bit data currently stored in DM2000 is written to the end of the data block.

Point

In a 32-bit data table, the number of stored data, maximum number of stored data, and data block are all handled as 32-bit data. For this reason, attention is required to the data table structure (number of required words).

Reference

When the data table is a 32-bit data table, single precision floating point type real number data also can be used as the write data.

FIFOR Read from FIFO table

Read FIFO

FIFOR ([execution condition,]*1 data table leading device, data store device)

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D1	Data table leading device*2	Specifies the leading device of the FIFO data table.	.U	.U	.D	.D	.D	—	—	—	—	○	—
D2	Data store device*2	Specifies the leading device to store data that is read from the FIFO data block.	Handled as same type as D1.					—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 [ ] can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

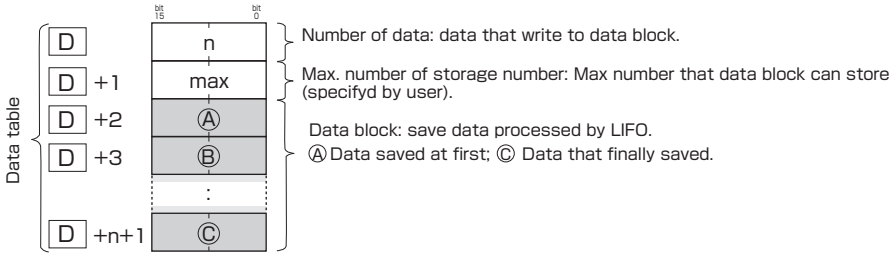
*2 When a bit device has been specified, 16/32 continuous bits are handled. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

Description of Operation

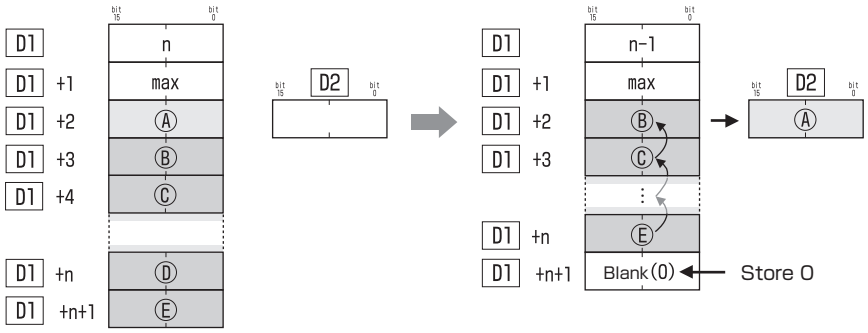
When the execution condition is established (TRUE), the leading data currently stored in data block specified by the D1 is read, and is stored in D2.
After the data is stored, the content of the data block is pushed forward by one data, and the number of stored data is decremented (-1).

● About data table and data block

The FIFOW and FIFOR instructions FIFO (First In First Out) process using a group of devices called a "data table". The figure below shows the structure of a data table.

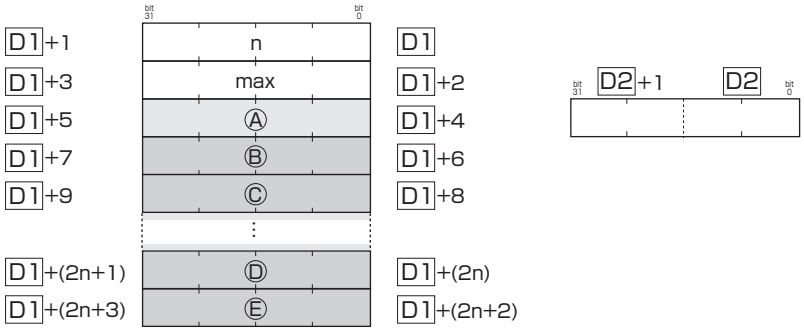


- D1 : When the data table leading device is 16 bits (.U/.S)
  (1) When the execution condition is ON, the 16-bit data currently stored in D1+2 is stored in the device specified by D2.
  (2) After the data is stored, the content of the data block is pushed forward by one data item at a time. n is then decremented (by 1) after the data is pushed.

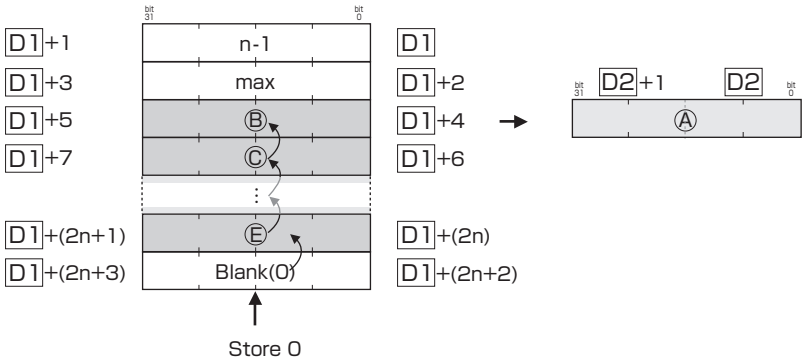


- D1 : When the data table leading device is 32 bits (.D/.L/.F)

  (1) When the execution condition is ON, the 32-bit data currently stored in [D1+4/D1+5] is stored in device specified by [D2/D2+1].



  (2) After the data is stored, the content of the data block is pushed forward by one data item at a time. n is then decremented (by 1) after the data is pushed.

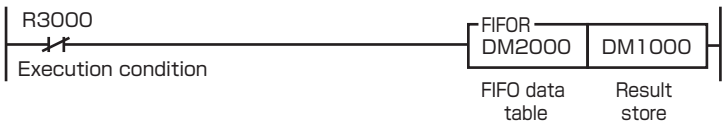


"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Table Processing Instructions."

● Example of format

FIFOR (R3000 = OFF, DM2000.U, DM1000)



Point

The function is not executed when either of the following conditions is established:

- **When there is no data in the data table**
- **When the number of data n exceeds the maximum number of stored data.**
- **When the maximum number of stored data is a number that causes the data table to exceed the device range**

Sample Program

● When an index register is used

Programmed Script

FIFOR (R3000, DM1000.D, DM2000.D:Z1)

Operation Description

When R3000 is turned ON, the start 32-bit data currently stored in data block starting from DM1000.D is read and stored in DM (2000+Z1).D.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

Hint

The data at the start of the data block is read as the data to be read by the FIFOR function.

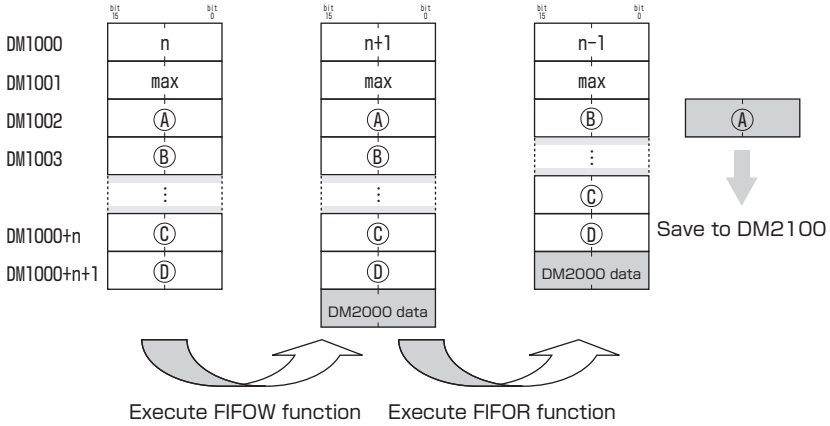
Programmed Script

```
FIFOW ( LDP ( R3100 ), DM2000, DM1000 )      'Write to the data block is performed
                                             'when R3100 turns ON.

FIFOR ( LDP ( R3101 ), DM1000, DM2100 )      'Read from the data block is performed
                                             'when R3101 turns ON.
```

Operation Description

When the rising edge of R3100 (change from OFF to ON state) has been detected, the data stored in DM2000 is written to the data block starting from DM1000.
When the rising edge of R3101 (change from OFF to ON state) has been detected, data is read from the table block and stored in DM2100.



MEMO

LIFOW Write to LIFO table

Write LIFO

LIFOW ([execution condition,]*1 data store device, data table leading device)

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Data store device*2	Specifies the device to which the data to write to the LIFO data block is currently stored.	Handled as same type as D.					—	—	—	—	○	○
D	Data table leading device	Specifies the leading device of the LIFO data table.	.U	.U	.D	.D	.D	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 [ ] can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

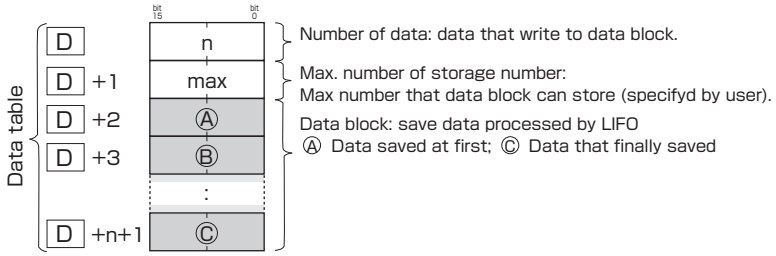
*2 When a bit device has been specified, 16/32 continuous bits are handled. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

Description of Operation

When the execution condition is established (TRUE), the data (16-bit/32-bit) stored in S is written to the end of the data block specified by the D .

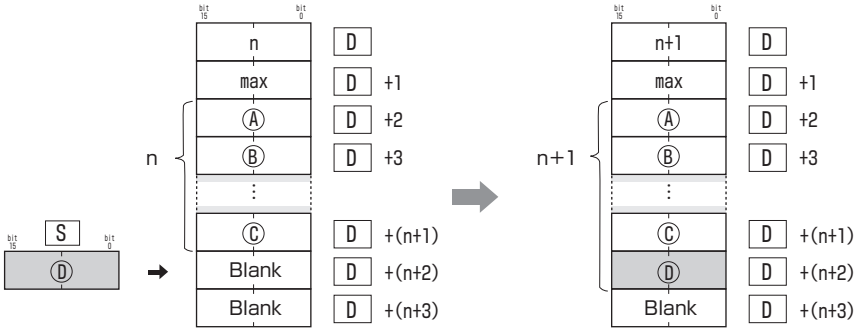
● About data table and data block

The LIFOW and LIFOR functions use a collection of word data called data tables to perform LIFO(Last In First Out) processing. The figure below shows the structure of a data table.

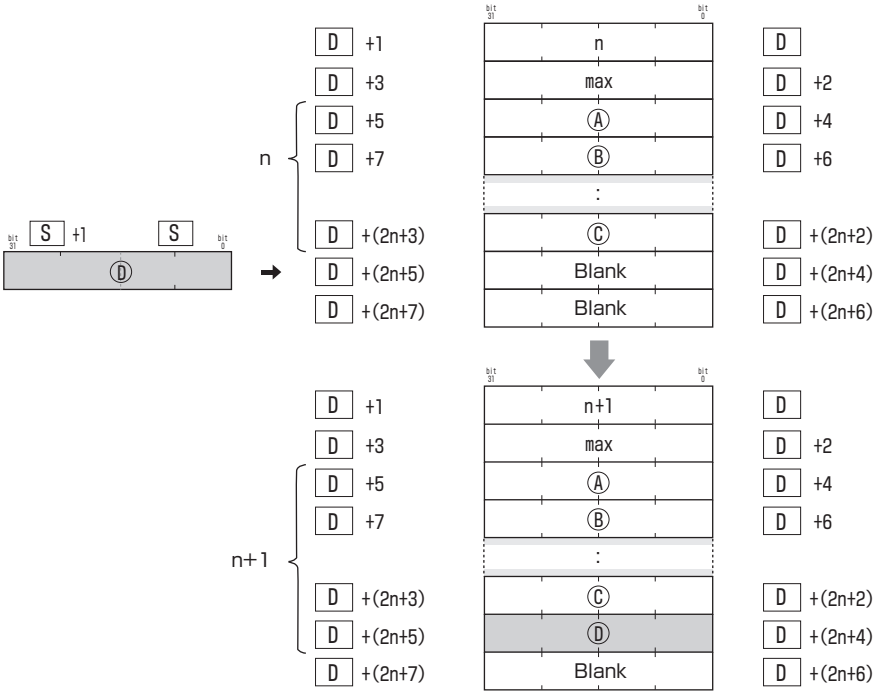


- Before executing this function, reset (store "0") the number of data (n) to be stored in D.
  Also, set the maximum number of stored data items to D+1.
- As the leading device D of the data table, specify the same device as leading device of the data table for the corresponding LIFOR function.

- S : When the data store device is 16 bits (.U/.S)
  - The data store destination becomes D +(n+2)
  - n is incremented by 1 after storage.



- S : When the data store device is 32 bits (.D/.L/.F)
  - The data store destination becomes D +(2n+4) • D +(2n+5).
  - After storage: n plus 1.

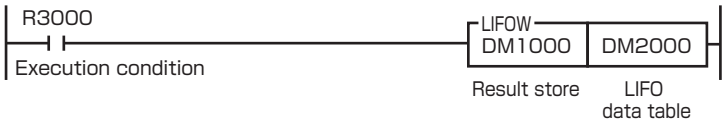


"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Table Processing Instructions."

● Example of format

LIFOW (R3000, DM1000, DM2000.U)



Point

The function is not executed when either of the following conditions is established:

- **When maximum number of stored data is "0"**
- **When the number of data n exceeds the maximum number of stored data.**
- **When the maximum number of stored data is a number that causes the data table to exceed the device range**

Sample Program

● When an index register is used

Programmed Script

LIFOW (R3000, DM1000:Z1, DM2000)

Operation Description

When R3000 turns ON, the 16-bit data currently stored in DM (1000+Z1) is written to the end of the data block starting from DM2000.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

Hint

The data table used by the LIFOW function is in the same structure as the data table used by the FIFOW function, and the same data tables can be used.

*For details, refer to the FIFOW function.

Also, the store method of the LIFOW and FIFOW functions when writing to data blocks is also the same.

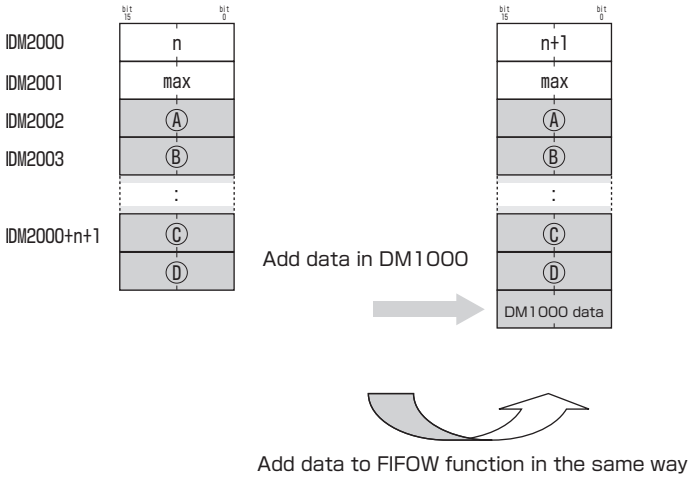
Programmed Script

LIFOW (DM1000, DM2000)

Operation Description

The 16-bit data currently stored in DM1000 is written to the end of the data block starting from DM2000 at every scan (no execution condition).

Ladder conversion



Reference

The same operation as the following Programmed Script is performed:

FIFOW (DM1000, DM2000)

"FIFOW function", page 8-48

MEMO

LIFOR Read from LIFO table

Read FIFO

LIFOR ([execution condition,]*1 data table leading device, data store device)

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D1	Data table leading device*2	Specifies the leading device of the LIFO data table.	.U	.U	.D	.D	.D	—	—	—	—	○	—
D2	Data store device*2	Specifies the leading device to store data that is read from the LIFO data block.	Handled as same type as D1.					—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 [ ] can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

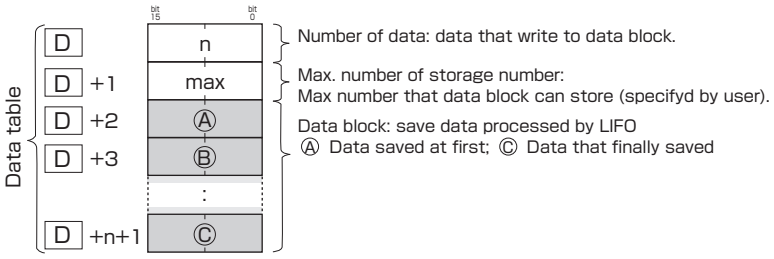
*2 When a bit device has been specified, 16/32 continuous bits are handled. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16/32 bits straddling the next channel.

Description of Operation

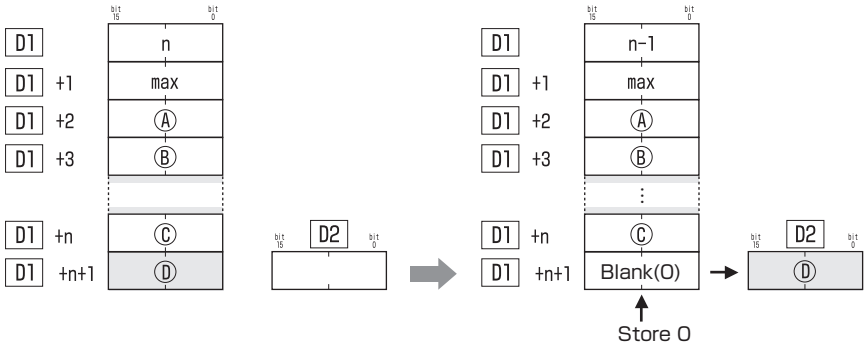
When the execution condition is established (TRUE), the final data currently stored in data block specified by the D1 is read, and is stored in D2.
After the data is stored, the content of the data block is pushed forward by one data, and the number of stored data is decremented (-1).
("0" is stored in last data section that is read.)

● About data table and data block

The LIFOW and LIFOR functions use a collection of word data called data tables to perform LIFO(Last In First Out) processing. The figure below shows the structure of a data table.

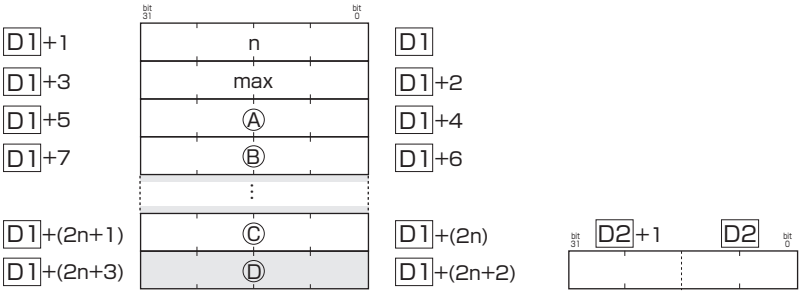


- D1 : When the data table leading device is 16 bits (.U/.S)
  (1) When the execution condition is ON, the 16- bit data currently stored in D1+n+1 is stored in the device specified by D2.
  (2) n is then decremented (by -1) after the data is stored.

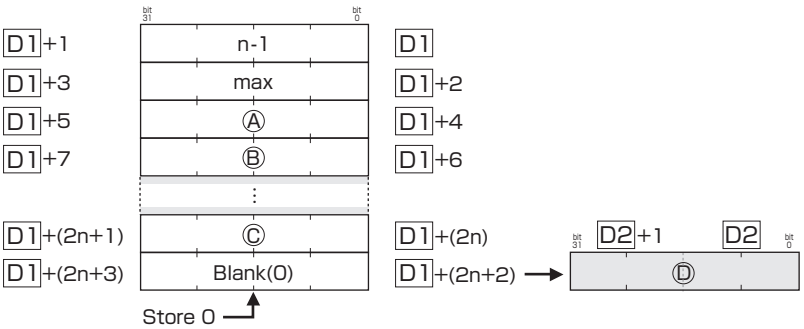


- D1 : When the data table leading device is 32 bits (.D/.L/.F)

  (1) When the execution condition is ON, the 32-bit data currently stored in [D1+(2n+2)/ D1+(2n+3)] is stored in device specified by [D2/D2+1].



  (2) n is then decremented (by -1) after the data is stored.

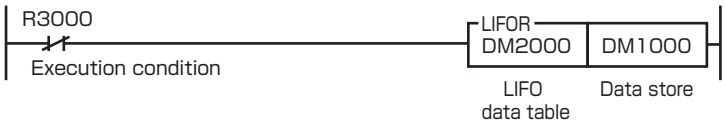


"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Table Processing Instructions."

● Example of format

LIFOR (R3000 = OFF, DM2000.U, DM1000)



Point

The function is not executed when either of the following conditions is established:

- **When there is no data in the data table**
- **When the number of data n exceeds the maximum number of stored data.**
- **When the maximum number of stored data is a number that causes the data table to exceed the device range**

Sample Program

● When an index register is used

Programmed Script

LIFOR (R3000, DM1000.D, DM2000.D:Z1)

Operation Description

When R3000 is turned ON, the final 32-bit data currently stored in data block starting from DM1000.D is read and stored in DM (2000+Z1).D.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

Hint

The data at the end of the data block is read as the data to be read by the LIFOR function.

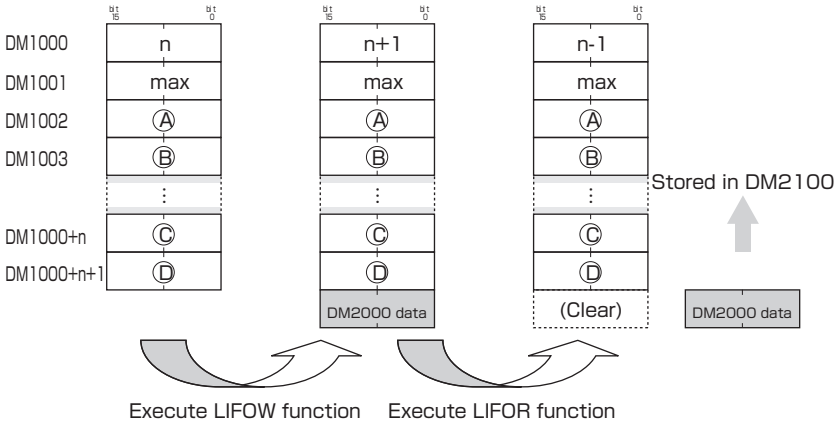
Programmed Script

LIFOW (LDP (R3100), DM2000, DM1000) 'Write to the data block is performed 'when R3100 turns ON.

LIFOR (LDP (R3101), DM1000, DM2100) 'Read from the data block is performed 'when R3101 turns ON.

Operation Description

When the rising edge of R3100 (change from OFF to ON state) has been detected, the data stored in DM2000 is written to the data block starting from DM1000.
When the rising edge of R3101 (change from OFF to ON state) has been detected, data is read from the table block and stored in DM2100.



MEMO

FWRIT Overwrite data table specified position

Rewrite the data table

FWRIT ([execution condition,][*1] data store device, data table leading device, write block position)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	For mula
S	Data store device*2	Specifies the device to which the data to overwrite is currently stored, or the overwrite data.	Handled as same type as D.					—	—	—	—	○	○
D	Data table leading device*2	Specifies the start of the table.	.U	.U	.D	.D	.D	—	—	—	—	○	—
n	Write block position	Specifies the device to which the position of the data block to overwrite is currently stored, or the overwrite position.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

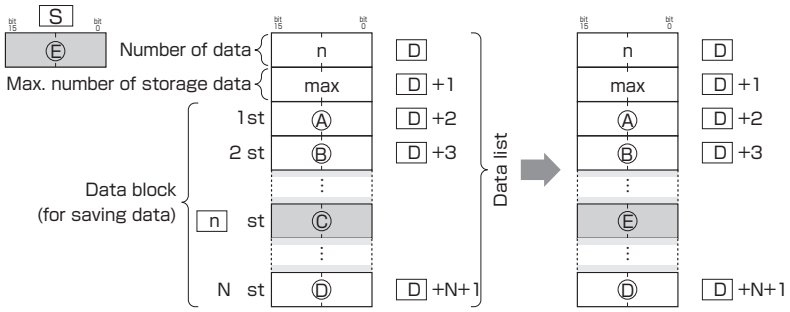
*1 [ ] can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When a bit device has been specified, 16/32 continuous bits are handled. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16/32 bits straddling the next channel.

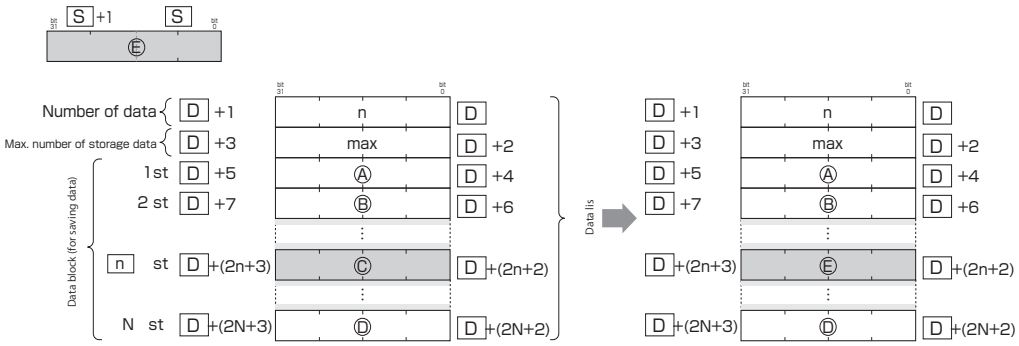
Description of Operation

When the execution condition is established (TRUE), the data (16-bit/32-bit) stored in S is overwritten at D of the data block specified by n.

- S : When the data store device is 16 bits (.U/.S)



- S : When the data store device is 32 bits (.D/.L/.F)



"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Table Processing Instructions."

● **Example of format**

FWRIT (R3000, DM1000.U, DM2000, 5)

Point

The function is not executed when either of the following conditions is established:

- **When the value of [n] is 0 or greater than number of data (N)**
- **When maximum number of stored data (max) is "0"**
- **When the number of data exceeds the maximum number of stored data**
- **When the maximum number of stored data is a number that causes the data table to exceed the device range**

Sample Program

● **When an index register is used**

Programmed Script

FWRIT (R3000, DM1000:Z1, DM2000:Z2, DM2100)

Operation Description

When R3000 turns ON, the 16-bit data currently stored in DM (1000+Z1) is overwritten on the block position indicated by DM2100 of the data block starting from DM (2000+Z2).

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

Hint

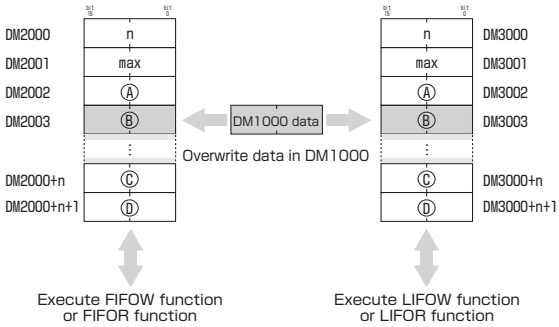
The FWRIT function can be used in either of the data tables used by the FIFOW function/FIFOR function or the LIFOW function/LIFOR function.

Programmed Script

FIFOW (EM5000, DM2000)	'Makes a FIFO table.
LIFOW (EM5100, DM3000)	'Makes a LIFO table.
FWRIT (DM1000, DM2000, 2)	'Overwrites a FIFO table.
FWRIT (DM1000, DM3000, 2)	'Overwrites a LIFO table.

Operation Description

The FWRIT function can be used in either of the data tables used by the FIFOW function/FIFOR function or the LIFOW function/LIFOR function.



FINS Insert to data table specified position

Insert data table

FINS ([execution condition,][*1] data store device, data table leading device, insert block position)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	For mula
S	Data store device*2	Specifies the device to which the data to overwrite is currently stored, or the overwrite data.	Handled as same type as D.					—	—	—	—	○	○
D	Data table leading device*2	Specifies the start of the table.	.U	.U	.D	.D	.D	—	—	—	—	○	—
n	Insert block position	Specifies the device to which the position of the data block to insert is currently stored, or the insert position.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

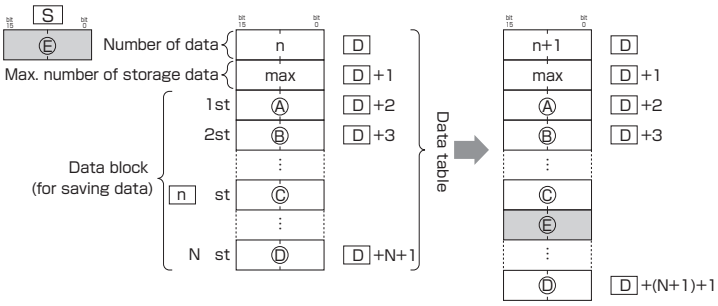
*1 [ ] can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When a bit device has been specified, 16/32 continuous bits are handled. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16/32 bits straddling the next channel.

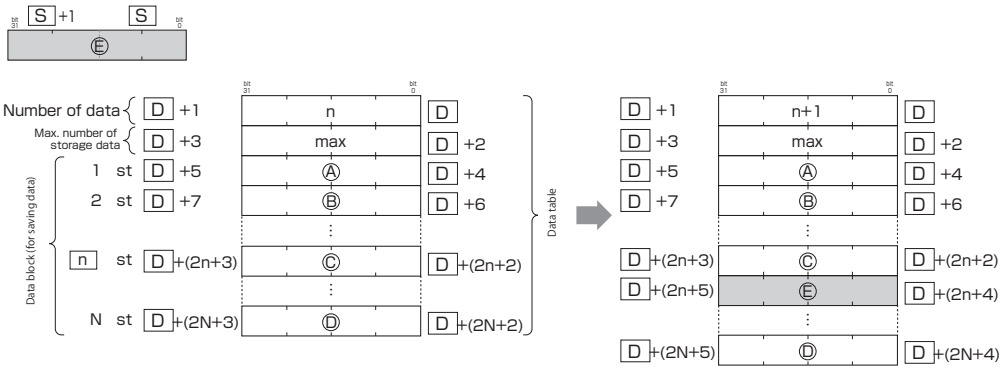
Description of Operation

When the execution condition is established (TRUE), the data (16-bit/32-bit) stored in S is inserted at D of the data block specified by n.

- S : When the data store device is 16 bits (.U/.S)



- S : When the data store device is 32 bits (.D/.L/.F)

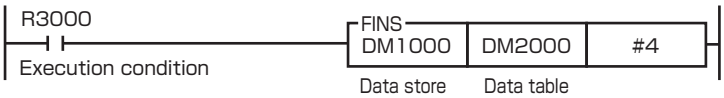


"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Table Processing Instructions."

● **Example of format**

FINS (R3000, DM1000.U, DM2000, 4)



Point

The function is not executed when either of the following conditions is established:

- **When the value of [n] is 0 or greater than number of data (N)**
- **When maximum number of stored data (max) is "0"**
- **When the number of data exceeds the maximum number of stored data**
- **When the maximum number of stored data is a number that causes the data table to exceed the device range**

Sample Program

● **When an index register is used**

Programmed Script

FINS (R3000, DM1000:Z1, DM2000:Z2, DM2100)

Operation Description

When R3000 turns ON, the 16-bit data currently stored in DM (1000+Z1) is inserted at the block position indicated by DM2100 of the data block starting from DM (2000+Z2).

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

Hint

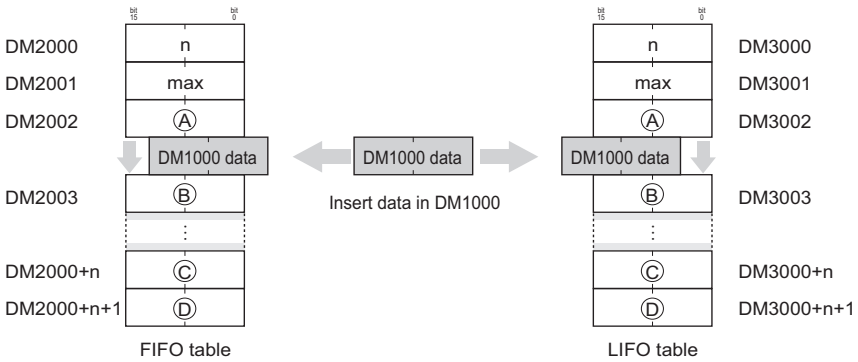
The FINS function can be used in either of the data tables used by the FIFOW function/FIFOR function or the LIFOW function/LIFOR function.

Programmed Script

FIFOW (EM5000, DM2000)	'Makes a FIFO table.
LIFOW (EM5100, DM3000)	'Makes a LIFO table.
FINS (DM1000, DM2000, 2)	'Inserts in the LIFO table.
FINS (DM1000, DM3000, 2)	'Inserts in the LIFO table.

Operation Description

The 16-bit data currently stored in DM1000 is inserted at each of the 2nd data blocks of the FIFO data blocks starting from DM2000 and the LIFO data blocks starting fromDM3000 at every scan (no execution condition).



FDEL Delete data table specified position

Delete data table

FDEL ([execution condition,]*1 data table leading device, delete block position)

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Data table leading device*2	Specifies the start of the table.	.U	.U	.D	.D	.D	—	—	—	—	○	—
n	Delete block position	Specifies the device to which the position of the data block to delete is currently stored, or the delete position.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 [ ] can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

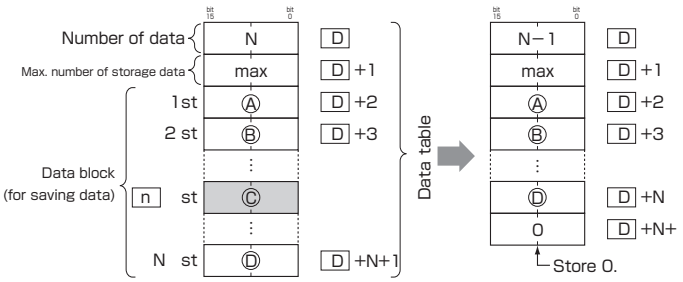
*2 When a bit device has been specified, 16/32 continuous bits are handled. When a relay other than the leading relay (e.g. R002, R1012) of the channel has been specified, the relay is handled in16/32 bits straddling the next channel.

Description of Operation

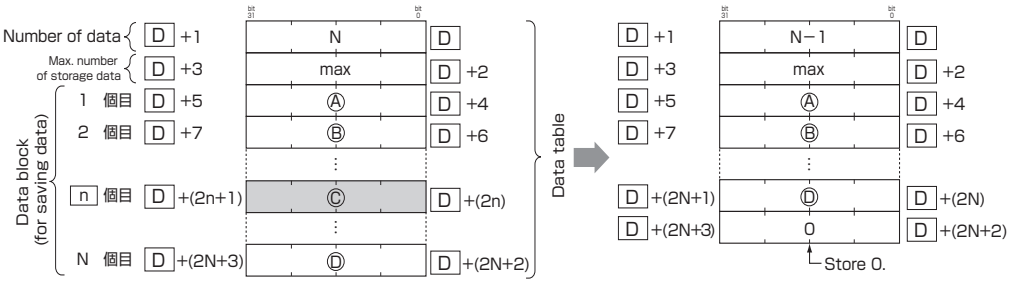
When the execution condition is established (TRUE), the data at D of the data block specified by the n is deleted.

After the data is deleted, the content of the data block is pushed forward by one data, and the number of stored data is decremented (-1).

- D : When the data table leading device is 16 bits (.U/.S)



- D : When the data table leading device is 32 bits (.D/.L/.F)

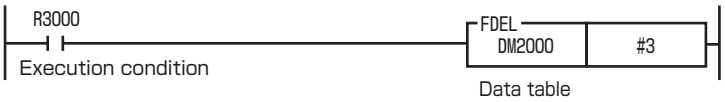


"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Table Processing Instructions."

● **Example of format**

FDEL (R3000, DM2000.U, 3)



Point

The function is not executed when either of the following conditions is established:

- **When the value of [n] is 0 or greater than number of data (N)**
- **When maximum number of stored data (max) is "0"**
- **When the number of data exceeds the maximum number of stored data**
- **When the maximum number of stored data is a number that causes the data table to exceed the device range**

Sample Program

● **When an index register is used**

Programmed Script

FDEL (R3000, DM2000:Z1, DM2100)

Operation Description

When R3000 turns ON, the data at the block position indicated by DM2100 of the data block starting from DM (2000+Z1) is deleted.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

Hint

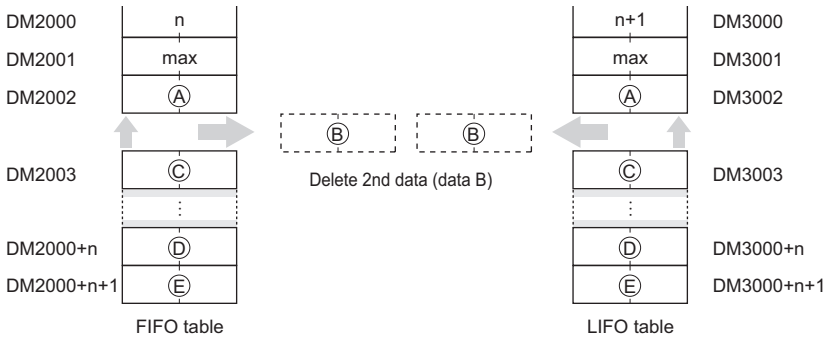
The FDEL function can be used in either of the data tables used by the FIFOW function/FIFOR function or the LIFOW function/LIFOR function.

Programmed Script

FIFOW (EM5000, DM2000)	'Makes a FIFO table.
LIFOW (EM5100, DM3000)	'Makes a LIFO table.
FDEL (DM2000, 2)	'Delete from FIFO table.
FDEL (DM3000, 2)	'Delete from LIFO table.

Operation Description

During every scanning (no execution condition), the data stored in the 2nd data block of DM2000 FIFO data block, and the 2nd data block of DM3000 LIFO data block will be deleted once.



WTIME Write time data to calendar timer

Write time

WTIME ([execution condition,][*1] Year, Month, Day, Hours, Minutes, Seconds, Week)

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Time setting target	Year data (0 to 99)	.U	.U	.U	.U	—	—	—	—	○	○	○
S2		Month data (1 to 12)	.U	.U	.U	.U	—	—	—	—	○	○	○
S3		Day data (1 to 31)	.U	.U	.U	.U	—	—	—	—	○	○	○
S4		Hour data (0 to 23)	.U	.U	.U	.U	—	—	—	—	○	○	○
S5		Minute data (0 to 59)	.U	.U	.U	.U	—	—	—	—	○	○	○
S6		Second data (0 to 59)	.U	.U	.U	.U	—	—	—	—	○	○	○
S7		Day of week data (0 to 6)	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

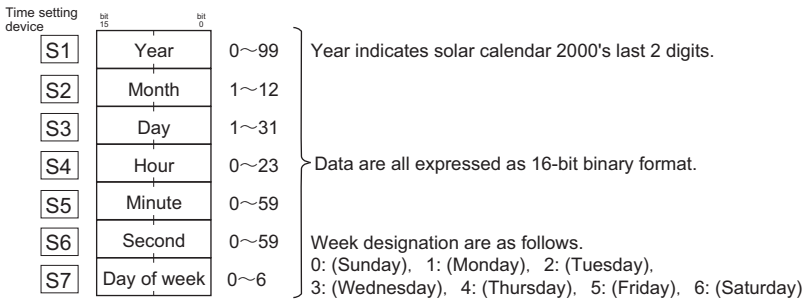
*1 [ ] can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

Reference When word data has been specified to S1 to S7, it is handled entirely as unsigned16-bit data. (The specified suffix is ignored).

* This function cannot be used with the KV-N14**. No error will occur when an attempt is made to execute the function, but the function will not be executed.
(A conversion error will not occur in KV-N14**, KV STUDIO)

Description of Operation

When the execution condition is established (TRUE), the time data currently stored in S1 to S7 is written to the built-in calendar timer.

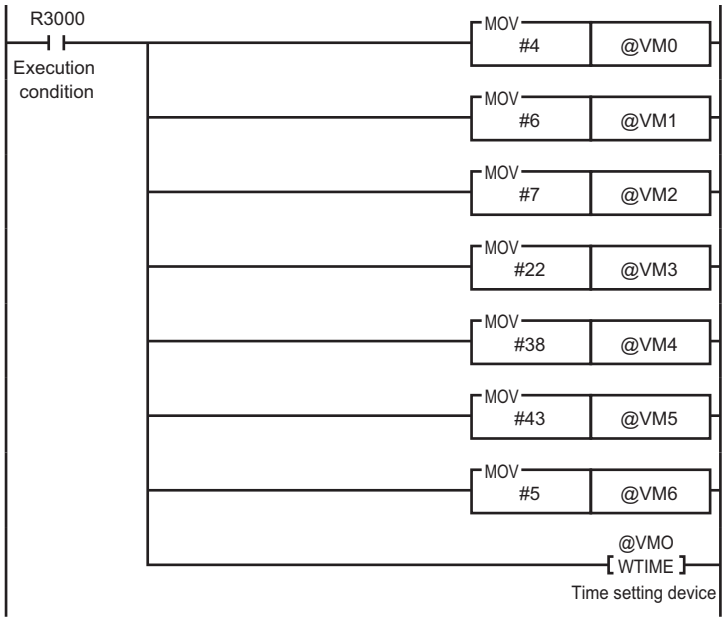


"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Clock Processing Instructions."

● **Example of format**

WTIME (R3000, 04, 06, 07, 22, 38, 43, 5)



Point

- **The time data cannot be written independently and directly to control memories.**
- **When the WTIME function is used, specify the time data in the correct format (year/month/day/hours/minutes/seconds).**
  The day of the week is not checked for the year, month, day, hour, minutes, and seconds settings.
- **When the data to write is in the inappropriate format as the time data, the WTIME function is not executed.**

SEC

Convert date/time format data to second format data

Return value [*1] = SEC (conversion source device)

Date/time → seconds conversion

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Conversion source device[*2*3]	Specifies the start that stores the date and time of the conversion source.	.U	.U	.U	.U	.F	—	—	—	—	○	—
R	Return value	.U type SEC function available[*4]	.U	.S	.D	.L	.F	[*5] .DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in internal registers.

*2 When a bit device has been specified, continuous 96 bits are handled as the date/time data and continuous 32 bits are handled as the second data. A relay other than the leading relay (e.g. R002 or R1012) of the channel can be specified.

*3 T, C, CTC, CTH, Z cannot be used.

*4 When the type other than .D is specified by return value, type conversion will be automatically conducted during conversion.

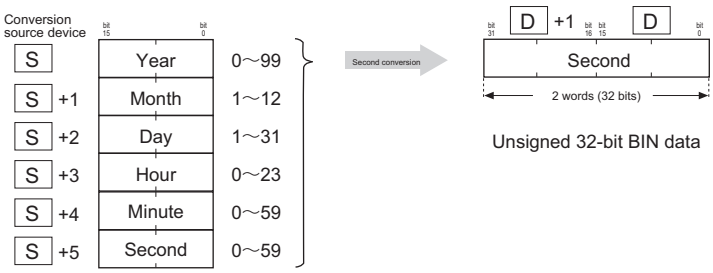
*5 In device Z, the ".DF" suffix cannot be specified.

Point

- **The argument device ".DF" can be used only with the KV STUDIO Ver. 6.00 and higher versions.**
- **The argument device ".DF" can only be specified with KV-8000/7000 Series CPU units, with KV-5500/5000/3000 Series CPU units that have CPU function version 2.0 or higher and with KV Nano Series base units.**
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

S The six words (96 bits) of date/time data currently stored starting from are converted to a total number of seconds referenced to year 2000, January 1st, 0:0:0 (hours: minutes: seconds), and stored in R as 32-bit BIN data (unsigned).



Year indicates solar calendar 2000's last 2 digits.

Reference

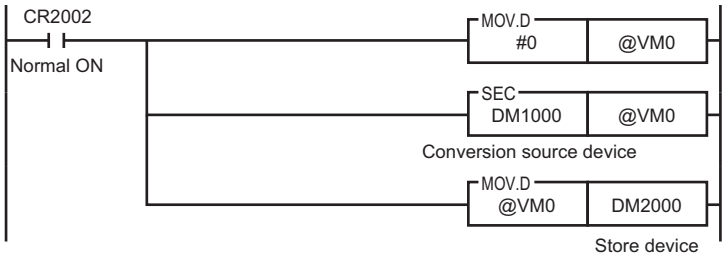
Date/time data is specified entirely as 16-bit BIN data, and can be used as both unsigned and signed.

"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Clock Processing Instructions."

● **Example of format**

DM2000.D = **SEC** (DM1000.U)



Point

When either of the following conditions is established, 0 is stored in [R]:

- **When the date/time data to convert is in an inappropriate format**
- **When a timer/counter is specified to [S] and [R] by indirect specification**

Sample Program

● **When this function is used in combination with other functions**

Programmed Script

RSEC (TOL (**SEC** (CM0700)) + 864000, DM2000)

Operation Description

The date/time ten days from the current date/time is calculated.
The current date/time (2000, January 1st, 0:0:0 (hours: minutes: seconds) data currently stored starting from CR0700 is converted to a total number of seconds from the reference date/time.
The total number of seconds ten days after obtained by adding 864000 seconds (ten days) to the total number of seconds of the conversion result is reverse-calculated to date/time data, and stored in six words from DM2000.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "RSEC function", page 8-74
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

RSEC Convert second format data to date/time format data

Seconds Date/time conversion

RSEC ([execution condition,][*1] conversion source device, conversion destination device)

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Conversion source device[*2*3]	Specifies the storage location for the seconds (total seconds) of conversion source.	—	—	.D	.D	—	—	—	—	—	○	○
D	Conversion source device[*2*3]	Specifies the start to store the converted date and time.	.U	.U	.U	.U	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

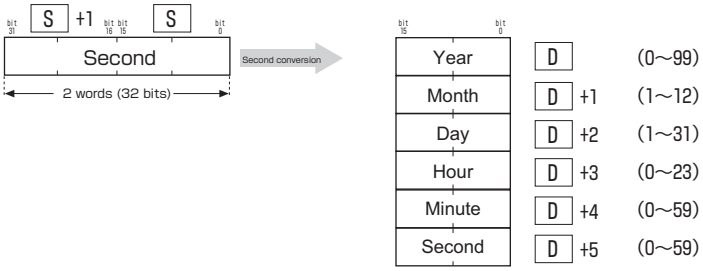
*1 [ ] can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When a bit device has been specified, continuous 96 bits are handled as the date/time data and continuous 32 bits are handled as the second data. A relay other than the leading relay (e.g. R002 or R1012) of the channel can be specified.

*3 Timer devices and counter devices cannot be used.

Description of Operation

The total number of seconds (32-bit BIN data (unsigned)) referenced to "year 2000, January 1st, 0:0:0" and specified by the S is converted to date/time data, and is stored in six words (96 bits) starting from the device specified by the D.



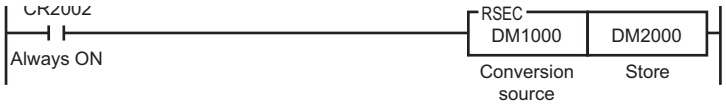
Reference The date/time data is stored entirely as 16-bit BIN data.

"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Clock Processing Instructions."

● Example of format

RSEC (DM1000.D, DM2000)



Point

The function is not executed when either of the following conditions is established:

- **When the second data to convert is in an inappropriate format**
- **When a timer/counter is specified to S and D by indirect specification**

Sample Program

● When this function is used in combination with other functions

Programmed Script

RSEC (TOL (SEC (CM0700)) - 259200, DM2000)

Operation Description

The date/time three days from the current date/time is calculated.
The current date/time (2000, January 1st, 0:0:0 (hours: minutes: seconds) data currently stored starting from CM0700 is converted to a total number of seconds from the reference date/time.
The total number of seconds three days after obtained by subtracting 259200 seconds (three days) to the total number of seconds of the conversion result is reverse-calculated to date/time data, and stored in six words from DM2000.

Reference

- Note that this function sometimes cannot be used depending on the argument of the function it is combined with.
  "SEC function", page 8-72
- The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

AJST Calendar timer adjustment

Adjust
AJST ([execution condition,]*1)

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 [ ] can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

* This function cannot be used with the KV-N14**. No error will occur when an attempt is made to execute the function, but the function will not be executed.
(A conversion error will not occur in KV-N14**, KV STUDIO)

Description of Operation

When the execution condition is established (TRUE), the "seconds" of the current time according to the built-in calendar timer is set to "0" when it is 0 to 29, or "minutes" is incremented by one digit and "seconds" is set to "0" when "seconds is 30 to 59 to adjust the timer to the nearest minute unit ("0" seconds).

Before perform		After perform
12 hours, 59 minutes, 15 seconds	→	12 hours, 59 minutes, 00 seconds
12 hours, 59 minutes, 30 seconds	→	13 hours, 00 minutes, 00 seconds

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "Clock Processing Instructions."

● Example of format

AJST (R3000)

Reference When the execution condition is omitted, the script becomes as follows:

Programmed Script **AJST** ( ) ' Execution condition omitted

Reference The KV script is converted to a ladder program before execution. To check the converted content, refer to "Precautions When Using Box Script""New a box script", step 5, (page 2-10)

MEMO

DI Disable interrupt
EI Enable interrupt

Disable interrupt
DI ([execution condition,]*1)
Enable interrupt
EI ([execution condition,]*1)

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 [ ] can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

Description of Operation

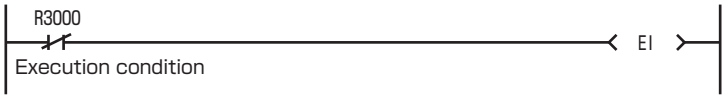
The DI function disables execution of the interrupt program when the execution condition is established (TRUE).
The EI function enables execution of the interrupt program when the execution condition is established (TRUE).
Execution of interrupt programs temporarily disabled by the DI function are enabled (disable canceled) by the EI function.

- Once the EI function is executed, interrupts up to execution of the DI function are enabled.
- Interrupts are disabled at all times when operation is started.
- Interrupts up to a nesting level of 16 can be stored if they occur during execution of the DI function (interrupts are disabled).
  When the EI function is executed, interrupts stored in memory are executed.
- To execute another interrupt program while an interrupt program is being executed, execute the EI function in the interrupt program. The sequence program accepts nesting of interrupt programs up to a nesting level of 16.
  For details, see 📖 KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "INT/RETI Instruction" and "High-speed Processing Instructions."

● Example of format

DI (R3000)

EI (R3000 = OFF)



Point **When the EI function is executed once in an interrupt disabled state at the start of operation, interrupts are enabled until the DI function is executed.**

Reference When the interrupt program is executed, other interrupts are disabled until execution of that interrupt program ends. To enable other interrupts, the EI function must be executed during execution of the interrupt program. When the EI function is executed, the interrupt program having the higher priority than the executed interrupt factor can be enabled (nested interrupt). Note, however, that interrupts by the same interrupt factor cannot be programmed.

RFSCTH High-speed Counter refresh

High-speed Counter refresh

RFSCTH ([Execution condition,][*1] High-speed counter No.)

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	High-speed counter No.	Specifies the high-speed counter No.	—	—	—	—	—	—	—	—	*2 ○	—	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 [ ] can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 The high-speed counter number is specified as a single digit number (0 to 3). The high-speed counter number that can be used differs according to the unit.

Description of Operation

When the execution condition is established (TRUE), change the current value of CTH specified by S high-speed counter no. to the latest value.

At the same time, the CR (control relay) of CTH overflow and change method shall be also changed to the latest status.

KV-5500/5000/3000/1000

High-speed counter	Control relay to be refresh	
	Overflow	Change method
CTH0	CR2403	CR2412
CTH1	CR2407	CR2413

KV Nano

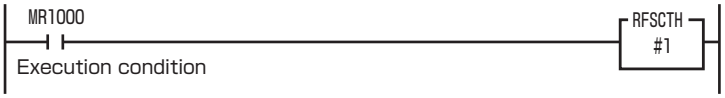
High-speed counter	Control relay to be refresh	
	Overflow	Change method
CTH0	CR4206	CR4207
CTH1	CR4406	CR4407
CTH2	CR4606	CR4607
CTH3	CR4806	CR4807

"About operation flag (CR2009 to CR2012)", page 1-18

For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "High-speed Processing Instructions."

● Example of format

RFSCTH (MR1000, 1)



DIC Set interrupt disabled range

Disable interrupt

DIC ([execution condition,]*1 interrupt state store device)

Argument/Return Value		Description	Recognized Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
D	Interrupt state store device*2	Specifies the device to store the interrupt disabled state before instruction execution to.	—	—	—	—	—	—	.B	—	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 [ ] can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When the interrupt state store device is a word device, only the LSB is targeted for storing the interrupt state. (Bits other than the LSB do not change.)

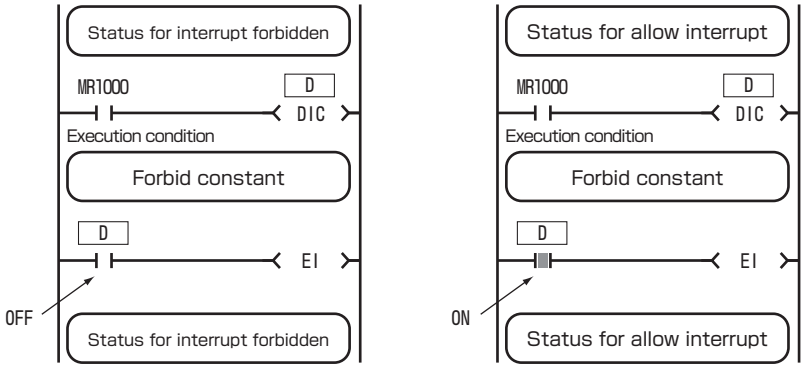
Description of Operation

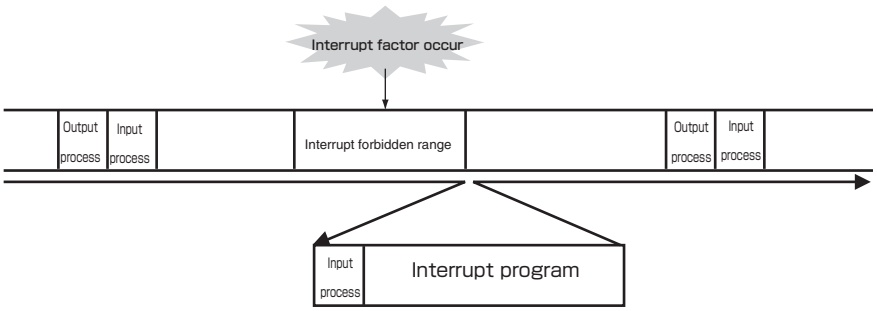
The DIC function disables execution of the interrupt program when the execution condition is established (TRUE).

At the same time, D is set if the state before function execution is an interrupt enabled state, and is reset if the state before function execution is an interrupt disabled state.

Reference

Since the interrupt status storage target and the interrupt status before the interrupt program is disabled are stored, they can be used when nesting is interrupted in combination with the execution condition of the EI function (interrupt enable).

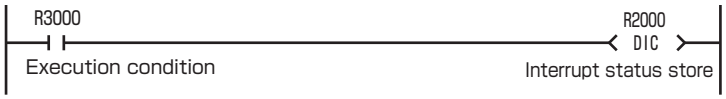




For details on execution timing, see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "High-speed Processing Instructions."

● **Example of format**

DIC (R3000, R2000)



Sample Program

Programmed Script

DIC (R2000)

FOR DM1000 = 0 TO 10

  DM2000.L = DM2000 + DM1000

NEXT

EI (R2000)

Operation Description

Execution of the interrupt program in the specified range is disabled at every scan (no execution condition). If the state before execution of the DIC function is the enabled state, R2000 is set, and if it is interrupt disabled, R2000 is reset.
When R2000 is ON (state before execution of interrupt processing is an interrupt enabled state), execution of the interrupt program by the EI function and below is enabled.

Reference

The KV script is converted to a ladder program before execution. To check the converted content, refer to "New a box script", step 5, (page 2-10)

IEDGE Set interrupt condition

Set interrupt condition

IEDGE ([execution condition,]*1 unit number, input number, detection condition)

Argument/Return Value		Description	Recognized Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
n1	Unit number	Specifies expansion unit number for writing input interrupt detection condition. (0 to 48)	.U	.U	.U	.U	—	—	—	—	*2 ○	—	—
n2	Input number	Specifies interrupt factor number. (0 to 31)	.U	.U	.U	.U	—	—	—	—	*3 ○	○	○
S	Detection condition	Specified detection condition (rising edge: 0 or 1, falling edge: 2, rising edge and falling edge: 3).	.U	.U	.U	.U	—	—	—	—	*4 ○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 [ ] can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

Description of Operation

When the execution condition is TRUE, S the 1-word data specified by n1 is set to the detection condition of the cause interrupt of n2 unit.

Depending on the value of S, the following detection conditions are available.

S	Detection condition
0 to 1	Rising edge
2	Falling edge
3	Rising edge and falling edge

Point

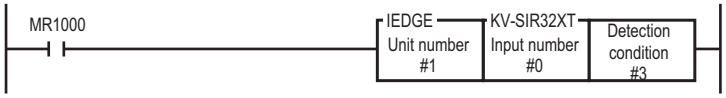
- **Specifying to S the range of 4 to 65535 does not result in operation error, but the detection condition is not changed.**
- **When execution of this instruction is completed, the detection condition is changed.**

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please see KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual, "IEDGE Instructions".

● Example of format

IEDGE (MR1000,1,0,3)



MEMO

PIDAT PID control with automatic tuning

PID control with AT

PIDAT ([execution condition,][*1] measured value, parameter, manipulated value)

Argument/Return Value		Description	Supported Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Measured value	Specifies the value measured form the control target.	.S	.S	.S	.S	—	—	—	—	—	○	○
D1	Parameter*2	Specifies the start to store the parameter.	.U	.U	.U	.U	—	—	—	—	—	○	—
D2	Manipulated value	Specifies the destination to store the PID operation result.	.S	.S	.S	.S	—	—	—	—	—	○	—
R	Return value	None	—	—	—	—	—	—	.B	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 The D1 parameter occupies 51 continuous words. Refer to the explanation of the PIDAT instruction code for details on the parameter.

Point

- The PIDAT function can be used only with KV STUDIO Ver. 6.00 or higher.
- This function can only be used with KV-8000/7000 Series CPU units, KV-5500/5000/3000 Series CPU units with CPU function version 2.0 or higher and KV Nano Series base units.
  "KV-5500/5000/3000 CPU Function Version", page 4

Description of Operation

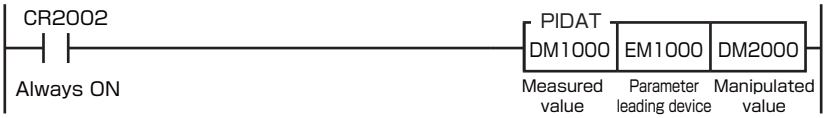
D2 (PID operation result) is retrieved from S and D1.
At the first call, (when executing initialization), the execution condition is turned ON and the D1 parameter work area is cleared to zero before calling the PIDAT function.
When continuing operations, the execution conditions remain ON, and the PIDAT function is called out.

Reference: After the PIDAT function is called after the execution conditions have turned OFF and ON, the initialization process will be executed.

Refer to "PIDAT instruction" of "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual" for details on the operation flags and D1.

● Example of format

PIDAT (DM1000, EM1000, DM2000)



MEMO

LOGE Enable logging of specified ID
LOGD Disable logging of specified ID

Enable logging
LOGE ([execution condition,]*1 logging ID)
Disable logging
LOGD ([execution condition,]*1 logging ID)

Argument/Return Value		Description	Supported Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
n	Logging ID	Specifies the destination that stores the logging ID for enabling/disabling logging, or the logging ID. (0 to 9)	.U	.U	.U	.U	—	—	—	—	*2 ○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)
*2 Cannot be specified by "$".

Description of Operation

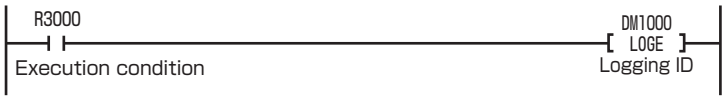
When the execution condition is established (TRUE), the LOGE function enables execution of logging by the logging ID specified by n. After the function is executed, the logging execution relay of each logging ID is set.
When the execution condition is established (TRUE), the LOGD function disables execution of logging by the logging ID specified by n. After the function is executed, the logging execution relay of each logging ID is reset.

Logging ID	Logging execution relay
0	CR0000
1	CR0100
2	CR0200
3	CR0300
4	CR0400
5	CR0500
6	CR0600
7	CR0700
8	CR0800
9	CR0900

"About operation flag (CR2009 to CR2012)", page 1-18
Refer to User's Manual of the CPU unit used.

● Example of format

LOGE (R3000, DM1000)



Point **When n is set to other than a constant or set to 10 or more, the function will not be executed.**

TRGD Acquire logging data

Acquire logging data

TRGD ([execution condition,]*1 logging ID)

Argument/Return Value		Description	Supported Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
n	Logging ID	Specifies the logging ID. (0 to 10)*2	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*1 Please specify the logging ID as 0 to 10. Please specify the logging ID as 10 when trigger instruction is used on real-time curve monitor.

Description of Operation

When the execution condition is established (TRUE), the data of the logging ID specified by n is acquired.

Under general condition, data is acquired by log, trace, real-time curve monitor via END processing timing. Using TRGD instruction can acquire data via any timing of ladder.

Point

- **Only up to eight TRGD functions (instructions) can be used per ID in one project.**
- **To acquire data using TRGD function (instruction), it is necessary to set trigger type to "trigger instruction" via logging, trace, real-time curve monitor setting of KV STUDIO. If it is not set to "trigger instruction", even if TRGD function (instruction) is executed, data cannot be acquired.**

"About operation flag (CR2009 to CR2012)", page 1-18

Refer to User's Manual of the CPU unit used.

● Example of format

TRGD (MR1000, 1)

TRGR Get device / variable data

Get device / variable data

TRGR ([execution condition,]*1 Operation recorder ID)

Argument/Return Value		Description	Supported Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
n	Operation recorder ID	Specifies the operation recorder ID. (0 to 3)	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

Description of Operation

When the execution condition is established (TRUE), the data of the operation recorder ID specified by n is acquired.

The data is generally acquired at the time of END processing, but the data can be acquired at any time of the ladder using the TRGR instruction.

Point

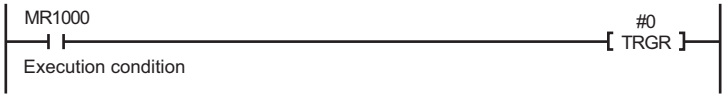
- **The TRGR instruction can be used only with KV STUDIO Ver.10.0 or later and the compatible model of KV-8000.**
- **To acquire data using TRGR function (instruction), it is necessary to set sampling period to "trigger instruction" in the operation recorder setting of KV STUDIO. If it is not set to "trigger instruction", even if TRGR function (instruction) is executed, data cannot be acquired.**

"About operation flag (CR2009 to CR2012)", page 1-18

Refer to "Operation Recorder Function" in "KV-8000 Series User's Manual" for details.

● Example of format

TRGR(MR1000, 0)



MEMO

MWRIT Write to storage device (memory card/CPU memory)

Return value*1 = MWRIT (save file name, save data storage destination, number of save data storage destination, save format storage destination, save result storage destination)

Write to storage device

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S1	Save file name*7	Specifies the storage location for the file name for data saving or the save file name. (A text string in the range 1 to 247 characters (including end code NUL and folder name) is specified.)	—	—	—	—	—	—	—	.T	*3 ○	○	○
S2	Save data storage destination*4*5	Specifies the start that stores the save data.	.U	.U	.U	.U	—	—	—	—	—	○	—
n	Number of save data storage destination	Specifies the storage location for the number of data in the save range or the number of save data.	.U	.U	.U	.U	—	—	—	—	—	○	○
D1	Save format storage destination*5*6	Specifies the start to store the parameters related to the save format, etc. and the data save status.	.U	.U	.U	.U	—	—	—	—	—	○	—
D2	Save result storage destination*2	Specifies the bit data to notify the save result.	—	—	—	—	—	—	.B	—	—	○	—
R	Return value	Turns ON when the function is successfully started up, and turns OFF when startup fails.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work device / work memory.

*2 When bit data is specified for the save result storage destination, 2 continuous bits are used as notification bits; when word data is specified, the LSB of each of 2 continuous words is used as notification bit data.

*3 To specify a text string using constants, enclose with " ".

*4 When saving index register Z and digital trimmer values set in the access window, copy the data once to a data memory DM, for example, and then set that data memory DM.

*5 When specifying bit data, specify the start of the channel.

*6 D1 When bit data is specified, it occupies 96 continuous bits. When word data is specified, 6 continuous words are occupied.

*7 T, C, CR, CTC, CTH, Z cannot be used.

Point **This function cannot be used with KV-N14□□/N24□□/N40□□/N60□□. (A conversion error will not occur in KV STUDIO, only a warning is displayed).**

Description of Operation

The data stored in the range specified by n is written to the memory card or CPU memory with the file name (text string) specified by S1 starting from S2.

The write process is performed according to the write parameters specified by D1, and the write completion notification and abnormal end alarm notification are stored in D2.

R is turned ON when the MWRIT function is started normally, and OFF when it fails.

For details on parameters related to the save format, etc. specified in D1, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "MWRIT instruction".

Reference

D2 The notification bits of the save result storage destination have the following meanings.

(Example) When MR2000 has been specified

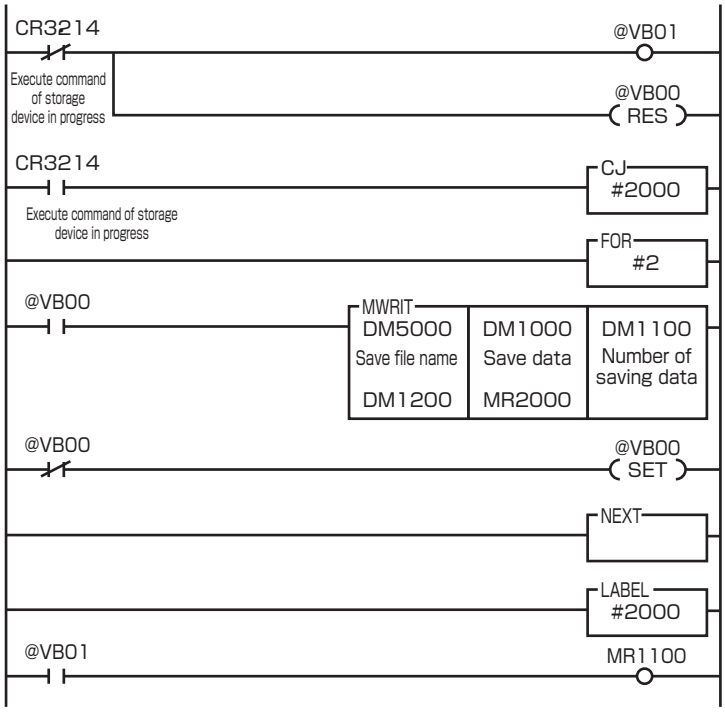
Notification Bit		ON (1, TRUE)	OFF (0, FALSE)
MR2000	Completion notification	Completed	Not completed
MR2001	Error notification	Normal end	Abnormal end

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Storage Instructions".

● Example of format

MR1100 = **MWRIT** (DM5000.T, DM1000.U, DM1100.U, DM1200.U, MR2000)



Point

The function is not executed when any of the following conditions is established:

- **When the text string specified by S1 does not contain the end code (NUL (00H))**
- **When the text string specified by S1 exceeds 247 characters (including end code "NUL (00H)")**
- **When a timer/counter is specified to S1 by indirect specification.**
- **When a bit other than the start of the channel is specified when bit data is specified for S1, S2 and D1.**
- **When a value outside of the range is specified to an argument**
- **When a value is specified for reservation of an argument**

MREAD Read from storage device (memory card/CPU memory)

Return value*1 = MREAD (read file name, read data storage destination, number of read data storage destination, read format storage destination, read result storage destination)

Read from storage device

Argument/Return Value	Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S1 Read file name*4*7	Specifies the storage location for the file name for data reading or the read file name. (A text string in the range 1 to 247 characters (including end code NUL and folder name) is specified.)	—	—	—	—	—	—	—	.T	*3 ○	○	○
S2 Read data storage destination*4	Specifies the start that stores the read data.	.U	.U	.U	.U	—	—	—	—	—	○	—
n Number of read data storage destination*5	Specifies the storage location for the number of data in the read range or the number of read data.	.U	.U	.U	.U	—	—	—	—	○	○	○
D1 Read format storage destination*6	Specifies the start to store the parameters related to the read format, etc. and the data read status.	.U	.U	.U	.U	—	—	—	—	—	○	—
D2 Read result storage destination*2*4	Specifies the bit data to notify the read result.	—	—	—	—	—	.B	—	—	—	○	—
R Return value	Turns ON when the function is successfully started up, and turns OFF when startup fails.	—	—	—	—	—	.B	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work device / work memory.
*2 When bit data is specified for the read result storage destination, 2 continuous bits are used as notification bits; when word data is specified, the LSB of each of 2 continuous words is used as notification bit data.
*3 To specify a text string using constants, enclose with " ".
*4 When specifying bit data, specify the start of the channel.
*5 When reading ASCII text string data, specify the maximum number of characters (number of bytes) in the text string.
*6 D2 When bit data is specified, it occupies 96 continuous bits. When word data is specified, 6 continuous words are occupied.
*7 T, C, CR, CTC, CTH, Z cannot be used.

Point **This function cannot be used with KV-N14□□/N24□□/N40□□/N60□□. (A conversion error will not occur in KV STUDIO, only a warning is displayed).**

Description of Operation

The data of the file name (text string) specified by S is read from the storage and stored in the range specified by n starting with D1.
The read process is performed according to the read parameters specified by D2, and the read completion notification and abnormal end alarm notification are stored in D3.
R is turned ON when the MREAD function is started normally, and OFF when it fails.
For details on parameters related to the read format, etc. specified in D2, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "MREAD instruction".

Reference

D3 The notification bits of the read result storage destination have the following meanings.

(Example) When MR2000 has been specified

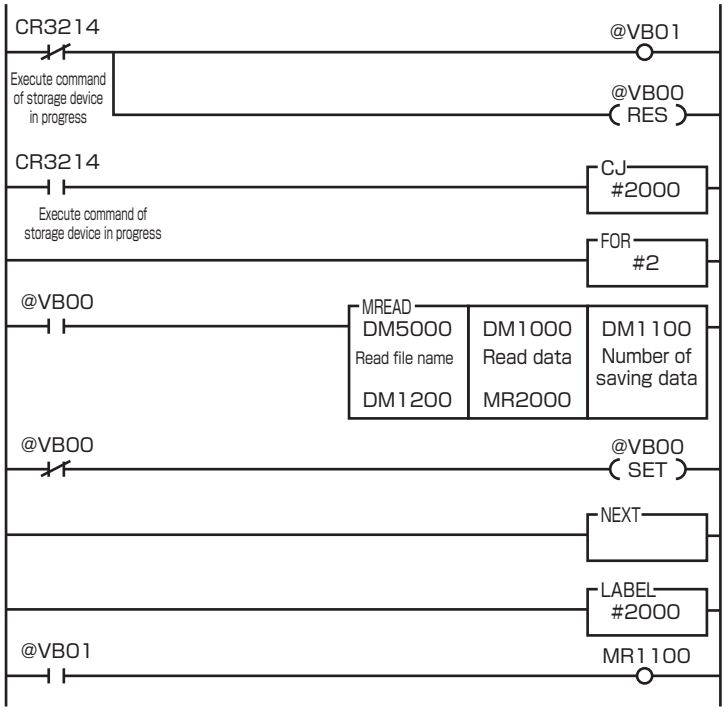
Notification Bit		ON	OFF
MR2000	Completion notification	Completed	Not completed
MR2001	Error notification	Normal end	Abnormal end

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Storage Instructions".

● Example of format

MR1100 = **MREAD** (DM5000.T, DM1000.U, DM1100.U, DM1200.U, MR2000)



Point

The function is not executed when any of the following conditions is established:

- **When the text string specified by S does not contain the end code (NUL (00H))**
- **When the text string specified by S exceeds 247 characters (including end code "NUL (00H)")**
- **When a timer/counter is specified to S by indirect specification.**
- **When a bit other than the start of the channel is specified when bit data is specified for S, D1 and D2.**
- **When a value outside of the range is specified to an argument**
- **When a value is specified for reservation of an argument**

MFREE Get storage device (memory card/CPU memory) free space in bytes

Return value*1 = MFREE ((drive number,)*4 free space storage destination, get result storage destination)

Get storage device free space

Argument/Return Value		Description	Supported Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
n	Drive number*4	Specifies the drive number (memory card:0, CPU memory:1) for checking the free space.	.U	.U	.U	.U	—	—	—	—	—	○	—
D1	Free space storage destination*2	Specifies the destination to store the free space (in bytes) obtained from the memory card (CPU memory). (32-bit data)	—	—	.D	.D	—	—	—	—	—	○	—
D2	Get result storage destination*3	Specifies the bit data to notify the free space get result.	—	—	—	—	—	—	.B	—	—	○	—
R	Return value	Turns ON when the function is successfully started up, and turns OFF when startup fails.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work device / work memory.

*2 When bit data is specified, 32 continuous bits from the start of the channel are processed.

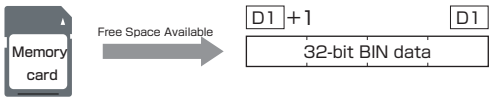
*3 When bit data is specified for the get result storage destination, 2 continuous bits are used as notification bits; when word data is specified, the LSB of each of 2 continuous words is used as notification bit data.

*4 For KV-8000/7000 Series, you can select memory card or CPU memory for checking the free space.

Point **This function cannot be used with KV-N14□□/N24□□/N40□□/N60□□. (A conversion error will not occur in KV STUDIO, only a warning is displayed).**

Description of Operation

The free space (in bytes) of the memory card or CPU memory specified by n is acquired and stored as 32-bit BIN data (2 words) in D1, and the completion notification of free space acquisition and abnormal end alarm notification are stored in D2.

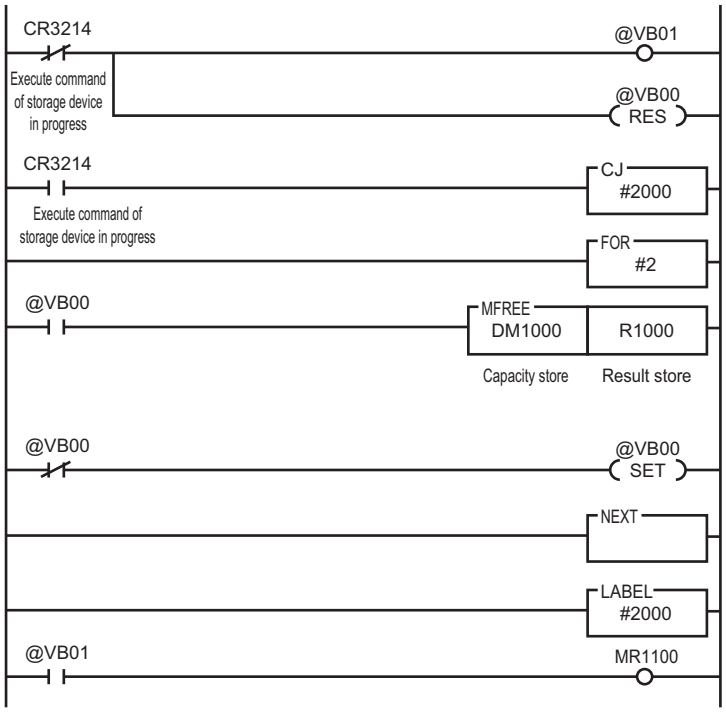


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Storage Instructions".

● **Example of format (For KV-5000)**

MR1100 = **MFREE** (DM1000.D, R1000)



● **Notification bit data (D2: completion notification, D2+1: error notification)**

Item		Details
D2	Completion notification	This device turns OFF when execution of the instruction starts, and turns ON when acquisition of free space in the memory card (or CPU memory) ends. The device turns ON whichever way execution ends, successfully or in error.
D2+1	Error notification	This device turns OFF when execution ends successfully, and turns ON when execution ends in error.

Point

- **The result of END processing after execution of the instruction is completed is written for the completion notification and error notification.**
- **Several scans are sometimes required for getting free space.**
- **The get free space sometimes differs from the actual free space when file operations were being performed via an FTP server function during acquisition processing or the logging function was being used.**
- **The MFREE function is not executed while a storage device function is being executed (CR3214 is ON).**
- **When a relay other than the leading relay of the channel has been specified to D1, the function is not executed.**
- **This instruction cannot be used in interrupt programs or in the initialization module.**

MFREEK Get storage device (memory card/CPU memory) free space in kilobytes

Get storage device free space

Return value*1 = MFREEK ((drive number,)*4 free space storage destination, get result storage destination)

Argument/Return Value		Description	Supported Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
n	Drive number*4	Specifies the drive number (memory card:0, CPU memory:1) for checking the free space.	.U	.U	.U	.U	—	—	—	—	—	○	—
D1	Free space storage destination*2	Specifies the target to store the obtained spare capacity (in kilobytes) of the memory card (CPU memory). (32-bit data)	—	—	.D	.D	—	—	—	—	—	○	—
D2	Get result storage destination*3	Specifies the bit data to notify the free space get result.	—	—	—	—	—	—	.B	—	—	○	—
R	Return value	Turns ON when the function is successfully started up, and turns OFF when startup fails.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work device / work memory.

*2 When bit data is specified, 32 continuous bits from the start of the channel are processed.

*3 When bit data is specified for the get result storage destination, 2 continuous bits are used as notification bits; when word data is specified, the LSB of each of 2 continuous words is used as notification bit data.

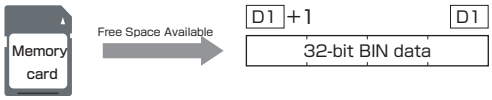
*4 For KV-8000/7000 Series, you can select memory card or CPU memory for checking the free space.

Point

- **The MFREEK function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can be specified only with KV-8000/7000 Series CPU unit, KV-5500/5000/ 3000 Series CPU unit with CPU function version 2.0 or higher, or KV-NC32T.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **This function cannot be used with KV-N14□□/N24□□/N40□□/N60□□. (A conversion error will not occur in KV STUDIO, only a warning is displayed).**

Description of Operation

The free space (in kilobytes) of the memory card or CPU memory specified by n is acquired and stored as 32-bit BIN data (2 words) in D1, and the completion notification of free space acquisition and abnormal end alarm notification are stored in D2.

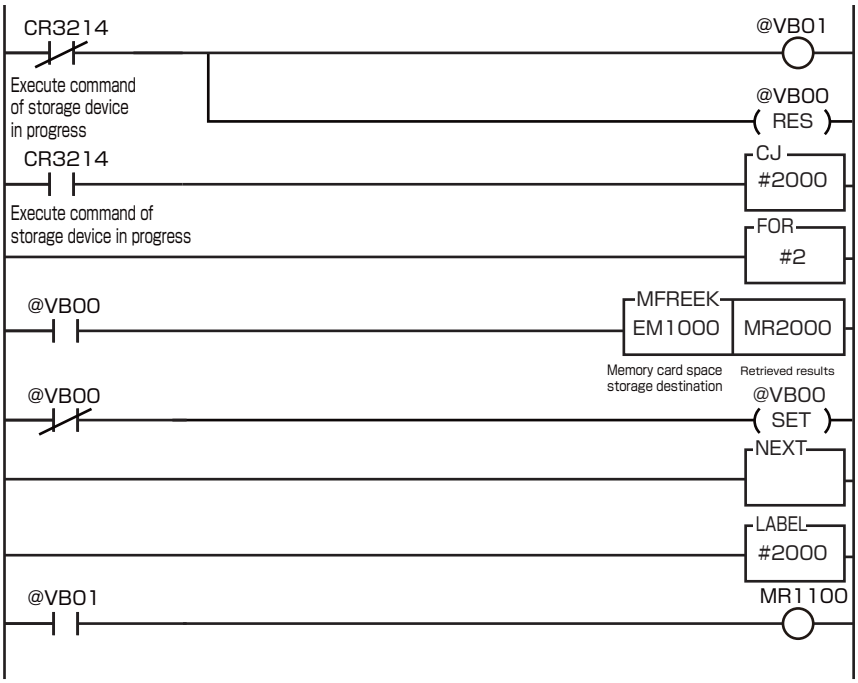


"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "MFREEK instruction".

● **Example of format (For KV-5000)**

MR1100 = **MFREEK** (EM1000.D, MR2000)



● **Notification bit data (D2: completion notification, D2+1: error notification)**

Item		Details
D2	Completion notification	This device turns OFF when execution of the instruction starts, and turns ON when acquisition of free space on the memory card (or CPU memory) ends. The device turns ON whichever way execution ends, successfully or in error.
D2+1	Error notification	This device turns OFF when execution ends successfully, and turns ON when execution ends in error.

Point

- **The result of END processing after execution of the instruction is completed is written for the completion notification and error notification.**
- **Several scans are sometimes required for getting free space.**
- **The get free space sometimes differs from the actual free space when file operations were being performed via an FTP server function during acquisition processing or the logging function was being used.**
- **The MFREEK function is not executed while a storage device function is being executed (CR3214 is ON).**
- **When a relay other than the leading relay of the channel has been specified to D1, the function is not executed.**
- **This instruction cannot be used in interrupt programs or in the initialization module.**

MMKDIR Make storage device (memory card/CPU memory) directory

Return value*1 = MMKDIR (created path name, make result storage destination)

Make storage device directory

Argument/Return Value	Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S Created path name*3*5	Specifies the storage location for the path name of the directory to make or the created directory path name. (A text string in the range 1 to 247 characters (including end code NUL) is specified.)	—	—	—	—	—	—	—	.T	*2 ○	○	○
D Make result storage destination*4	Specifies the destination to notify the make result.	—	—	—	—	—	—	.B	—	—	○	—
R Return value	Turns ON when the function is successfully started up, and turns OFF when startup fails.	—	—	—	—	—	—	.B	—	—	○	—

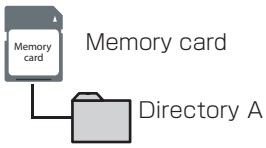
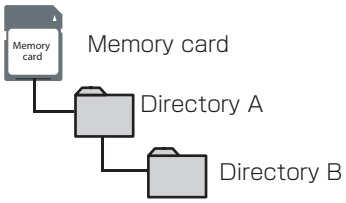
*1 The return value can be omitted. When omitted, the operation result is stored in a work device / work memory.
*2 To specify a text string using constants, enclose with " ".
*3 When specifying bit data, specify the start of the channel.
*4 When bit data is specified, it occupies 2 continuous bits. When word data is specified, the LSB of each of the 2 continuous words is used as notification bit data. Bits other than the LSB do not change.
*5 T, C, CR, CTC, CTH, Z cannot be used.

Point
- **This function cannot be used with KV-N14□□/N24□□/N40□□/N60□□. (A conversion error will not occur in KV STUDIO, only a warning is displayed).**

Description of Operation

A directory is made in the memory card or CPU memory using the path name (text string) specified by S.
The make result is stored in D.

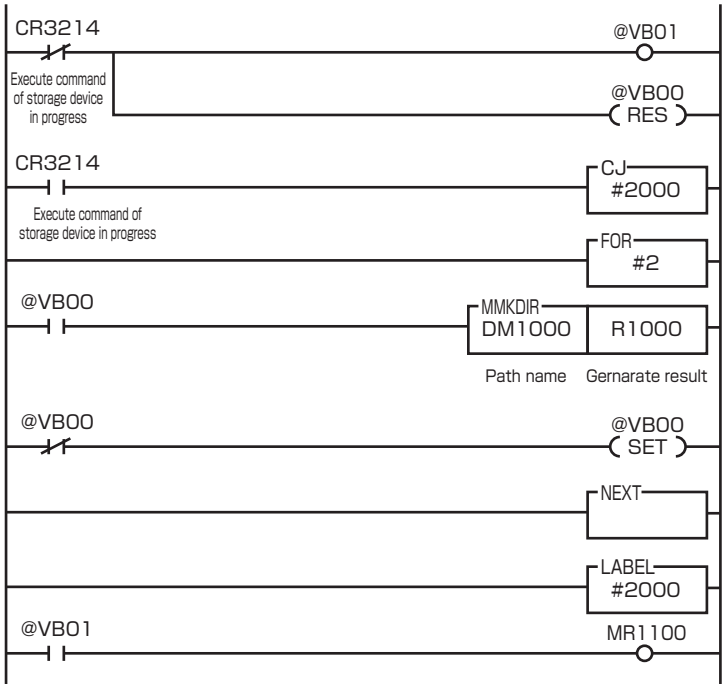
Example

S	"Directory A"	"Directory A\Directory B"
Example of directory structure		
Note	When a directory already exists, nothing happens, processing ends successfully, and the completion notification relay turns ON.	Prepare a new directory (directory A) in advance. It cannot be prepared together withdirectory B.

"About operation flag (CR2009 to CR2012)", page 1-18
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Storage Instructions".

● **Example of format**

MR1100 = **MMKDIR** (DM1000.T, R1000)



● **Notification bit data (D: completion notification, D+1: error notification)**

Item		Details
D	Completion notification	Turns OFF when execution of the function starts, and turns ON when the directory has finished being made. Turns ON whichever way execution of the function ends, successfully or in error.
D+1	Error notification	Turns OFF when execution of the function ends successfully, and turns ON when execution ends in error.

Point

- **Several scans are sometimes required for making the directory.**
- **The notification bit data is written by END processing after the function is executed.**
- **The storage location when indirect specification is performed or index modification is specified for the notification bit data is the storage location at the rising edge of the execution condition.**
- **The MMKDIR function is not executed while a storage device function is being executed (CR3214 is ON).**
- **The function is not executed when any of the following conditions is established:**
  - **When the text string specified by S does not contain the end code (NUL (00H))**
  - **When the text string specified by S exceeds 247 characters (including end code "NUL (00H)")**
  - **When a timer/counter is specified to S by indirect specification.**
  - **When a bit other than the start of the channel is specified when bit data is specified for S.**
- **This instruction cannot be used in interrupt programs or in the initialization module.**

Hint

Several scans are sometimes required to write the directory for the MMKDIR function. When the value of the created path name is changed during make directory processing, the directory make result is not assured.
Therefore, it is necessary to use special device CR3214 (Storage instruction executing) or result (end notification) of folder creation process as execution condition and perform exclusive control.

Programmed Script

```
IF MR1000 AND R5000 THEN
    RES ( R5000 )
    MR2000 = MMKDIR ( DM1000, R5000 )
END IF
IF MR2000 = OFF THEN
    SET ( R5000 )
END IF
```

Operation Description

When MR1000 (DIR make request) is ON, and R5000 is ON (previous DIR make request is completed) is ON, the creation of the directory this time is executed.
R5000 is turned OFF (reset for judgment of completion of the DIR make request this time), and the directory is made on the memory card using the path name (text string) stored in DM1000.
The make result is stored in R5000 (notification of DIR make completion this time) and R5001 (DIR make result notification this time).
When startup of the MMKDIR function fails (MR2000＝OFF), R5000 is turned ON to start up the MMKDIR function again.

MEMO

MWRIT
MREAD
MFREE
MFREEK
MMKDIR
MRMDIR
MDEL
MPRINT
MREADL
MCOPY
MMOV
MREN
MSTAT

MRMDIR Delete storage device (memory card/CPU memory) directory

Delete storage device directory

Return value*1 = MRMDIR (delete path name, delete result storage destination)

Argument/Return Value	Description	Supported Type								Const	Device / Variable	Formula
		.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S Delete path name*3*5	Specifies the storage location for the path name of the directory to delete or the deleted directory path name. (A text string in the range 1 to 247 characters (including end code NUL) is specified.)	—	—	—	—	—	—	—	.T	*2 ○	○	○
D Delete result storage destination*4	Specifies the destination to notify the delete result.	—	—	—	—	—	—	.B	—	—	○	—
R Return value	Turns ON when the function is successfully started up, and turns OFF when startup fails.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work relay.
*2 To specify a text string using constants, enclose with " ".
*3 When specifying bit data, specify the start of the channel.
*4 When bit data is specified, it occupies 2 continuous bits. When word data is specified, the LSB of each of the 2 continuous words is used as notification bit data. Bits other than the LSB do not change.
*5 T, C, CR, CTC, CTH, Z cannot be used.

Point **This function cannot be used with KV-N14□□/N24□□/N40□□/N60□□. (A conversion error will not occur in KV STUDIO, only a warning is displayed).**

Description of Operation

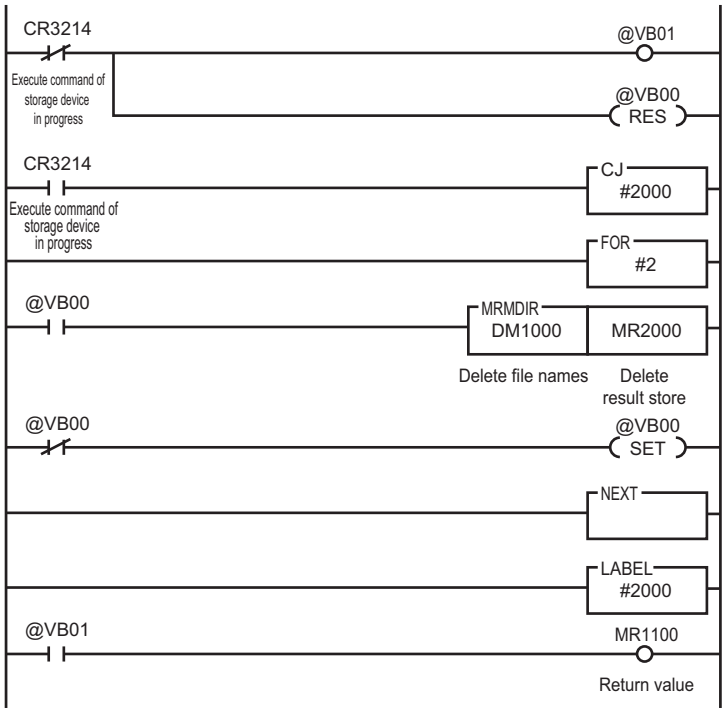
The directory with the path name (text string) specified by S is deleted from the memory card or CPU memory.
The delete result is stored in D.

- When the directory specified by S does not exist in the storage device
  [D+1] will be ON, and 3 is stored in CM2390 (file or directory does not exist).

- When file and directory exist in the directory specified by S
  Delete all files and directories existent in the directory, and then delete the specified directory.
  "About operation flag (CR2009 to CR2012)", page 1-18
  For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Storage Instructions".

● **Example of format**

MR1100 = **MRMDIR** (DM1000.T, MR2000)



● **Notification bit storage (D: completion notification, D+1: error notification)**

Item		Details
D	Completion notification	Turns OFF when execution of the function starts, and turns ON when the directory has finished being deleted. Turns ON whichever way execution of the function ends, successfully or in error.
D+1	Error notification	Turns OFF when execution of the function ends successfully, and turns ON when execution ends in error.

Point

- **Several scans are sometimes required for deleting the directory.**
- **The notification bit data is written by END processing after the function is executed.**
- **The storage location when indirect specification is performed or index modification is specified for the notification bit data is the storage location at the rising edge of the function execution.**
- **The MRMDIR function is not executed while a storage device function is being executed (CR3214 is ON).**
- **The function is not executed when any of the following conditions is established:**
  - **When the text string specified by S does not contain the end code (NUL (00H))**
  - **When the text string specified by S exceeds 247 characters (including end code "NUL (00H)")**
  - **When a timer/counter is specified to S by indirect specification.**
  - **When a bit other than the start of the channel is specified when bit data is specified for S.**
- **This instruction cannot be used in interrupt programs or in the initialization module.**

MDEL Delete storage device (memory card/CPU memory) file

Return value*1 = MDEL (delete file name, delete result storage destination)

Delete storage device file

Argument/Return Value	Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S Delete file name*3*5	Specifies the storage location for the file name to delete or the deleted file name. (A text string in the range 1 to 247 characters (including end code NUL) is specified.)	—	—	—	—	—	—	—	.T	*2 ○	○	○
D Delete result storage destination*4	Specifies the destination to notify the delete result.	—	—	—	—	—	—	.B	—	—	○	—
R Return value	Turns ON when the function is successfully started up, and turns OFF when startup fails.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work relay.
*2 To specify a text string using constants, enclose with " ".
*3 When specifying bit data, specify the start of the channel.
*4 When bit data is specified, it occupies 2 continuous bits. When word data is specified, the LSB of each of the 2 continuous words is used as notification bit data. Bits other than the LSB do not change.
*5 T, C, CR, CTC, CTH, Z cannot be used.

Point **This function cannot be used with KV-N14□□/N24□□/N40□□/N60□□. (A conversion error will not occur in KV STUDIO, only a warning is displayed).**

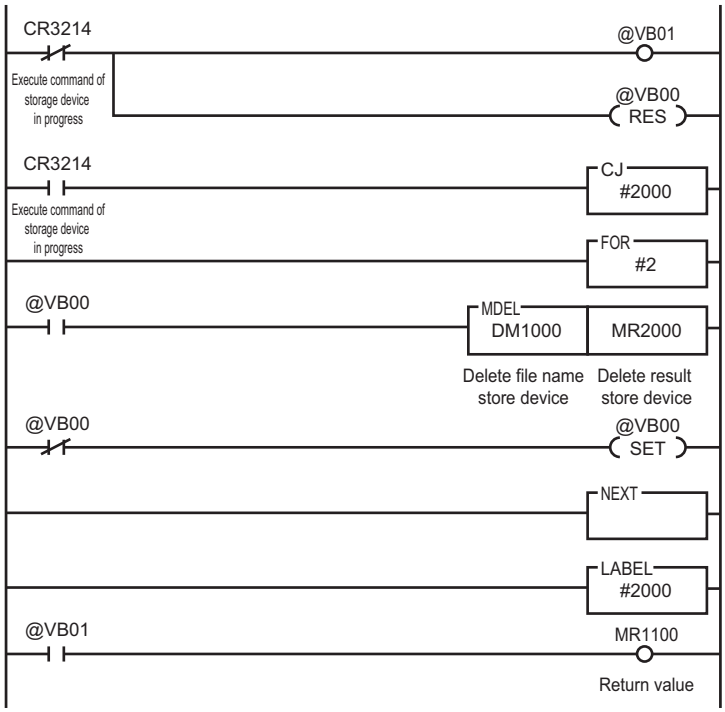
Description of Operation

The file with the file name (text string) specified by S is deleted from the memory card or CPU memory.
The delete result is stored in D.

- When the directory specified by S does not exist in the storage device [D+1] will be ON, and 3 is stored in CM2390 (file or directory does not exist).
  "About operation flag (CR2009 to CR2012)", page 1-18
  For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Storage Instructions".

● **Example of format**

MR1100 = **MDEL** (DM1000.T, MR2000)



● **Notification bit storage (D: completion notification, D+1: error notification)**

Item		Details
D	Completion notification	During function operation, OFF; file deletion is complete, ON. Turns ON whichever way execution of the function ends, successfully or in error.
D+1	Error notification	Turns OFF when execution of the function ends successfully, and turns ON when execution ends in error.

Point

- **Several scans are sometimes required for deleting the directory.**
- **The notification bit data is written by END processing after the function is executed.**
- **The storage location when indirect specification is performed or index modification is specified for the notification bit data is the storage location at the rising edge of the function execution.**
- **The MDEL function is not executed while a storage device function is being executed (CR3214 is ON).**
- **The function is not executed when any of the following conditions is established:**
  - **When the text string specified by S does not contain the end code (NUL (00H))**
  - **When the text string specified by S exceeds 247 characters (including end code "NUL (00H)")**
  - **When a timer/counter is specified to S by indirect specification.**
  - **When a bit other than the start of the channel is specified when bit data is specified for S.**
- **This instruction cannot be used in interrupt programs or in the initialization module.**

MPRINT Write text to a file in storage device (memory card/CPU memory)

Write text string to storage device

Return value*1 = MPRINT (save file name, write text string, writing result storage destination)

Argument/Return Value		Description	Supported Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Save file name*2	Specifies the storage location for the file name for data saving or the save file name. (A text string in the range 1 to 247 characters (including end code NUL) is specified.)	.U	.S	.D	.L	.F	.DF	.B	.T	○	○	—
S2	Write result text string*2	Specifies the start that stores the text string to be written or the text string to be written.	.U	.S	.D	.L	.F	.DF	.B	.T	○	○	—
D	Writing result storage destination	Specifies the bit data to notify the writing result.	—	—	—	—	—	—	.B	—	—	○	—
R	Return value	Turns ON when the function is successfully started up, and turns OFF when startup fails.	.U	.S	.D	.L	.F	.DF	—	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work device / work memory.
*2 When bit data is specified for the write result storage destination, 2 continuous bits are used as the notification bit.
*3 To specify a text string using constants, enclose with " ".
*4 When specifying bit data, specify the start of the channel.
*5 T, C, CTC, CTH, Z cannot be used.

Point

- **The MPRINT function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can be specified only with KV-8000/7000 Series CPU unit, KV-5500/ 5000/ 3000 Series CPU unit with CPU function version 2.0 or higher, or KV-NC32T.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **This function cannot be used with KV-N14□□/N24□□/N40□□/N60□□. (A conversion error will not occur in KV STUDIO, only a warning is displayed).**

Description of Operation

The text string specified by S2 is written to the memory card at the end of the file name (text string) specified by S1.
When the S1 name does not exist, a new file will be created.
D stores the write completion notification and the error notification of abnormal end.
R is turned ON when the MPRINT function is started normally, and OFF when it fails.

Reference

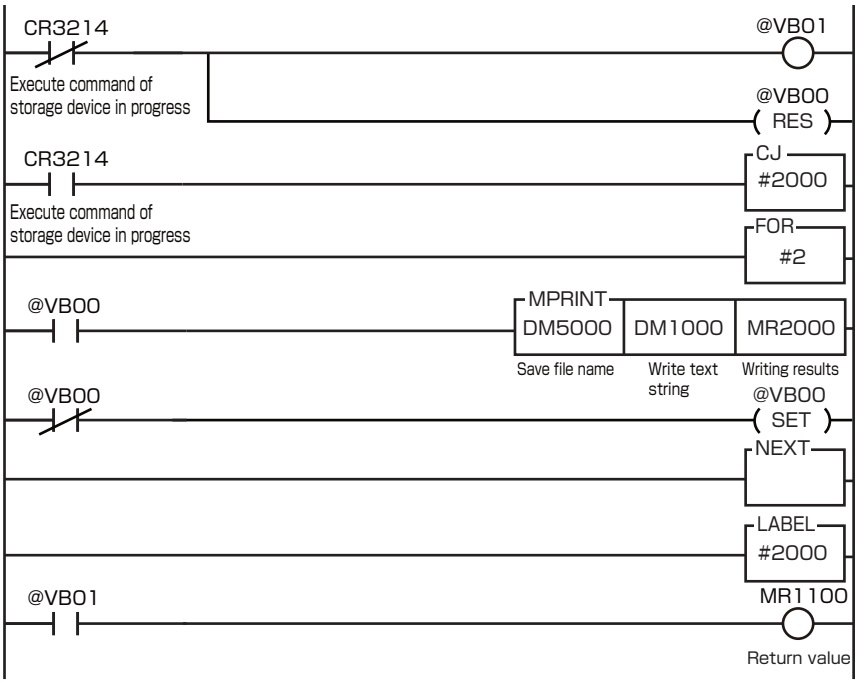
The notification bits of D have the following meanings.
(Example) When MR2000 has been specified

Notification Bit		ON (1, TRUE)	OFF (0, FALSE)
MR2000	Completion notification	Completed	Not completed
MR2001	Error notification	Normal end	Abnormal end

"About operation flag (CR2009 to CR2012)", page 1-18
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Storage Instructions".

● **Example of format**

MR1100 = **MPRINT** (DM5000.T, DM1000.T, MR2000)



MWRIT
MREAD
MFREE
MFREEK
MMKDIR
MRMDIR
MDEL
MPRINT
MREADL
MCOPY
MMOV
MREN
MSTAT

MREADL Read one line from file in storage device (memory card/CPU memory)

Read one line from storage device

Return value*1 = MREADL (read file name, read text string storage destination, read parameter, read result storage destination)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S	Read file name*6	Specifies the storage location for the file name for data reading or the read file name. (A text string in the range 1 to 247 characters (including end code NUL and folder name) is specified.)	—	—	—	—	—	—	—	.T	*4 ○	○	○
D1	Read text string storage destination*5	Specifies the start to store the read text string.	—	—	—	—	—	—	—	.T	—	○	—
D2	Read parameter*2	Specifies the parameter to read.	.U	.U	.U	.U	—	—	—	—	—	○	—
D3	Read result storage destination*3	Specifies the bit data to notify the read result.	—	—	—	—	—	—	.B	—	—	○	—
R	Return value	Turns ON when the function is successfully started up, and turns OFF when startup fails.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work device / work memory.
*2 The read parameter occupies four continuous words.
*3 The read result storage destination uses 4 continuous bits as notification bits.
*4 To specify a text string using constants, enclose with " ".
*5 When specifying bit data, specify the start of the channel.
*6 T, C, CTC, CTH, Z cannot be used.

Point

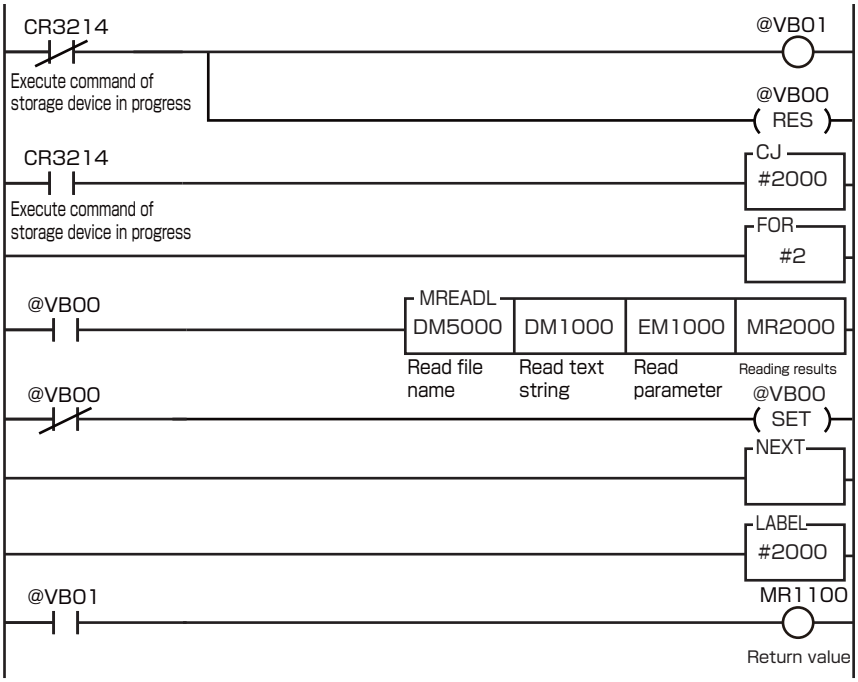
- **The MREADL function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can be specified only with KV-8000/7000 Series CPU unit, KV-5500/5000/3000 Series CPU unit with CPU function version 2.0 or higher, or KV-NC32T.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **This function cannot be used with KV-N14□□/N24□□/N40□□/N60□□. (A conversion error will not occur in KV STUDIO, only a warning is displayed).**

Description of Operation

One line is read from the file name (text string) specified by S to D1.
(The character codes which express a line are 0DH (CR), 0AH (LF) and 0D0AH (CR+LF).)
The maximum number of bytes in the read characters is set with "D2+2" (maximum number of read bytes).
If the length of text data of one line in the file is longer than that specified by "D2+2", "D3+3" (no line feed character notification) is ON, and the text data is stored in D1 up to "D2+2" bytes.
D3 stores the read completion notification and abnormal end alarm notification.
R is turned ON when the MREADL function is started normally, and OFF when it fails.
"About operation flag (CR2009 to CR2012)", page 1-18
Refer to "MREADL instruction" of "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual" for details on the D2 and D3 parameters.

● **Example of format**

MR1100 = **MREADL** (DM5000.T, DM1000.T, EM1000, MR2000)



MWRIT
MREAD
MFREE
MFREEK
MMKDIR
MRMDIR
MDEL
MPRINT
MREADL
MCOPY
MMOV
MREN
MSTAT

MCOPY Copy file in storage device (memory card/CPU memory)

Return value*1 = MCOPY (copy source file name/folder name, copy destination file name/folder name, copy format parameter, copy results storage destination)

Copy storage device file

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S1	Copy source file name/folder name*5*6	Specifies the storage location for the file name / folder name of the copy source or the file name / folder name. (A text string in the range 1 to 247 characters (including end code NUL and folder name) is specified.)	—	—	—	—	—	—	—	.T	*4 ○	○	○
S2	Copy destination file name/folder name*5*6	Specifies the storage location for the file / folder name of the copy destination or the file / folder name. (A text string in the range 1 to 247 characters (including end code NUL and folder name) is specified.)	—	—	—	—	—	—	—	.T	*4 ○	○	—
S3	Copy format parameter*2	Specifies the parameter to read.	.U	.U	.U	.U	—	—	—	—	○	○	—
D	Copy result storage destination*3	Specifies the bit data to notify the copy result.	—	—	—	—	—	—	.B	—	—	○	—
R	Return value	Turns ON when the function is successfully started up, and turns OFF when startup fails.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work device / work memory.
*2 The copy format parameter occupies one word.
*3 The copy result storage device uses two continuous bits as notification bits.
*4 To specify a text string using constants, enclose with " ".
*5 When specifying bit data, specify the start of the channel.
*6 T, C, CTC, CTH, Z cannot be used.

Point

- **The MCOPY function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can be specified only with KV-8000/7000 Series CPU unit, KV-5500/5000/3000 Series CPU unit with CPU function version 2.0 or higher, or KV-NC32T.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **This function cannot be used with KV-N14□□/N24□□/N40□□/N60□□. (A conversion error will not occur in KV STUDIO, only a warning is displayed).**

Description of Operation

The file name (text string) specified by S1 is copied to the file specified by S2.
If a file or folder with the same name is found at the copy destination, whether to overwrite the data or generate an error will be judged according to S3.

When copy format parameter value is 0: If file with same name is found, data is overwritten.
When copy format parameter value is 1: If file with same name is found, error occurs without overwriting the data.

D stores the copy completion notification and abnormal end alarm notification.
R is turned ON when the MCOPY function is started normally, and OFF when it fails.

Combinations of file and folder names which can be specified in S1 and S2

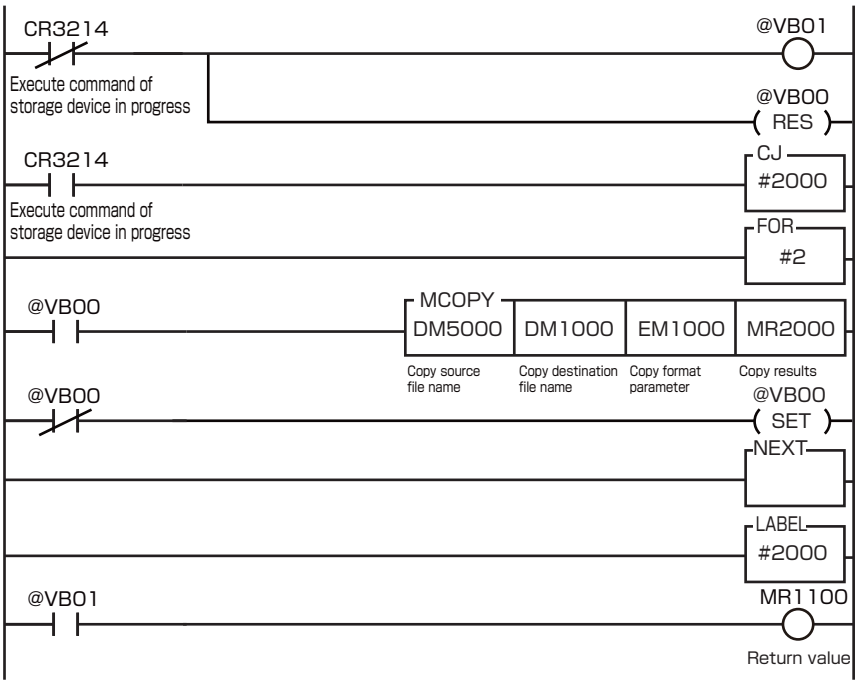
		S2	
		File name	Folder name
S1	File name	Normal operation	Normal operation
	Folder name	Error occurs	Normal operation

"About operation flag (CR2009 to CR2012)", page 1-18

Refer to "MCOPY instruction" of "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual" for details on the S3 parameter and D copy result.

● Example of format

MR1100 = **MCOPY** (DM5000.T, DM1000.T, EM1000, MR2000)



MMOV Move file in storage device (memory card/CPU memory)

Return value*1 = MMOV (move source file name/folder name, move destination file name/folder name, move format parameter, move result storage destination)

Storage device file move

Argument/Return Value	Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S1 Move source file name/folder name*5*6	Specifies the storage location for the file name / folder name of the move source or the file name / folder name. (A text string in the range 1 to 247 characters (including end code NUL and folder name) is specified.)	—	—	—	—	—	—	—	.T	*4 ○	○	○
S2 Move destination file name/folder name*5*6	Specifies the storage location for the file name / folder name of the move destination or the file name / folder name. (A text string in the range 1 to 247 characters (including end code NUL and folder name) is specified.)	—	—	—	—	—	—	—	.T	*4 ○	○	—
S3 Move format parameter*2	Specifies the move format parameter.	.U	.U	.U	.U	—	—	—	—	○	○	—
D Move result storage destination*3	Specifies the bit data to notify the move result.	—	—	—	—	—	—	.B	—	—	○	—
R Return value	Turns ON when the function is successfully started up, and turns OFF when startup fails.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work device / work memory.
*2 The move format parameter occupies one word.
*3 The move result storage device uses two continuous bits as the notification bits.
*4 To specify a text string using constants, enclose with " ".
*5 When specifying bit data, specify the start of the channel.
*6 T, C, CTC, CTH, Z cannot be used.

Point

- **The MMOV function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can be specified only with KV-8000/7000 Series CPU unit, KV-5500/5000/ 3000 Series CPU unit with CPU function version 2.0 or higher, or KV-NC32T.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **This function cannot be used with KV-N14□□/N24□□/N40□□/N60□□. (A conversion error will not occur in KV STUDIO, only a warning is displayed).**

Description of Operation

The file name (text string) specified by S1 is moved to the file specified by S2.
If a file or folder with the same name is found at the move destination, whether to overwrite the data or generate an error will be judged according to S3.

When move format parameter value is 0: If file with same name is found, data is overwritten.
When move format parameter value is 1: If file with same name is found, error occurs without overwriting the data.

D stores the move completion notification and abnormal end alarm notification.
R is turned ON when the MMOV function is started normally, and OFF when it fails.

Combinations of file and folder names which can be specified in S1 and S2

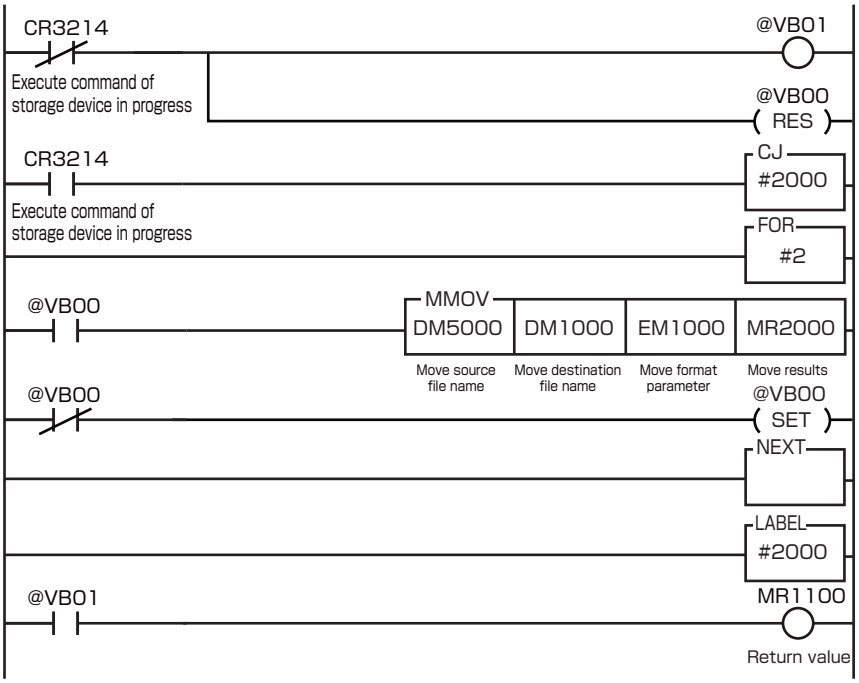
		S2	
		File name	Folder name
S1	File name	Normal operation	Normal operation
	Folder name	Error occurs	Normal operation

"About operation flag (CR2009 to CR2012)", page 1-18

Refer to "MMOV instruction" of "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual" for details on the S3 parameter and D move result.

● Example of format

MR1100 = **MMOV** (DM5000.T, DM1000.T, EM1000, MR2000)



MREN Rename file in storage device (memory card/CPU memory)

Return value*1 = MREN (file name/folder name to be renamed, new file name/folder name, file name rename result storage destination)

Storage device file rename

Argument/Return Value	Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S1 File name/folder name to be renamed*4*5	Specifies the storage location for the file name / folder name to be renamed or the file name / folder name. (A text string in the range 1 to 247 characters (including end code NUL and folder name) is specified.)	—	—	—	—	—	—	—	.T	*3 ○	○	○
S2 New file name/ folder name*4*5	Specifies the storage location for the new file / folder name or the file / folder name. (A text string in the range 1 to 247 characters (including end code NUL and folder name) is specified.)	—	—	—	—	—	—	—	.T	*3 ○	○	○
D File name rename result storage destination*2	Specifies the bit data to notify the file name rename result.	—	—	—	—	—	—	.B	—	—	○	—
R Return value	Turns ON when the function is successfully started up, and turns OFF when startup fails.	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work device / work memory.
*2 The move result storage device uses two continuous bits as the notification bits.
*3 To specify a text string using constants, enclose with " ".
*4 When specifying bit data, specify the start of the channel.
*5 T, C, CTC, CTH, Z cannot be used.

Point

- **The MREN function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can be specified only with KV-8000/7000 Series CPU unit, KV-5500/ 5000/ 3000 Series CPU unit with CPU function version 2.0 or higher, or KV-NC32T.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **This function cannot be used with KV-N14□□/N24□□/N40□□/N60□□. (A conversion error will not occur in KV STUDIO, only a warning is displayed).**

Description of Operation

The file name / folder name (text string) specified by S1 is renamed to the file name / folder name specified by S2.

D stores the rename completion notification and abnormal end alarm notification.

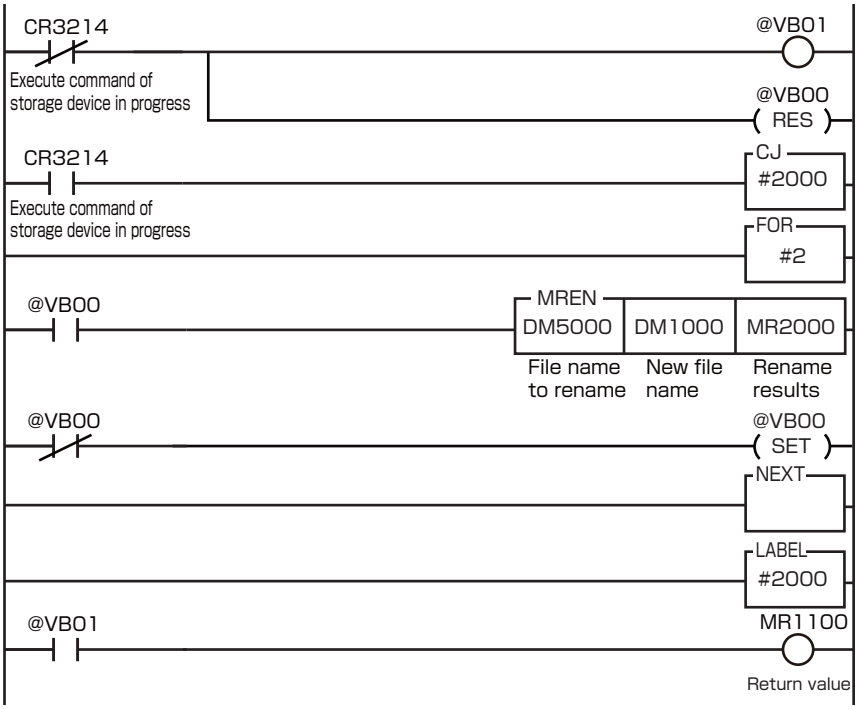
R is turned ON when the MREN function is started normally, and OFF when it fails.

"About operation flag (CR2009 to CR2012)", page 1-18

Refer to "MREN instruction" of "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual" for details on the operation.

● **Example of format**

MR1100 = **MREN** (DM5000.T, DM1000.T, MR2000)



MWRIT
MREAD
MFREE
MFREEK
MMKDIR
MRMDIR
MDEL
MPRINT
MREADL
MCOPY
MMOV
MREN
MSTAT

MSTAT Get status of file in storage device (memory card/CPU memory)

Get storage device file status

Return value*1 = MSTAT (file name, file status storage destination, retrieved result storage destination)

Argument/Return Value		Description	Supported Type .U	.S	.D	.L	.F	.DF	.B	.T	Const #/$	Device / Variable	Formula
S1	File name*5	Specifies the storage location for the name of the file for which the status is retrieved or the file name. (A text string in the range 1 to 247 characters (including end code NUL and folder name) is specified.)	—	—	—	—	—	—	—	.T	*3 ○	○	○
D1	File status storage destination*4	Specifies the destination to store the file status.	.U	.U	.U	.U	—	—	—	—	—	○	—
D2	Retrieved result storage destination*2	Specifies the bit data to notify the retrieved result.	—	—	—	—	—	—	.B	—	—	○	—
R	Return value	.U type SEG function available	—	—	—	—	—	—	.B	—	—	○	—

*1 The return value can be omitted. When omitted, the operation result is stored in a work device / work memory.
*2 The read result storage destination uses 3 continuous bits as notification bits.
*3 To specify a text string using constants, enclose with " ".
*4 The file status storage destination occupies 10 continuous words.
*5 T, C, CTC, CTH, Z cannot be used.

Point

- **The MSTAT function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can be specified only with KV-8000/7000 Series CPU unit, KV-5500/5000/ 3000 Series CPU unit with CPU function version 2.0 or higher, or KV-NC32T.**
  "KV-5500/5000/3000 CPU Function Version", page 4
- **This function cannot be used with KV-N14□□/N24□□/N40□□/N60□□. (A conversion error will not occur in KV STUDIO, only a warning is displayed).**

Description of Operation

The status of the file specified by S is retrieved and stored in D1.

D2 stores the retrieval completion notification, abnormal end alarm notification and presence of the file.

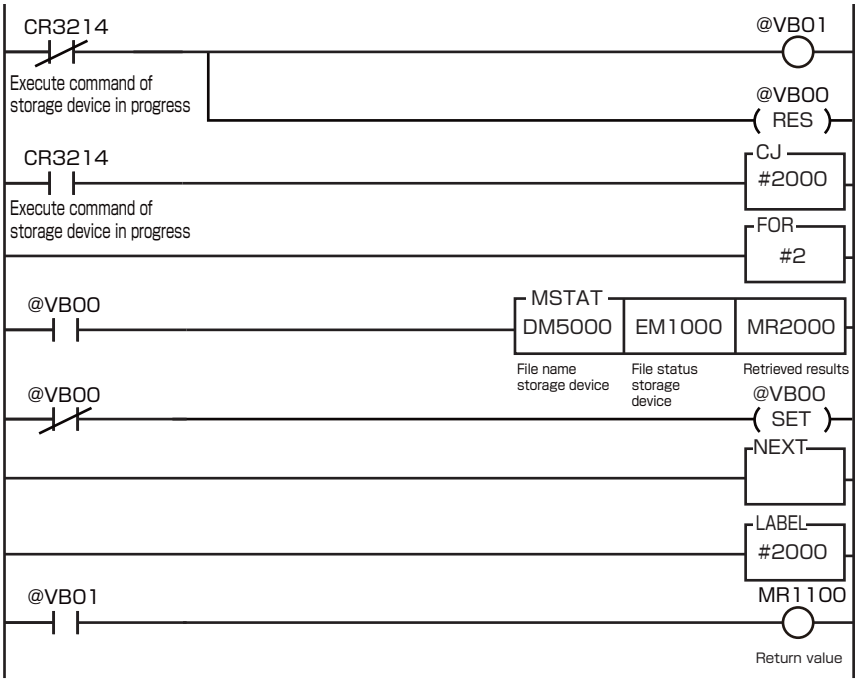
R is turned ON when the MSTAT function is started normally, and OFF when it fails.

"About operation flag (CR2009 to CR2012)", page 1-18

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "MSTAT instruction".

● **Example of format**

MR1100 = **MSTAT** (DM5000.T, EM1000, MR2000)



MWRIT
MREAD
MFREE
MFREEK
MMKDIR
MRMDIR
MDEL
MPRINT
MREADL
MCOPY
MMOV
MREN
MSTAT

AWNUM Display access window message 1

Display user message 1

AWNUM (display switching, AW write data)

Argument/Return Value		Description	Supported Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Display switching	Specifies display switching of the access window.	—	—	—	—	—	—	.B	—	—	○	○
S2	AW write data*1	Specifies the destination to store the value displayed as user message 1 in the access window, or the user message 1 data.	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 When bit data is specified, it occupies 16 continuous bits. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16 bits straddling the next channel.

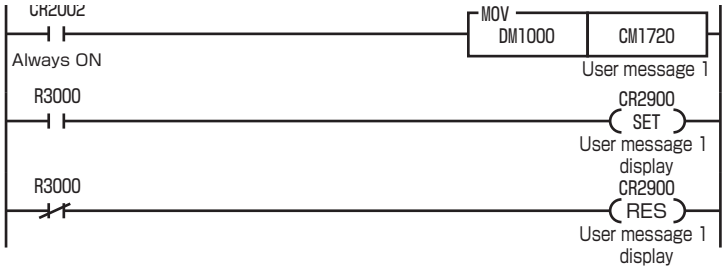
Point **There is no access window in KV-NC32T. If the expansion access window cassette is not connected with the KV-NC32T or KV Nano Series, an error will not occur and the function will not execute. (A conversion error will not occur in KV STUDIO).**

Description of Operation

When S1 is ON, the value specified by S2 is displayed as the user message 1 in the access window.
When S1 is OFF, the display of user message 1 is canceled.

● Example of format

AWNUM (R3000, DM1000)



"About operation flag (CR2009 to CR2012)", page 1-18
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Access Window Instructions".

Reference When user messages 1 and 2 are displayed simultaneously, user message 1 is given priority and displayed.

For details on user message display, see "KV-5500/5000/3000 Series User's Manual".

Point

- **When display switching is ON, CR2900 is set, and when it is OFF, CR2900 is reset. In addition, the value of [S2] is copied to CM01720 at all times when the display switching is ON, and the result is reflected immediately in the access window.**
- **If the value of [S2] is changed when display switching is ON, the display content of user message 1 is also changed.**
- **If the access window is operated even when display switching is ON, the message display is canceled (CR2900 is also reset at this time).**
  If the AWNUM function is called again, CR2900 turns ON.
- **When a project contains multiple AWNUM functions, the state of the access window conforms to the last executed AWNUM function.**
- **This instruction cannot be used in interrupt programs or in the initialization module.**

AWMSG Display access window message 2

Display user message 2

AWMSG (display switching, AW write data)

Argument/Return Value		Description	Supported Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Display switching	Specifies display switching of the access window.	—	—	—	—	—	—	.B	—	—	○	○
S2	AW write data*1*2	Specifies the destination to store the text string data to be displayed as user message 2 in the access window, or the user message 2 data.	—	—	—	—	—	—	—	.T	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 The maximum number of characters in a text string that can be displayed in the access window is 24.
- Text string constants enclosed by double quotation marks can be specified.
- When using bit data, it is necessary to specify the start of the channel.

*2 T, C, CR, CTC, CTH, Z cannot be used.

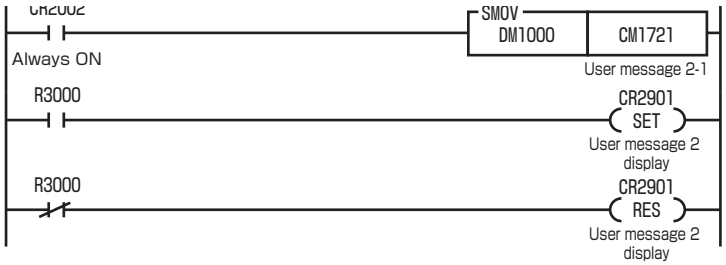
Point **There is no access window in KV-NC32T. If the expansion access window cassette is not connected with the KV-NC32T or KV Nano Series, an error will not occur and the function will not execute. (A conversion error will not occur in KV STUDIO).**

Description of Operation

When S1 is ON, the value specified by S2 is displayed as the user message 2 in the access window.
When S1 is OFF, the display of user message 2 is canceled.

● Example of format

AWMSG (R3000, DM1000.T)



"About operation flag (CR2009 to CR2012)", page 1-18
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Access Window Instructions".

Reference When user messages 1 and 2 are displayed simultaneously, user message 1 is given priority and displayed.

For details on user message display, see "KV-5500/5000/3000 Series User's Manual".

Point

- **The access window of KV-5500/5000/3000/1000 and KV Nano Series cannot display 2- byte characters.**
- **When display switching is ON, CR2901 is set, and when it is OFF, CR2901 is reset. In addition, the value of [S2] is copied to CM01721 to CM01737 at all times when the display switching of the previously stored text string is ON, and the result is reflected immediately in the access window.**
- **If the text string stored initially in [S2] is changed when display switching is ON, the display content of user message 2 is also changed.**
- **If the access window is operated even when display switching is ON, the message display is canceled (CR2901 is also reset at this time).**
  If the AWMSG function is called again, CR2900 turns ON.
- **When a project contains multiple AWMSG functions, the state of the access window conforms to the last executed AWMSG function.**
- **The text strings that can be displayed in the access window are only 1-byte characters (ASCII codes 20h to 7Eh, A1h to DFh) in 24 characters from the text string currently stored starting from [S2]. 2-byte characters such as Chinese characters cannot be displayed.**
- **This instruction cannot be used in interrupt programs or in the initialization module.**
- **The function is not executed when any of the following conditions is established:**
  - **When a bit other than the start of the channel is specified when bit data is specified for [S2].**
  - **When a timer/counter is specified to [S2] by indirect specification.**

AWSHOW User message display

User message display

AWSHOW (Execution condition, displayed user message ID)

Argument/Return Value		Description	Supported Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Execution condition	When execution conditions are turned from OFF to ON, the user message is displayed in the access window.	—	—	—	—	—	—	.B	—	—	○	○
S2	Displayed user message ID	Specifies the user message ID displayed in the access window. (1 to 63)	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

Description of Operation

When S1 turns from OFF to ON, the user message specified by S2 is displayed.

Reference: To remove user message display, operate the access window or execute the AWHIDE command.

"About operation flag (CR2009 to CR2012)", page 1-18

For details, see "KV-8000/7000/5000/3000/1000 Series / KV Nano Series Instruction Reference Manual", "Access Window Instructions", the User's Manual of the CPU unit used, or "User Message".

Point

- **When the execution condition is ON, the S2 value does not change even if altered.**
- **CR is set for user message of ID designated at the rising edge of executions conditions.**
- **If the AWSHOW function and the AWHIDE function are executed simultaneously, follow operation of the function executed last.**
- **If there are multiple AWSHOW functions in a project, the state of the access window follows the first AWSHOW function that is executed.**
- **This instruction cannot be used in interrupt programs or in the initialization module.**

AWHIDE Hide user message

Hide user message

AWHIDE (Execution condition, user message ID to be hidden)

Argument/Return Value		Description	Supported Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S1	Execution condition	When execution conditions are turned from OFF to ON, user message display is hidden from access window.	—	—	—	—	—	—	.B	—	—	○	○
S2	User message ID to be hidden	Specifies the user message ID to be hidden. (1 to 63)	.U	.U	.U	.U	—	—	—	—	○	○	○
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

Description of Operation

When S1 turns from OFF to ON, the user message specified by S2 is hidden.

"About operation flag (CR2009 to CR2012)", page 1-18

For details, see "KV-8000/7000/5000/3000/1000 Series / KV Nano Series Instruction Reference Manual", "Access Window Instructions", the User's Manual of the CPU unit used, or "User Message".

Point

- **When the execution condition is ON, the S2 value does not change even if altered.**
- **CR is set for user message of ID designated at the start of executions conditions.**
- **If the AWSHOW function and the AWHIDE function are executed simultaneously, follow operation of the function executed last.**
- **This instruction cannot be used in interrupt programs or in the initialization module.**

RFSFRC FREE RUN counter refresh

FREE RUN counter refresh

RFSFRC ([execution condition,]*1)

Argument/Return Value	Description	Supported Type								Const	Device / Variable	Formula
		.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
R Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

Description of Operation

When the execution condition is established (TRUE), FREE RUN counter is refreshed to the latest value.

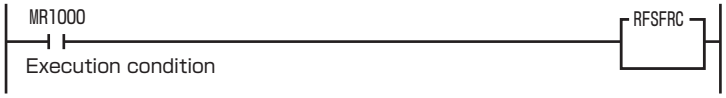
The updated free run counters are as follows.

KV-8000/7500/7300	CM1030 to CM1031: 1ms counter
	CM1032 to CM1033: 10us counter
	CM1034 to CM1035: 1us counter
	CM1036 to CM1037: 0.1us counter
KV-5500/5000/3000, KV Nano	CM708 to CM709: 1ms counter

For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "FREE RUN instruction".

● Example of format

RFSFRC (MR1000)



MEMO

SPRD Read sensor parameter

SPRD (execution condition, unit number, node address, slot number, parameter number, completion code)

Argument/Return Value		Description	Supported Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Execution condition	Specifies the bit data to be used for the execution condition.	—	—	—	—	—	—	.B	—	—	○	
n1	Unit number*1	Specifies a unit number (0 to 48, KV-8000/7500/5500 is "0").	—	—	—	—	—	—	—	—	○	—	○
n2	Node address*3	Specifies the node address (1 to 256) or storage location	.U	.U	.U	.U	—	—	—	—	○	○	○
n3	Slot number*3	Specifies the slot number or the storage destination for the slot number.	.U	.U	.U	.U	—	—	—	—	○	○	○
n4	Parameter number*3	Specifies the parameter number or the storage destination for the parameter number.	.U	.U	.U	.U	—	—	—	—	○	○	○
D	Completion code*2*4*5	Specifies the start of the device to store the completion code or parameter read result.	.U	.U	.U	.U	.F	.DF	.B	.T	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 $ (HEX specification) cannot be used. The range for KV Nano Series unit numbers is 0 to 15.

*2 CTC, CTH, Z cannot be used.

*3 When bit data is specified, 16 continuous bits are processed. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16 bits straddling the next channel.

*4 When specifying bit data, specify the start of the channel.

*5 The number of stored words (four words or more) differs according to the execution conditions.

Point

- **The SPRD function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can be specified only with KV-8000/7000 Series CPU unit, KV-5500/5000/3000 Series CPU unit with CPU function version 2.0 or higher, or KV Nano Series basic unit.**

"KV-5500/5000/3000 CPU Function Version", page 4
"KV Nano Series CPU Function Version", page 4

Description of Operation

When S is ON, the parameter specified in the sensor n4 of the node address n2 and slot number n3 connected to the n1 unit is read. Upon completion, the completion bit is stored in bit 0 of D, the execution failure bit in bit 1 of D, the completion code in D+1, the detailed completion code in D+2, the number of array elements in D+3 (1 if not an array), and the contents of the parameters read in D+4 onward.

D ～: Execution result storage location

Result storage location			Details
D	Bit 0	Completion bit	This device turns OFF when execution of the instruction starts, and turns ON when reading of the parameter is finished. This device turns ON regardless of whether the instruction ends normally or abnormally.
	Bit 1	Execution failed bit	If execution of the instruction fails, this device turns ON simultaneously with the completion bit. This device turns OFF when execution of the instruction starts.
	Bits 2 to 15	Reserved for system	OFF (fixed value)
D +1	Completion code		The completion code is stored when reading of the parameters is completed. (0 when normal)
D +2	Detailed completion code		The detailed completion code is stored when reading of the parameters is completed.
D +3	Number of array elements		The number of array elements in the read parameter is stored. 1 is stored if the parameter is not an array, and 0 is stored if the instruction ends abnormally.
D ＋4~	Read results		The parameter value is stored. The number of data items occupied differs according to the read parameter.

Point **The execution will fail if an "EtherNet/IP device reserved" sensor is specified.**

If the sensor parameter read instruction is executed at the rising edge of the execution conditions, the process will continue even if the input conditions are turned OFF at the next scan.
The completion bit and failure bit of the result storage location are turned ON at the completion and turned OFF at the start of execution.
When the instruction is executed, the number of sensor setting instruction executions is incremented by 1, and is decremented when the execution finishes.
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Sensor Setting Instructions".

● Example of format

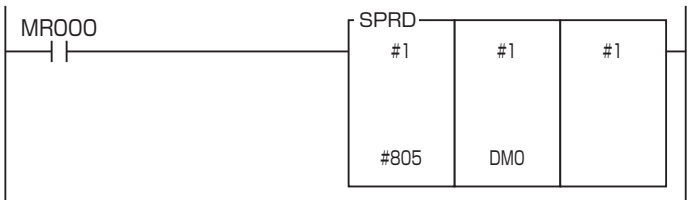
Programmed Script

SPRD (MR0,1,1,1,805,DM0)

Operation Description

When MR000 turns ON, the contents of parameter No. 805 for the node address 1, slot No. 1 sensor connected to the first connected unit are read out and stored in DM0 and following.

Ladder conversion



SPWR Sensor parameter write

SPWR (execution condition, unit number, node address, slot number, parameter number, completion code)

Argument/Return Value		Description	Supported Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Execution condition	Specifies the bit data to be used for the execution condition.	—	—	—	—	—	—	.B	—	—	○	
n1	Unit number*1	Specifies a unit number (0 to 48, KV-8000/7500/5500 is "0").	—	—	—	—	—	—	—	—	○	—	○
n2	Node address*3	Specifies the node address (1 to 256) or storage location	.U	.U	.U	.U	—	—	—	—	○	○	○
n3	Slot number*3	Specifies the slot number or the storage destination for the slot number.	.U	.U	.U	.U	—	—	—	—	○	○	○
n4	Parameter number*3	Specifies the parameter number or the storage destination for the parameter number.	.U	.U	.U	.U	—	—	—	—	○	○	○
D	Completion code*2*4*5	Specifies the start of the device to store the completion code or parameter read result.	.U	.U	.U	.U	.F	.DF	.B	.T	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 $ (HEX specification) cannot be used. The range for KV Nano Series unit numbers is 0 to 15.

*2 CTC, CTH, Z cannot be used.

*3 When bit data is specified, 16 continuous bits are processed. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16 bits straddling the next channel.

*4 When specifying bit data, specify the start of the channel.

*5 The number of stored words (four words or more) differs according to the execution conditions.

Point

- **The SPWR function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can be specified only with KV-8000/7000 Series CPU unit, KV-5500/5000/3000 Series CPU unit with CPU function version 2.0 or higher, or KV Nano Series basic unit.**

"KV-5500/5000/3000 CPU Function Version", page 4
"KV Nano Series CPU Function Version", page 4

Description of Operation

When S is ON, the setting value specified in D+4 onward is written into the parameter specified in the sensor n4 of the node address n2 and slot number n3 connected to the n1 unit. Upon completion, the completion bit is stored in bit 0 of D, the execution failure bit in bit 1 of D, the completion code in D+1,and the detailed completion code in D+2.

D ～: Execution result storage location

Result storage location			Details
D	Bit 0	Completion bit	This device turns OFF when execution of the instruction starts, and turns ON when writing of the parameter is finished. This device turns ON regardless of whether the instruction ends normally or abnormally.
	Bit 1	Execution failed bit	If execution of the instruction fails, this device turns ON simultaneously with the completion bit. This device turns OFF when execution of the instruction starts.
	Bits 2 to 15	Reserved for system	OFF (fixed value)
D +1	Completion code		The completion code is stored when writing of the parameter is completed. (0 when normal)
D +2	Detailed completion code		The detailed completion code is stored when writing of the parameter is completed.
D +3	Reserved for system		0 (fixed value
D ＋4～	Setting value		The setting value is stored. The number of data items occupied differs according to the write parameter.

Point **The execution will fail if an "EtherNet/IP device reserved" sensor is specified.**

If the sensor parameter write instruction is executed at the rising edge of the execution conditions, the process will continue even if the input conditions are turned OFF at the next scan.
The completion bit and failure bit of the result storage location are turned ON at the completion and turned OFF at the start of execution.
When the instruction is executed, the number of sensor setting instruction executions is incremented by 1, and is decremented when the execution finishes.
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Sensor Setting Instructions".

● Example of format

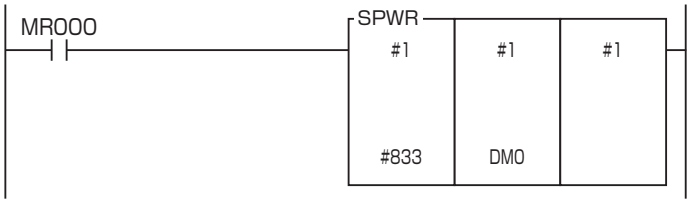
Programmed Script

SPWR (MR0,1,1,1,833,DM0)

Operation Description

When MR000 turns ON, the setting value specified with DM4 and following is written to parameter No. 833 (setting value) of the node address 1, slot No. 1 sensor connected to the unit No. 1 KV-EP21V. The results are stored in DM0 and following.

Ladder conversion



SSVC Sensor service execution

SSVC (execution condition, unit number, node address, slot number, parameter number, completion code)

Argument/Return Value		Description	Supported Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Execution condition	Specifies the bit data to be used for the execution condition.	—	—	—	—	—	—	.B	—	—	○	
n1	Unit number*1	Specifies a unit number (0 to 48, KV-8000/7500/5500 is "0").	—	—	—	—	—	—	—	—	○	—	○
n2	Node address*3	Specifies the node address (1 to 256) or storage location	.U	.U	.U	.U	—	—	—	—	○	○	○
n3	Slot number*3	Specifies the slot number or the storage destination for the slot number.	.U	.U	.U	.U	—	—	—	—	○	○	○
n4	Parameter number*3	Specifies the parameter number or the storage destination for the parameter number.	.U	.U	.U	.U	—	—	—	—	○	○	○
D	Completion code*2*4*5	Specifies the start of the device to store the completion code or parameter read result.	.U	.U	.U	.U	.F	.DF	.B	.T	—	○	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 $ (HEX specification) cannot be used.

*2 CTC, CTH, Z cannot be used.

*3 When bit data is specified, 16 continuous bits are processed. When a relay other than the leading relay (e.g. R002 or R1012) of the channel has been specified, the relay is handled in 16 bits straddling the next channel.

*4 When specifying bit data, specify the start of the channel.

*5 Up to six words are used depending on the specified service.

Point

- **The SSVC function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can be specified only with KV-8000/7000 Series CPU unit, KV-5500/5000/3000 Series CPU unit with CPU function version 2.0 or higher, or KV Nano Series basic unit.**

"KV-5500/5000/3000 CPU Function Version", page 4
"KV Nano Series CPU Function Version", page 4

Description of Operation

S When the execution condition is ON, the service specified with the sensor n4 of the node address n2 and slot No. n3 connected to the n1 unit is executed. Upon completion, the completion bit is stored in bit 0 of D, the execution failed bit in bit 1 of D, the completion code in D+1, and the detailed completion code in D+2.

D ～: Execution result storage location

Result storage location			Details
D	Bit 0	Completion bit	This device turns OFF when execution of the instruction starts, and turns ON when execution of the service is finished. This device turns ON regardless of whether the instruction ends normally or abnormally.
	Bit 1	Execution failed bit	If execution of the instruction fails, this device turns ON simultaneously with the completion bit. This device turns OFF when execution of the instruction starts.
	Bits 2 to 15	Reserved for system	OFF (fixed value)
D +1	Completion code		The completion code is stored when execution of the service is completed. (0 when normal)
D +2	Detailed completion code		The detailed completion code is stored when execution of the service is completed.
D +3	Reserved for system		0 (fixed value
D +4 D +5	Send data		Specifies the send data (maximum two words) according to the service being executed.

Point **The execution will fail if an "EtherNet/IP device reserved" sensor is specified.**

If the sensor service read instruction is executed at the rising edge of the execution conditions, the process will continue even if the input conditions are turned OFF at the next scan.
The completion bit and failure bit of the result storage location are turned ON at the completion and turned OFF at the start of execution.
When the instruction is executed, the number of sensor setting instruction executions is incremented by 1, and is decremented when the execution finishes.
For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Sensor Setting Instructions".

● Example of format

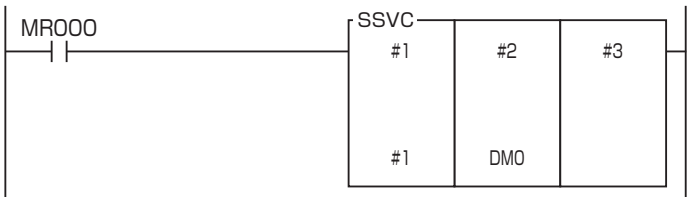
Programmed Script

SSVC (MR0,1,2,3,1,DM0)

Operation Description

When MR000 turns ON, the service No. 1 service of the node address 2, slot No. 3 sensor connected to the unit No. 1 KV-EP21V is executed. The results are stored in DM0 and following.

Ladder conversion



RFSCI Refresh cyclic communication input

RFSCI ([execution condition,][*1] start, number of data items)

Argument/Return Value		Description	Supported Type								Const	Device / Variable	For mula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Start[*2]	Specifies the start to refresh the cyclic communication input data.	.U	.S	.D	.L	.F	—	—	—	—	○	—
n	Number of data items	Specifies the number of data items to refresh.	.U	.U	.U	.U	—	—	—	—	○	—	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When specifying bit data, specify the start of the channel.

Point

- **The RFSCI function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can be specified only with KV-8000/7000 Series CPU unit, KV-5500/5000/3000 Series CPU unit with CPU function version 2.0 or higher, or KV Nano Series basic unit.**

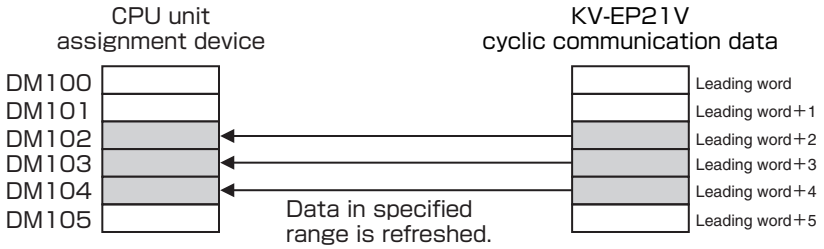
"KV-5500/5000/3000 CPU Function Version", page 4
"KV Nano Series CPU Function Version", page 4

Description of Operation

When the execution condition is ON, the cyclic communication input data from the data assigned in S to the number of data items specified by n is refreshed. (When the start is 32-bit data, the number of refreshed words is the number of data × 2.)

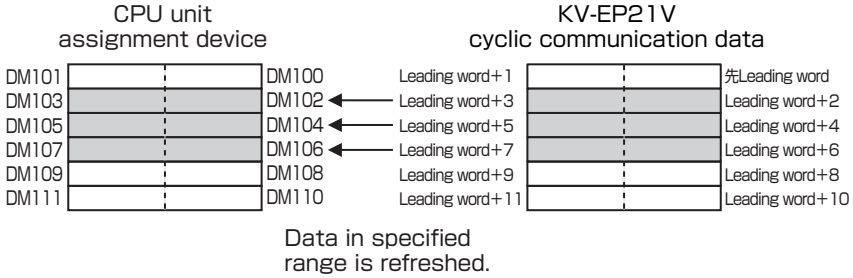
- S : When the start is 16 bits (.U/.S)

  Example) When the start is specified as DM102 (.U) and the number of data is specified as 3



- S : When the start is 32 bits (.D/.L/.F)

  Example) When the start is specified as DM102.D and the number of data is specified as 3



For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Cyclic Communication Refresh Instructions".

● Example of format

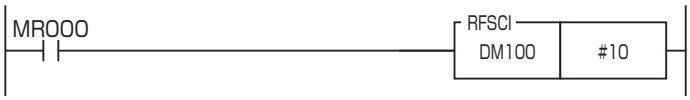
Programmed Script

RFSCI (MR0,DM100.U,10)

Operation Description

When MR000 turns ON, 10 words of cyclic communication input data assigned after data memory DM100 is refreshed.

Ladder conversion



RFSCO Refresh cyclic communication output

RFSCO ([execution condition,][*1] start, number of data items)

Argument/Return Value		Description	Supported Type								Const	Device / Variable	Formula
			.U	.S	.D	.L	.F	.DF	.B	.T	#/$		
S	Start[*2]	Specifies the start to refresh the cyclic communication output data.	.U	.S	.D	.L	.F	—	—	—	—	○	—
n	Number of data items	Specifies the number of data items to refresh.	.U	.U	.U	.U	—	—	—	—	○	—	—
R	Return value	None	—	—	—	—	—	—	—	—	—	—	—

*1 "[ ]" can be omitted. (When the execution condition has been omitted, the instruction is executed at all times (at every scan).)

*2 When specifying bit data, specify the start of the channel.

Point

- **The RFSCO function can be used only with KV STUDIO Ver. 6.00 or higher.**
- **This function can be specified only with KV-8000/7000 Series CPU unit, KV-5500/5000/3000 Series CPU unit with CPU function version 2.0 or higher, or KV Nano Series basic unit.**

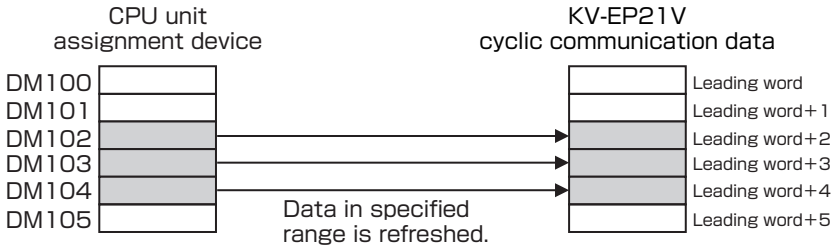
"KV-5500/5000/3000 CPU Function Version", page 4
"KV Nano Series CPU Function Version", page 4

Description of Operation

When the execution condition is ON, the cyclic communication output data from the data assigned in D to the number of data items specified by n is refreshed. (When the start is 32-bit data, the number of refreshed words is the number of data × 2.)

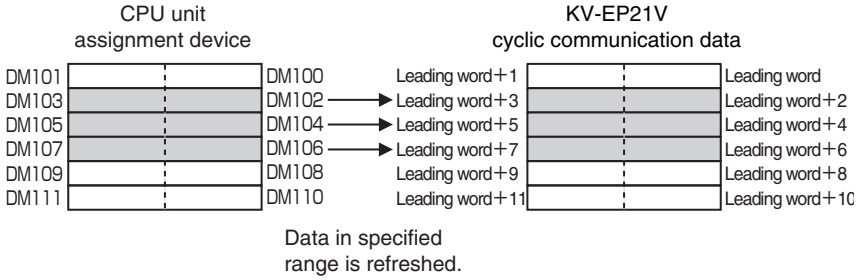
- D : When the start is 16 bits (.U/.S)

  Example) When the start is specified as DM102 (.U) and the number of data is specified as 3



- D : When the start is 32 bits (.D/.L/.F)

  Example) When the start is specified as DM102.D and the number of data is specified as 3



For details, please refer to "KV-8000/7000/5000/3000/1000 Series and KV Nano Series Instruction Reference Manual", "Cyclic Communication Refresh Instructions".

● Example of format

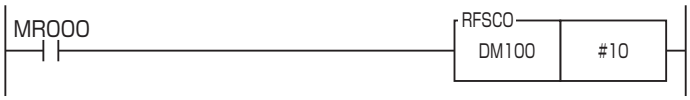
Programmed Script

RFSCO (MR0,DM100.U,10)

Operation Description

When MR000 turns ON, 10 words of cyclic communication output data assigned after data memory DM100 is refreshed.

Ladder conversion



MEMO

APPENDICES

CR/CM List ····· A-2
ASCII Code Table ····· A-53
List of Characters That Cannot Be Used ····· A-54
List of Unusable Functions ····· A-58
Index ····· A-59
Functions Index ····· A-63

CR/CM List

Control Relays (CR) (For KV-8000)

Control relays (CR0000 - CR7915) can be used to perform operations such as checking the operation status of the CPU unit and setting and monitoring the built-in functions.
Control relays reserved for the system cannot be used.

CR	Attribute R: Read only Blank: Read/write	Function	
CR0000		Logging or tracing function (ID0)	Logging and tracing execution (ON: enabled, OFF: disabled)
CR0001	R		Logging or tracing operation in progress
CR0002	R		Logging or tracing file writing completion
CR0003	R		Logging buffer less than 50% warning
CR0004	R		Logging buffer overflow
CR0005	R		Data to be written in RUN mode during logging present
CR0006 and CR0007	—		Reserved for the system
CR0008	R		Logging or tracing error
CR0009	R		Logging trace No memory space error
CR0010	R		Logging or tracing device or trigger setting error
CR0011	R		Tracing file save trigger monitor
CR0012	R		Tracing data capture completion
CR0013	R		Logging or tracing setting to be written present
CR0014 and CR0015	—		Reserved for the system
CR0100 to CR0915		Logging or tracing function (ID1 to ID9)	
CR1000 to CR1015		Real-time chart monitor (ID10)	
CR1100 to CR2001	—	Reserved for the system	
CR2002	R	Always turned ON	
CR2003	R	Always turned OFF	
CR2004	R	10 ms clock pulse (duty cycle: 50%)	
CR2005	R	100 ms clock pulse (duty cycle: 50%)	
CR2006	R	1 s clock pulse (duty cycle: 50%)	
CR2007	R	Turned OFF for just one scan when operation starts	
CR2008	R	Turned ON for just one scan when operation starts	
CR2009		Turned ON when the calculation result is negative or when an overflow occurs	
CR2010	R	Turned ON when the calculation result is zero	
CR2011	R	Turned ON when the calculation result is positive	
CR2012	R	Turned ON when a calculation execution error occurs	
CR2013 to CR2302	—	Reserved for the system	
CR2303		Fixed scan time operation (ON: enabled, OFF: disabled)	
CR2304	R	Turned ON for just one scan when the scan time setting (the fixed scan time operation and END processing time settings) is exceeded	
CR2305	—	Reserved for the system	
CR2306	R	Indicates backup battery errors (ON: error, OFF: normal)	
CR2307		Clears the minimum and maximum scan time values	
CR2308		Sets the END processing time (ON: enabled, OFF: disabled; if CR2303 is turned ON at the same time, CR2303 is given priority)	
CR2309 and CR2310	—	Reserved for the system	
CR2311	R	Indicates RTC low voltage error (ON: error has occurred, OFF: normal)	
CR2312	—	Reserved for the system	
CR2313		Forces switch of the VT sensor application screen (ON: switch, OFF: do not switch)	
CR2314	R	Project password authentication status (ON: authentication disabled, OFF: authentication enabled)	
CR2315 to CR2813	—	Reserved for the system	

CR	Attribute R: Read only Blank: Read/write	Function
CR2814		Zero suppression during execution of DASC and FASC instructions (all suffixes; ON: execute, OFF: do not execute)
CR2815		Omission of "+" during execution of DASC.S, DASC.L, and FASC instructions (ON: execute, OFF: do not execute)
CR2900 to CR3005	—	Reserved for the system
CR3100	R	Inter-unit synchronization refreshing time exceeded
CR3101 to CR3115	—	Reserved for the system
CR3200		CPU memory save save request
CR3201	R	CPU memory save being performed
CR3202	R	CPU memory save complete
CR3203	R	CPU memory save error
CR3204 to CR3206	—	Reserved for the system
CR3207		Instruction RAM mode operation The RAM mode activation of file register instructions, storage device instructions towards the CPU memory. (ON: enable, OFF: disable)
CR3208 to CR3209	—	Reserved for the system
CR3210	R	Arbitrary logging/trace operation
CR3211 to CR3213	—	Reserved for the system
CR3214	R	Storage instruction executing
CR3215	—	Reserved for the system
CR3300 to CR3415		Alarm relay
CR3500	R	Alarm operation relay (ON: one of CR3300 to CR3415 is turned ON)
CR3501		Record an entry in the log when an alarm relay switches from ON to OFF (ON: record, OFF: do not record)
CR3502		The alarm relay log is cleared when this switches from OFF to ON
CR3503 to CR3512	—	Reserved for the system
CR3513	R	Logging internal buffer overflow
CR3514		HKEY instruction - Disables multiple keys
CR3515	R	HKEY instruction - Scan completion
CR3600 to CR3615	R	HKEY information storage area
CR3700 to CR3901	—	Reserved for the system
CR3902	R	Indicates that the serious error log is full
CR3903	R	Indicates that the minor error log is full
CR3904	R	RUN-PRG selector switch status (ON: RUN, OFF: PROG)
CR3905	R	A serious error has occurred
CR3906	R	A minor error has occurred
CR3907 to CR3908	—	Reserved for the system
CR3909		Clears the current error (Set this relay in the ladder program)
CR3910		Clear all error and event history when OFF→ON
CR3911 to CR4000	—	Reserved for the system
CR4001 to CR4300	R	Expansion unit status (expansion units 1 - 48)
CR4301 to CR4315	—	Reserved for the system
CR4400	R	AW up key being pressed
CR4401	R	AW down key being pressed
CR4402	R	AW left key being pressed
CR4403	R	AW right key being pressed
CR4404	R	AW MENU key being pressed
CR4405	R	AW ENT key being pressed
CR4406 to CR4414	—	Reserved for the system
CR4415	R	AW user message being displayed
CR4500		AW line 1 blinking
CR4501		AW line 2 blinking
CR4502		AW line 3 blinking
CR4503		AW line 4 blinking

CR	Attribute R: Read only Blank: Read/write	Function	
CR4504 to CR4600	—	Reserved for the system	
CR4601 to CR4915		AW user message 1 to 63	
CR5000 to CR5315	—	Reserved for the system	
CR5400		Memory card mount request	
CR5401		Memory card unmount request	
CR5402	R	Memory card used	
CR5403	R	Memory card recognized	
CR5404	R	Memory card is available	
CR5405	R	Memory card is write protected	
CR5406 to CR5407	—	Reserved for the system	
CR5408		CPU memory mount request	
CR5409		CPU memory unmount request	
CR5410	R	CPU memory used	
CR5411	R	CPU memory recognized	
CR5412 to CR4915	—	Reserved for the system	
CR5500		Project load execution request	
CR5501	—	Reserved for the system	
CR5502		Project save execution request	
CR5503	—	Reserved for the system	
CR5504	R	Project load execution complete	
CR5505	R	Project load execution failure	
CR5506	R	Project save execution complete	
CR5507	R	Project save execution failure	
CR5508		Recipe loading request	
CR5509	—	Reserved for the system	
CR5510	R	Recipe loading completed	
CR5511	R	Recipe loading failed	
CR5512		Recipe saving request	
CR5513	—	Reserved for the system	
CR5514	R	Recipe saving completed	
CR5515	R	Recipe saving failed	
CR5600		Operation record function(ID0)	Operation record 0 Save request
CR5601	R		Operation record 0 Operating
CR5602	R		Operation record 0 Save trigger monitor
CR5603	R		Operation record 0 Saving
CR5604	R		Operation record 0 Collecting
CR5605	R		Operation record 0 Setting error
CR5606	—		Reserved for the system
DR5607	R		Operation record 0 Save completed
CR5608	R		Operation record 0 Save error
CR5609	R		Device record 0 Buffer shortage
CR5610 to CR5613	—		Reserved for the system
CR5614			Operation record 0 Buffer area full save mode
CR5615	—		Reserved for the system
CR5700 to CR5919		Operation record function(ID1 to ID3)	
CR6000 to CR7915		Reserved for the system	

Control Memory Entries CM (For KV-8000)

Control memory entries (CM0000 - CM7599) can be used to perform operations such as checking the operation status of the CPU unit and setting and monitoring the built-in functions.
Control memory entries reserved for the system cannot be used.

CM	Attribute R: Read only Blank: Read/write	Function	Hold Yes: Hold
CM0000 and CM0699	—	Reserved for the system	
CM0700	R	Read from RTC (year)	
CM0701	R	Read from RTC (month)	
CM0702	R	Read from RTC (day)	
CM0703	R	Read from RTC (hour)	
CM0704	R	Read from RTC (minute)	
CM0705	R	Read from RTC (second)	
CM0706	R	Read from RTC (day of the week)	
CM0707 to CM709	—	Reserved for the system	
CM0710	R	File register current bank number	
CM0711	—	Reserved for the system	
CM0712	R	Computing error detailed information	
CM0713 to CM0899	—	Reserved for the system	
CM900	R	CPU function version	
CM901 to CM906	—	Reserved for the system	
CM907	R	Project Version	
CM908 to CM999	—	Reserved for the system	
CM1000 to CM1001	R	Scan time measuring value (1 μs) (32-bit)	
CM1002 to CM1003	R	END processing time measured value (1 μs unit) (32-bit)	
CM1004 to CM1005	R	Minimum fixed scanning time (1 μs unit) (32-bit)	
CM1006 to CM1007	R	Maximum fixed scanning time (1 μs unit) (32-bit)	
CM1008 to CM1009		Fixed scanning time run set value (1 μs unit) (32-bit)	
CM1010 to CM1011		END processing time set value (1 μs unit) (32-bit)	
CM1012 to CM1013	R	Scanning time when setting is exceeded (1 μs unit) (32-bit)	
CM1014 to CM1015	R	END processing time exceeds value (1 μs unit) (32-bit)	
CM1016 to CM1029	—	Reserved for the system	
CM1030 to CM1031	R	32-bit 1 ms Free Run counter	
CM1032 to CM1033	R	32-bit 10 μs Free Run counter	
CM1034 to CM1035	R	32-bit 1 μs Free Run counter	
CM1036 to CM1037	R	32-bit 0.1 μs Free Run counter	
CM1038 to CM1499	—	Reserved for the system	
CM1500 to CM1509	R	Log/trace (ID0 to ID9) File Save Counter	
CM1510 to CM1629	—	Reserved for the system	
CM1630 to CM1639	R	Log/trace Ring Buffer minimum free space (ID0 to ID9) (KBytes)	
CM1640 to CM1649	—	Reserved for the system	
CM1650	R	Number of sensor setting instruction executions	
CM1651 to CM1657	—	Reserved for the system	
CM1658	R	Number of consecutive project password authentication failures	Yes
CM1659	R	Total number of project password authentication failures	Yes
CM1660 to CM1669	R	Logging, trace ring buffer free capacity (ID0 - ID9) (in kbyte units)	
CM1670 to CM1679	—	Reserved for the system	
CM1680 to CM1689	R	Log/trace File Save Counter (ID0 to ID9) (Memory card use)	Yes
CM1690 to CM1699	—	Reserved for the system	
CM1700 to CM1701		Digital trimmer 0 upper limit (32-bit)	
CM1702 to CM1703		Digital trimmer 1 upper limit (32-bit)	
CM1704 to CM1705		Digital trimmer 2 upper limit (32-bit)	
CM1706 to CM1707		Digital trimmer 3 upper limit (32-bit)	
CM1708 to CM1709		Digital trimmer 4 upper limit (32-bit)	
CM1710 to CM1711		Digital trimmer 5 upper limit (32-bit)	
CM1712 to CM1713		Digital trimmer 6 upper limit (32-bit)	

CM	Attribute R: Read only Blank: Read/write	Function		Hold Yes: Hold
CM1714 to CM1715		Digital trimmer 7 upper limit (32-bit)		
CM1716 to CM1719	—	Reserved for the system		
CM1720	R	AW user message display number		
CM1721 to CM1735	—	Reserved for the system		
CM1736		AW initial display language setting		
CM1737		AW initial display comment number setting		
CM1738		Access window operation disabled setting		
CM1739	—	Reserved for the system		
CM1740 to CM1772	R	Ladder project name (up to 32 characters + end code)		
CM1773 to CM1779	—	Reserved for the system		
CM1780		Access window initial screen setting, enabled		
CM1781		Access window initial screen setting, device type		
CM1782		Access window initial screen setting, screen ID		
CM1783		Access window initial screen setting, screen ID		
CM1784		Access window initial screen setting, device number		
CM1785		Access window initial screen setting, device number		
CM1786		Access window initial screen setting, display format		
CM1787		Access window initial screen setting, device name representation		
CM1788		Access window initial screen setting, key lock status		
CM1789	—	Reserved for the system		
CM1790		Automatic loading folder number (request)		
CM1791	R	Automatic loading complete code		
CM1792	R	Automatic loading folder number (complete)		
CM1793	—	Reserved for the system		
CM1794		Run load folder number (request)		
CM1795	R	Run load complete code		
CM1796	R	Run load folder number (complete)		
CM1797 to CM1868	—	Reserved for the system		
CM1869	R	Number of connected expansion units		
CM1870	—	Recipe loading Drive number		
CM1871		Recipe loading Folder number		
CM1872		Reserved for the system		
CM1873		Recipe loading Completion code		
CM1874		Reserved for the system		
CM1875		Recipe saving Drive number		
CM1876		Recipe saving Reference folder number		
CM1877		Recipe saving Output folder number		
CM1878		Recipe saving Completion code		
CM1879		Reserved for the system		
CM1880		Project load drive number (request)		
CM1881		Project load folder number (request)		
CM1882	R	Project load drive number (complete)		
CM1883	R	Project load folder number (complete)		
CM1884	R	Project load completion code		
CM1875 to CM1877	—	Reserved for the system		
CM1888		Project save drive number (request)		
CM1889		Project save folder number (request)		
CM1890	R	Project save drive number (complete)		
CM1891	R	Project save folder number (complete)		
CM1892	R	Project save completion code		
CM1893 to CM1999	—	Reserved for the system		
CM2000 to CM2001	R	Operation record function(ID0)	Device record 0 Average sampling period	
CM2002 to CM2003	R		Device record 0 Average number of changed devices	
CM2004 to CM2005	R		Device record 0 Maximum collection period	
CM2006 to CM2009	—		Reserved for the system	
CM2010 to CM2039		Operation record function(ID1～ID3)		
CM2040 to CM2389	—	Reserved for the system		
CM2390	R	Error code for storage device instruction		

CM	Attribute R: Read only Blank: Read/write	Function		Hold Yes: Hold
CM2391		Power OFF during storage access （Becomes value other than 0 during occurrence. Please change it to 0 from the program.）		
CM2392		Memory card retry occurrence times (still hold in case of power off)		
CM2393		Memory card check error times (still hold in case of power off)		
CM2394		Memory card timeout occurrence times (still hold in case of power off)		
CM2395		Reset delay time when error occurs on the memory card		
CM2396 to CM2399	—	Reserved for the system		
CM2400 to CM2401	R	Inter-unit synchronization refreshing beginning time (0.1μs)		
CM2402 to CM2403	R	Inter-unit synchronization refreshing finishing time (0.1μs)		
CM2404 to CM2405		Inter-unit synchronization refreshing beginning time maximum (0.1μs) (To reset, please change it to 0 from this program.)		Yes
CM2406 to CM2407		Inter-unit synchronization refreshing finishing time maximum (0.1μs) (To reset, please change it to 0 from this program.)		Yes
CM2408 to CM2409	R	Inter-unit synchronization refreshing operation time (0.1μs)		
CM2410 to CM5099	—	Reserved for the system		
CM5100	R	Details of the latest serious error	Year and month (Indicated as decimal values. The last two digits show Month, and previous digits from hundred show Year. Example: 1210 → October 2012)	Yes
CM5101	R		Day	Yes
CM5102	R		Hour	Yes
CM5103	R		Minute	Yes
CM5104	R		Second	Yes
CM5105	R		Error number	Yes
CM5106	R		Number of valid pieces of data	Yes
CM5107	R		Detailed information, piece 01	Yes
CM5108	R		Detailed information, piece 02	Yes
:	R			Yes
CM5126	R		Detailed information, piece 20	Yes
CM5127 to CM5149	—	Reserved for the system		
CM5150	R	Details of the latest minor error	Year and month (Indicated as decimal values. The last two digits show Month, and previous digits from hundred show Year. Example: 1210 → October 2012)	Yes
CM5151	R		Day	Yes
CM5152	R		Hour	Yes
CM5153	R		Minute	Yes
CM5154	R		Second	Yes
CM5155	R		Error number	Yes
CM5156	R		Number of valid pieces of data	Yes
CM5157	R		Detailed information, piece 01	Yes
CM5158	R		Detailed information, piece 02	Yes
:	R			Yes
CM5176	R		Detailed information, piece 20	Yes
CM5177 to CM5871	—	Reserved for the system		
CM5872 to CM5873	R	Used amount of CPU memory（32 bit, Byte unit）		Yes
CM5874 to CM5999	—	Reserved for the system		
CM6000		Operation record function(ID0)	Operation record 0 Save completion code	
CM6001	—		Reserved for the system	
CM6002			Operation record 0 Save counter	
CM6003	—		Reserved for the system	
CM6004	R		Operation record 0 Save progress detail 1	
CM6005	R		Operation record 0 Save progress detail 2	
CM6006	R		Operation record 0 Save progress	
CM6007 to CM6009	—		Reserved for the system	
CM6010 to CM6171	R		Operation record 0 Save path	
CM6172 to CM6199	—		Reserved for the system	
CM6200 to CM6799		Operation record function (ID1 to ID3)		
CM6800 to CM7599	—	Reserved for the system		

■ Error details information - For KV-8000

- Error number 128 (calculation error)

Detailed Information, Piece n	Explanation
Detailed information 1 (CM0712/CM5157)	Details of errors 0 (Not used) 1 Incorrect indirect specification. 2 The indirectly specified target is T/C. 3 Incorrect simple indirect specification. 4 Outside of operand range. 5 Floating-point real number overflow 6 Incorrect device specification. 7 The ROOT instruction operand is negative. 8 No bits turned ON with the ENCO instruction. 9 Floating-point real number format error. 10 Incorrect trigonometric function instruction operand. 11 Data cannot be converted. 12 Improper table specification. 13 Incorrect time comparison instruction operand. 14 Incorrect cam switch operand. 15 Incorrect frequency counter instruction operand. 16 Divide by zero. 17 Illegal logging ID. 18 Direct processing error. 19 Unit on which direct processing cannot be performed. 20 Jump destination subroutine does not exist. 21 Data being written during RUN mode, so instruction cannot be executed. 22 Error for expansion unit-dedicated instructions (unit number). 23 Reserved for the system. 24 PIDAT, outside of operation setting range. 25 PIDAT, outside of sampling period range. 26 PIDAT, outside of comparison constant range. 27 PIDAT, outside of integral constant range. 28 PIDAT, outside of differential constant range. 29 PIDAT comparison constant, two-degree freedom parameters outside of range. 30 PIDAT integral constant, two-degree freedom parameters outside of range. 31 PIDAT differential constant, two-degree freedom parameters outside of range. 32 PIDAT, outside of operation quantity upper limit or lower limit range. 33 PIDAT, outside of operation quantity change width limit range. 34 PIDAT, outside of measured value change width range. 35 PIDAT, AT state transition error. 36 PIDAT, outside of operation control flag range. 37 Operation period exceeded. 38 PIDAT, outside of AT adjustment parameter range. 39 PIDAT, outside of AT hysteresis range. 40 PIDAT, outside of AT timeout range. 41 PIDAT, outside of AT calculation control mode setting range. 42 PIDAT, AT timeout. 43 PIDAT, AT comparison constant exceeded. 44 PIDAT, AT integral constant exceeded. 45 PIDAT, AT differential constant exceeded. 46 Root of negative number cannot be calculated. 47 Incorrect ABS instruction operand. 48 Calculation overflow. 49 UR, UM, or UV unit not connected. 50 Unit device (R) not assigned. 51 Unit device (DM) not assigned. 52 Unit categories disagree 53 Operation recorder ID illegal 54 Stack overflow
Detailed information 2 (CM5158)	Fixed to 00H
Detailed information 3 (CM5159)	Upper bits of the step number at which the error occurred

Detailed Information, Piece n	Explanation
Detailed information 4 (CM5160)	Lower bits of the step number at which the error occurred
Detailed information 5 to 20 (CM5161 to CM5176)	The module name is saved according to the project language settings (any parts not occupied with the module name are "00H")

- No. 50, 53, 55, 56, 58, 59, 63 (unit number errors)

Detailed Information, Piece n	Explanation
Detailed information 1 (CM5107)	Unit number (the unit number will not be stored if it cannot be determined according to the occurrence conditions of the error)
Detailed information 2 to 20 (CM5108 to CM5126)	Fixed to 00H

- No. 54 (number of units error)

Detailed Information, Piece n	Explanation
Detailed information 1 (CM5107)	The number of units as specified in the unit setup information
Detailed information 2 (CM5108)	The number of units that are actually connected
Detailed information 3 to 20 (CM5109 to CM5126)	Fixed to 00H

- No. 127 (automatic loading failure error)

Detailed Information, Piece n	Explanation
Detailed information 1 (CM5157)	Details of error
Detailed information 2 to 20 (CM5158 to CM5176)	Fixed to 00H

- No. 129 (unit error))

Expansion Information, Piece n	Explanation
Expansion Information1 (CM5157)	Unit number
Expansion Information 2 (CM5158)	Error number of each unit
Expansion Information 3 to 20 (CM5159 to CM5176)	Fixed to 00H

- Error numbers other than those listed above

Expansion Information, Piece n	Explanation
Expansion Information 1 to 20	Fixed to 00H

Control Relays CR (For KV-7500/7300)

Control relays (CR0000 - CR7915) can be used to perform operations such as checking the operation status of the CPU unit and setting and monitoring the built-in functions.
Control relays reserved for the system cannot be used.

CR	Attribute R: Read only Blank: Read/write	Function	
CR0000		Logging or tracing function (ID0)	Logging and tracing execution (ON: enabled, OFF: disabled)
CR0001	R		Logging or tracing operation in progress
CR0002	R		Logging or tracing file writing completion
CR0003	R		Logging buffer less than 50% warning
CR0004	R		Logging buffer overflow
CR0005	R		Data to be written in RUN mode during logging present
CR0006 and CR0007	—		Reserved for the system
CR0008	R		Logging or tracing error
CR0009	R		Logging trace No memory space error
CR0010	R		Logging or tracing device or trigger setting error
CR0011	R		Tracing file save trigger monitor
CR0012	R		Tracing data capture completion
CR0013	R		Logging or tracing setting to be written present
CR0014 and CR0015	—		Reserved for the system
CR0100 to CR0915		Logging or tracing function (ID1 to ID9)	
CR1000 to CR1015		Real-time chart monitor (ID10)	
CR1100 to CR2001	—	Reserved for the system	
CR2002	R	Always turned ON	
CR2003	R	Always turned OFF	
CR2004	R	10 ms clock pulse (duty cycle: 50%)	
CR2005	R	100 ms clock pulse (duty cycle: 50%)	
CR2006	R	1 s clock pulse (duty cycle: 50%)	
CR2007	R	Turned OFF for just one scan when operation starts	
CR2008	R	Turned ON for just one scan when operation starts	
CR2009		Turned ON when the calculation result is negative or when an overflow occurs	
CR2010	R	Turned ON when the calculation result is zero	
CR2011	R	Turned ON when the calculation result is positive	
CR2012	R	Turned ON when a calculation execution error occurs	
CR2013 to CR2302	—	Reserved for the system	
CR2303		Fixed scan time operation (ON: enabled, OFF: disabled)	
CR2304	R	Turned ON for just one scan when the scan time setting (the fixed scan time operation and END processing time settings) is exceeded	
CR2305	—	Reserved for the system	
CR2306	R	Indicates backup battery errors (ON: error, OFF: normal)	
CR2307		Clears the minimum and maximum scan time values	
CR2308		Sets the END processing time (ON: enabled, OFF: disabled; if CR2303 is turned ON at the same time, CR2303 is given priority)	
CR2309 and CR2310	—	Reserved for the system	
CR2311	R	Indicates RTC low voltage error (ON: error has occurred, OFF: normal)	
CR2312	—	Reserved for the system	
CR2313		Forces switch of the VT sensor application screen (ON: switch, OFF: do not switch)	

CR	Attribute R: Read only Blank: Read/write	Function
CR2314	R	Project password authentication status (ON: authentication disabled, OFF: authentication enabled)
CR2315 to CR2813	—	Reserved for the system
CR2814		Zero suppression during execution of DASC and FASC instructions (all suffixes; ON: execute, OFF: do not execute)
CR2815		Omission of "+" during execution of DASC.S, DASC.L, and FASC instructions (ON: execute, OFF: do not execute)
CR2900 to CR3115	—	Reserved for the system
CR3200		CPU memory save save request
CR3201	R	CPU memory save being performed
CR3202	R	CPU memory save complete
CR3203	R	CPU memory save error
CR3204 to CR3206	—	Reserved for the system
CR3207		Instruction RAM mode operation The RAM mode activation of file register instructions, storage device instructions towards the CPU memory. (ON: enable, OFF: disable)
CR3208 to CR3209	—	Reserved for the system
CR3210	R	Arbitrary logging/trace operation
CR3211 to CR3213	—	Reserved for the system
CR3214	R	Storage instruction executing
CR3215	—	Reserved for the system
CR3300 to CR3415		Alarm relay
CR3500	R	Alarm operation relay (ON: one of CR3300 to CR3415 is turned ON)
CR3501		Record an entry in the log when an alarm relay switches from ON to OFF (ON: record, OFF: do not record)
CR3502		The alarm relay log is cleared when this switches from OFF to ON
CR3503 to CR3512	—	Reserved for the system
CR3513	R	Logging internal buffer overflow
CR3514		HKEY instruction - Disables multiple keys
CR3515	R	HKEY instruction - Scan completion
CR3600 to CR3615	R	HKEY information storage area
CR3700 to CR3815	—	Reserved for the system
CR3900		The serious error log is cleared when this switches from OFF to ON
CR3901		The minor error log is cleared when this switches from OFF to ON
CR3902	R	Indicates that the serious error log is full
CR3903	R	Indicates that the minor error log is full
CR3904	R	RUN-PRG selector switch status (ON: RUN, OFF: PROG)
CR3905	R	A serious error has occurred
CR3906	R	A minor error has occurred
CR3907		The power on log is cleared when this switches from OFF to ON
CR3908		The power off log is cleared when this switches from OFF to ON
CR3909		Clears the current error (Set this relay in the ladder program)
CR3910 to CR4000	—	Reserved for the system
CR4001 to CR4300	R	Expansion unit status (expansion units 1 - 48)
CR4301 to CR4315	—	Reserved for the system
CR4400	R	AW up key being pressed
CR4401	R	AW down key being pressed
CR4402	R	AW left key being pressed

CR	Attribute R: Read only Blank: Read/write	Function
CR4403	R	AW right key being pressed
CR4404	R	AW MENU key being pressed
CR4405	R	AW ENT key being pressed
CR4406 to CR4414	—	Reserved for the system
CR4415	R	AW user message being displayed
CR4500		AW line 1 blinking
CR4501		AW line 2 blinking
CR4502		AW line 3 blinking
CR4503		AW line 4 blinking
CR4504 to CR4600	—	Reserved for the system
CR4601 to CR4915		AW user message 1 to 63
CR5000 to CR5315	—	Reserved for the system
CR5400		Memory card mount request
CR5401		Memory card unmount request
CR5402	R	Memory card used
CR5403	R	Memory card recognized
CR5404	R	Memory card is available
CR5405	R	Memory card is write protected
CR5406 to CR5407	—	Reserved for the system
CR5408		CPU memory mount request
CR5409		CPU memory unmount request
CR5410	R	CPU memory used
CR5411	R	CPU memory recognized
CR5412 to CR7915	—	Reserved for the system

Control Memory Entries CM (For KV-7500/7300)

Control memory entries (CM0000 - CM5999) can be used to perform operations such as checking the operation status of the CPU unit and setting and monitoring the built-in functions.
Control memory entries reserved for the system cannot be used.

CM	Attribute R: Read only Blank: Read/ write	Function	Hold Yes: Hold
CM0000 and CM0699	—	Reserved for the system	
CM0700	R	Read from RTC (year)	
CM0701	R	Read from RTC (month)	
CM0702	R	Read from RTC (day)	
CM0703	R	Read from RTC (hour)	
CM0704	R	Read from RTC (minute)	
CM0705	R	Read from RTC (second)	
CM0706	R	Read from RTC (day of the week)	
CM0707 to CM709	—	Reserved for the system	
CM0710	R	File register current bank number	
CM0711	—	Reserved for the system	
CM0712	R	Computing error detailed information	
CM0713 to CM0899	—	Reserved for the system	
CM900	R	CPU function version	
CM901 to CM906	—	Reserved for the system	
CM907	R	Project Version	
CM908 to CM999	—	Reserved for the system	
CM1000 and CM1001	R	Scan time measuring value (1 μs) (32-bit)	
CM1002 and CM1003	R	END processing time measured value (1 μs unit) (32-bit)	
CM1004 and CM1005	R	Minimum fixed scanning time (1 μs unit) (32-bit)	
CM1006 and CM1007	R	Maximum fixed scanning time (1 μs unit) (32-bit)	
CM1008 and CM1009		Fixed scanning time run set value (1 μs unit) (32-bit)	
CM1010 and CM1011		END processing time set value (1 μs unit) (32-bit)	
CM1012 and CM1013	R	Scanning time when setting is exceeded (1 μs unit) (32-bit)	
CM1014 and CM1015	R	END processing time exceeds value (1 μs unit) (32-bit)	
CM1016 to CM1029	—	Reserved for the system	
CM1030 and CM1031	R	32-bit 1 ms Free Run counter	
CM1032 and CM1033	R	32-bit 10 μs Free Run counter	
CM1034 and CM1035	R	32-bit 1 μs Free Run counter	
CM1036 and CM1037	R	32-bit 0.1 μs Free Run counter	
CM1038 to CM1499	—	Reserved for the system	
CM1500 to CM1509	R	Log/trace (ID0 to ID9) File Save Counter	

CM	Attribute R: Read only Blank: Read/ write	Function	Hold Yes: Hold
CM1510 to CM1629	—	Reserved for the system	
CM1630 to CM1639	R	Log/trace Ring Buffer minimum free space (ID0 to ID9) (KBytes)	
CM1640 to CM1649	—	Reserved for the system	
CM1650	R	Number of sensor setting instruction executions	
CM1651 to CM1657	—	Reserved for the system	
CM1658	R	Number of consecutive project password authentication failures	Yes
CM1659	R	Total number of project password authentication failures	Yes
CM1660 to CM1669	R	Logging, trace ring buffer free capacity (ID0 - ID9) (in kbyte units)	
CM1670 to CM1679	—	Reserved for the system	
CM1680 to CM1689	R	Log/trace File Save Counter (ID0 to ID9) (Memory card use)	Yes
CM1690 to CM1699	—	Reserved for the system	
CM1700 and CM1701		Digital trimmer 0 upper limit (32-bit)	
CM1702 and CM1703		Digital trimmer 1 upper limit (32-bit)	
CM1704 and CM1705		Digital trimmer 2 upper limit (32-bit)	
CM1706 and CM1707		Digital trimmer 3 upper limit (32-bit)	
CM1708 and CM1709		Digital trimmer 4 upper limit (32-bit)	
CM1710 and CM1711		Digital trimmer 5 upper limit (32-bit)	
CM1712 and CM1713		Digital trimmer 6 upper limit (32-bit)	
CM1714 and CM1715		Digital trimmer 7 upper limit (32-bit)	
CM1716 to CM1719	—	Reserved for the system	
CM1720	R	AW user message display number	
CM1721 to CM1735	—	Reserved for the system	
CM1736		AW initial display language setting	
CM1737		AW initial comment number setting	
CM1738		Access window operation disabled setting	
CM1739	—	Reserved for the system	
CM1740 to CM1772	R	Ladder project name (up to 32 characters + end code)	
CM1773 to CM1779	—	Reserved for the system	
CM1780		Access window initial screen setting, enabled	
CM1781		Access window initial screen setting, device type	
CM1782		Access window initial screen setting, screen ID	
CM1783		Access window initial screen setting, screen ID	
CM1784		Access window initial screen setting, device number	
CM1785		Access window initial screen setting, device number	
CM1786		Access window initial screen setting, display format	
CM1787		Access window initial screen setting, device name representation	
CM1788		Access window initial screen setting, key lock status	

CM	Attribute R: Read only Blank: Read/ write	Function		Hold Yes: Hold
CM1789	–	Reserved for the system		
CM1790		Automatic loading folder number (request)		
CM1791	R	Automatic loading complete code		
CM1792	R	Automatic loading folder number (complete)		
CM1793	–	Reserved for the system		
CM1794		Run load folder number (request)		
CM1795	R	Run load complete code		
CM1796	R	Run load folder number (complete)		
CM1797 to CM1868	–	Reserved for the system		
CM1869	R	Number of connected expansion units		
CM1870 to CM2389	–	Reserved for the system		
CM2390	R	Error code for storage device instruction		
CM2391		Power OFF during storage access （Becomes value other than 0 during occurrence. Please change it to 0 from the program.）		
CM2392		Memory card retry occurrence times (still hold in case of power off)		
CM2393		Memory card check error times (still hold in case of power off)		
CM2394		Memory card timeout occurrence times (still hold in case of power off)		
CM2395		Reset delay time when error occurs on the memory card		
CM2396 to CM5099	–	Reserved for the system		
CM5100	R	Details of the latest serious error	Year and month	Yes
CM5101	R		Day	Yes
CM5102	R		Hour	Yes
CM5103	R		Minute	Yes
CM5104	R		Second	Yes
CM5105	R		Number	Yes
CM5106	R		Number of valid pieces of data	Yes
CM5107	R		Detailed information, piece 01	Yes
CM5108	R		Detailed information, piece 02	Yes
:	R			Yes
CM5126	R		Detailed information, piece 20	Yes
CM5127 to CM5149	–	Reserved for the system		
CM5150	R	Details of the latest minor error	Year and month	Yes
CM5151	R		Day	Yes
CM5152	R		Hour	Yes
CM5153	R		Minute	Yes
CM5154	R		Second	Yes
CM5155	R		Number	Yes
CM5156	R		Number of valid pieces of data	Yes
CM5157	R		Detailed information, piece 01*	Yes
CM5158	R		Detailed information, piece 02*	Yes
:	R			Yes
CM5176	R		Detailed information, piece 20*	Yes
CM5177 to CM5871	–	Reserved for the system		
CM5872 to CM5873	R	Used amount of CPU memory（32 bit, Byte unit）		Yes
CM5874 to CM5999	–	Reserved for the system		

* The detailed information which depends on the occurring error number is displayed in the following page.

- Error number 128 (calculation error)

Detailed Information, Piece n	Explanation
Detailed information 1 (CM5157)	Details of errors 0 (Not used) 1 Incorrect indirect specification. 2 The indirectly specified target is T/C. 3 Incorrect simple indirect specification. 4 Outside of operand range. 5 Floating-point real number overflow 6 Incorrect device specification. 7 The ROOT instruction operand is negative. 8 No bits turned ON with the ENCO instruction. 9 Floating-point real number format error. 10 Incorrect trigonometric function instruction operand. 11 Data cannot be converted. 12 Improper table specification. 13 Incorrect time comparison instruction operand. 14 Incorrect cam switch operand. 15 Incorrect frequency counter instruction operand. 16 Divide by zero. 17 Illegal logging ID. 18 Direct processing error. 19 Unit on which direct processing cannot be performed. 20 Jump destination subroutine does not exist. 21 Data being written during RUN mode, so instruction cannot be executed. 22 Error for expansion unit-dedicated instructions (unit number). 23 Reserved for the system. 24 PIDAT, outside of operation setting range. 25 PIDAT, outside of sampling period range. 26 PIDAT, outside of comparison constant range. 27 PIDAT, outside of integral constant range. 28 PIDAT, outside of differential constant range. 29 PIDAT comparison constant, two-degree freedom parameters outside of range. 30 PIDAT integral constant, two-degree freedom parameters outside of range. 31 PIDAT differential constant, two-degree freedom parameters outside of range. 32 PIDAT, outside of operation quantity upper limit or lower limit range. 33 PIDAT, outside of operation quantity change width limit range. 34 PIDAT, outside of measured value change width range. 35 PIDAT, AT state transition error. 36 PIDAT, outside of operation control flag range. 37 Operation period exceeded. 38 PIDAT, outside of AT adjustment parameter range. 39 PIDAT, outside of AT hysteresis range. 40 PIDAT, outside of AT timeout range. 41 PIDAT, outside of AT calculation control mode setting range. 42 PIDAT, AT timeout. 43 PIDAT, AT comparison constant exceeded. 44 PIDAT, AT integral constant exceeded. 45 PIDAT, AT differential constant exceeded. 46 Root of negative number cannot be calculated. 47 Incorrect ABS instruction operand. 48 Calculation overflow. 49 UR, UM, or UV unit not connected. 50 Unit device (R) not assigned. 51 Unit device (DM) not assigned.
Detailed information 2 (CM5158)	Fixed to 00H
Detailed information 3 (CM5159)	Upper bits of the step number at which the error occurred
Detailed information 4 (CM5160)	Lower bits of the step number at which the error occurred
Detailed information 5 to 20 (CM5161 to CM5176)	The module name is saved according to the project language settings (any parts not occupied with the module name are "00H")

- No. 50, 53, 55, 56, 58, 59, 63 (unit number errors)

Detailed Information, Piece n	Explanation
Detailed information 1 (CM5107)	Unit number (the unit number will not be stored if it cannot be determined according to the occurrence conditions of the error)
Detailed information 2 to 20 (CM5108 to CM5126)	Fixed to 00H

- No. 54 (number of units error)

Detailed Information, Piece n	Explanation
Detailed information 1 (CM5107)	The number of units as specified in the unit setup information
Detailed information 2 (CM5108)	The number of units that are actually connected
Detailed information 3 to 20 (CM5109 to CM5126)	Fixed to 00H

- No. 127 (automatic loading failure error)

Detailed Information, Piece n	Explanation
Detailed information 1 (CM5157)	Details of error
Detailed information 2 to 20 (CM5158 to CM5176)	Fixed to 00H

- No. 129 (unit error))

Expansion Information, Piece n	Explanation
Expansion Information1 (CM5157)	Unit number
Expansion Information 2 (CM5158)	Error number of each unit
Expansion Information 3 to 20 (CM5159 to CM5176)	Fixed to 00H

- Error numbers other than those listed above

Expansion Information, Piece n	Explanation
Expansion Information 1 to 20	Fixed to 00H

Control Relays CR (For KV-5500/5000/3000)

The control relay (CR0000 to CR7915) can check the operating state of the CPU unit, and set up or monitor each built-in function.

The system-reserved control relay cannot be used.

Device No.	Attributes R : Read only. Blank: R/W	Function	
CR0000		Logging /tracking function (ID0)	Logging/trace execution (ON: enable, OFF: disable)
CR0001	R		Logging/trace operation
CR0002	R		Logging/trace file write end
CR0003	R		Logging buffer 50% alarm
CR0004	R		Logging buffer overflow
CR0005	R		Logging write during RUN
CR0006 to CR0007	-		Reserved for the system
CR0008	R		Logging/trace error
CR0009	R		Logging/trace memory plugboard without capacity error
CR0010	R		Abnormal logging/trace device/trigger setting
CR0011	R		Trace file saving trigger monitor
CR0012	R		Trace data acquisition end
CR0013 to CR0015	-		Reserved for the system
CR0100 to CR0915		Logging/trace function (ID1 to ID9)	(identical with CR0000 to CR0015)
CR1000 to CR1015		Real-time graphic monitor (ID10)	(identical with CR0000 to CR0015)
CR1100 to CR2001	-	Reserved for the system	
CR2002	R	Always ON	
CR2003	R	Always OFF	
CR2004	R	10ms clock pulse (Duty ratio %: 50%)	
CR2005	R	100ms clock pulse (Duty ratio %: 50%)	
CR2006	R	1s clock pulse (Duty ratio %: 50%)	
CR2007	R	OFF scan period at operation start	
CR2008	R	ON scan period at operation start	
CR2009		ON when operation result is negative or overflowed	
CR2010	R	ON when the operation result is 0	
CR2011	R	ON when the operation result is positive	
CR2012	R	ON when operation execution error	
CR2013 to CR2015	-	Reserved for the system	
CR2100	R	CTH0 special internal clock (1μs)	
CR2101	R	CTH0 special internal clock (10μs)	
CR2102	R	CTH0 special internal clock (100μs)	
CR2103		Operation when high-speed counter comparator CTC0 is ON	Auto reset of CTH0 (ON: enable, OFF: disable)
CR2104			Comparator match output to R500/R504 is disabled (ON: enable, OFF: disable)
CR2105			Comparator match output to R500/R504 is set as OFF (ON: enable, OFF: disable)
CR2106			Comparator match output to R500/R504 is set as ON (ON: enable, OFF: disable)
CR2107			Invert ON/OFFstate of comparator match output to R500/R504 whenever in ON state (ON: enable, OFF: disable)

Device No.	Attributes R : Read only. Blank: R/W	Function	
CR2108		Operation when high-speed counter comparator CTC1 is ON	Comparator match output to R500/R504 is disabled (ON: enable, OFF: disable)
CR2109			Comparator match output to R500/R504 is set as OFF (ON: enable, OFF: disable)
CR2110			Comparator match output to R500/R504 is set as ON (ON: enable, OFF: disable)
CR2111			Invert ON/OFFstate of comparator match output to R500/R504 whenever in ON state. (ON: enable, OFF: disable)
CR2112			Auto reset of CTH0 (ON: enable, OFF: disable)
CR2113 to CR2115		Option of counting input methods of high-speed counter CTH0	

	1-phase		1Override	2Override	4Override	2-phase
	no direction	With direction				
CR2113	OFF	ON	OFF	ON	OFF	ON
CR2114	OFF	OFF	ON	ON	OFF	OFF
CR2115	OFF	OFF	OFF	OFF	ON	ON

CR2200	R	CTH1 special internal clock (1μs)	
CR2201	R	CTH1 special internal clock (10μs)	
CR2202	R	CTH1 special internal clock (100μs)	
CR2203		Operation when high-speed counter comparator CTC2 is ON	Auto reset of CTH1 (ON: enable, OFF: disable)
CR2204			Comparator match output to R501/R505 is disabled (ON: enable, OFF: disable)
CR2205			Comparator match output to R501/R505 is set as OFF (ON: enable, OFF: disable)
CR2206			Comparator match output to R501/R505 is set as ON (ON: enable, OFF: disable)
CR2207			Invert ON/OFFstate of comparator match output to R501/R505 whenever in ON state (ON: enable, OFF: disable)
CR2208		Operation when high-speed counter comparator CTC3 is ON	Comparator match output to R501/R505 is disabled (ON: enable, OFF: disable)
CR2209			Comparator match output to R501/R505 is set as OFF (ON: enable, OFF: disable)
CR2210			Comparator match output to R501/R505 is set as ON (ON: enable, OFF: disable)
CR2211			Invert ON/OFFstate of comparator match output to R501/R505 whenever in ON state (ON: enable, OFF: disable)
CR2212			Auto reset of CTH1 (ON: enable, OFF: disable)
CR2213 to CR2215		Option of counting input methods of high-speed counter CTH1	

	1-phase		1Override	2Override	4Override	2-phase
	no direction	With direction				
CR2213	OFF	ON	OFF	ON	OFF	ON
CR2214	OFF	OFF	ON	ON	OFF	OFF
CR2215	OFF	OFF	OFF	OFF	ON	ON

CR2300		External output disable (ON: enable, OFF: disable)
CR2301		External input refresh disable (ON: enable, OFF: disable)
CR2302		External output is disabled during interruption (ON: enable, OFF: disable)
CR2303		Fixed scanning time operation (ON: enable, OFF: disable)
CR2304	R	In case scanning time setting (fixed scanning time operation .END processing time setting) overflow, only 1-time scanning is ON
CR2305		Set input time constant of all inputs of CPU unit (ON: set→CM1620, OFF: setting disable [10ms])
CR2306	R	ON when back-up battery is abnormal

Device No.	Attributes R : Read only. Blank: R/W	Function
CR2307		Clearing of maximum/minimum value of scanning time
CR2308		END processing time setting (ON: enable, OFF: disable)
CR2309 to CR2312	-	Reserved for the system
CR2313		VT sensor application screen forced changeover (Change: ON, disable change: OFF)
CR2314	-	Reserved for the system
CR2315	R	In battery-less operation (during normal operation: OFF, during battery-less operation: ON)
CR2400, CR2401		CTH0 external signal preset Not used / Rise / Decline / Level CR2400: OFF / ON / OFF / ON CR2401: OFF / OFF / ON / ON
CR2402		Option of CTH0 counting method (ON: ring, OFF: linear)
CR2403		Detect CTH0 overflow/underflow (ON: detected, OFF: undetected)
CR2404, CR2405		CTH1 external signal preset Not used / Rise / Decline / Level CR2404: OFF / ON / OFF / ON CR2405: OFF / OFF / ON / ON
CR2406		Option of CTH1 counting method (ON: ring, OFF: linear)
CR2407		Detect CTH1 overflow/underflow (ON: detected, OFF: undetected)
CR2408		CTH0 preset disable (ON: disable, OFF: not disable)
CR2409		CTH1 preset disable (ON: disable, OFF: not disable)
CR2410		CTH0 internal permit relay (ON: enable counting, OFF: disable counting)
CR2411		CTH1 internal permit relay (ON: enable counting, OFF: disable counting)
CR2412	R	CTH0 change direction (ON: at the side of subtraction, OFF: at the side of adding)
CR2413	R	CTH1 change direction (ON: at the side of subtraction, OFF: at the side of adding)
CR2414	R	CTH0 special internal block (50ns)
CR2415	R	CTH1 special internal block (50ns)
CR2500		Frequency counter Start operation (ON: enable, OFF: disable)
CR2501, CR2502		Frequency counter Input source setting Single-phase without direction / Single-phase with direction / 2-phase 1-multiple / 2-phase CR2501: OFF / ON / OFF / ON CR2502: OFF / OFF / ON / ON
CR2503		Frequency counter Hz/rpm switching (ON: rpm, OFF: Hz)
CR2504		Frequency counter Use the function for updating the operation speed once (ON: available, OFF: not available)
CR2505		CTH0 comparator match output relay switching (ON: R504, OFF: R500)
CR2506		Specified frequency pulse output (CTH0) (ON: enable, OFF: disable)
CR2507		Specify frequency pulse output (CTH0) error
CR2508		Frequency counter (CTH1) Start operation (ON: enable, OFF: disable)
CR2509, CR2510		Frequency counter (CTH1) Input source setting Single-phase without direction / Single-phase with direction / 2-phase 1-multiple / 2-phase CR2509: OFF / ON / OFF / ON CR2510: OFF / OFF / ON / ON
CR2511		Frequency counter (CTH1) Hz/rpm switching (ON: rpm, OFF: Hz)
CR2512		Frequency counter (CTH1) Use the function for updating the operation speed once (ON: available, OFF: not available)
CR2513		CTH1 comparator match output relay switching (ON: R505, OFF: R501)
CR2514		Specified frequency pulse output (CTH1) (ON: enable, OFF: disable)
CR2515		Specify frequency pulse output (CTH1) error

Device No.	Attributes R : Read only. Blank: R/W	Function	
CR2600,CR2601		INT R000 interrupt polarity	
CR2602,CR2603		INT R001 interrupt polarity (identical with CR2600, CR2601)	
CR2604,CR2605		INT R002 interrupt polarity (identical with CR2600, CR2601)	
CR2606,CR2607		INT R003 interrupt polarity (identical with CR2600, CR2601)	
CR2608,CR2609		INT R004 interrupt polarity (identical with CR2600, CR2601)	
CR2610,CR2611		INT R005 interrupt polarity (identical with CR2600, CR2601)	
CR2612,CR2613		INT R006 interrupt polarity (identical with CR2600, CR2601)	
CR2614,CR2615		INT R007 interrupt polarity (identical with CR2600, CR2601)	
CR2700,CR2701		INT R008 interrupt polarity (identical with CR2600, CR2601)	
CR2702,CR2703		INT R009 interrupt polarity (identical with CR2600, CR2601)	
CR2704 to CR2715	-	Reserved for the system	
CR2800		Serial communication	Send BREAK signal (sending during ON)
CR2801	R		1 scanning ON when receiving file data
CR2802	R		Scan data load
CR2803	R		1 scanning ON in the event of occurrence of text data receiving error
CR2804			Start to send text data (ON during sending)
CR2805			Receive text data in bytes (ON: enable, OFF: disable)
CR2806			Send text data in bytes (ON: enable, OFF: disable)
CR2807 to CR2813	-	Reserved for the system	
CR2814		Zero suppression is performed when DASC instruction (all suffixes) and FASC instruction are executed (ON: enable, OFF: disable)	
CR2815		When DASC.S, DASC.L and FASC instruction execution are executed, "+" is omitted. (ON: enable, OFF: disable)	
CR2900		Access window	User message display1 (ON: display, OFF: not display)
CR2901			User message display2 (ON: display, OFF: not display)
CR2902 to CR2915	-	Reserved for the system	
CR3000		CPU positioning function axis X	Limit switch CW input
CR3001			Limit switch CCW input
CR3002			Origin sensor input
CR3003			Stop sensor input
CR3004	R		In the course of zero suppression
CR3005	R		Origin suppression end
CR3006			Positioning end relay
CR3007			Error/clearing error
CR3008			Warning/clearing warning
CR3009			Pulse output/emergency stop
CR3010			Driver operation start
CR3011			Current value change request (level detection)
CR3012			Operation speed change request (level detection)
CR3013			Consistency relay of comparator 2
CR3014	-		Reserved for the system
CR3015	-		

	1-phase w/o direction	Single-phase with direction	2phase 1 override	2-phase
CR2600	OFF	ON	OFF	ON
CR2601	OFF	OFF	ON	ON

Device No.	Attributes R : Read only. Blank: R/W	Function	
CR3100		CPU positioning function axis Y	Limit switch CW input
CR3101			Limit switch CCW input
CR3102			Origin sensor input
CR3103			Stop sensor input
CR3104	R		In the course of zero suppression
CR3105	R		Origin suppression end
CR3106			Positioning end relay
CR3107			Error/clearing error
CR3108			Warning/clearing warning
CR3109			Pulse output/emergency stop
CR3110			Driver operation start
CR3111			Current value change request (level detection)
CR3112			Operation speed change request (level detection)
CR3113			Consistency relay of comparator 2
CR3114	-		Reserved for the system
CR3115	-		
CR3200 to CR3209	-	Reserved for the system	
CR3210	R	Random Logging / trace memory plugboard in use	
CR3211	R	Memory card in use	
CR3212	R	Memory card identification end	
CR3213	R	With memory card	
CR3214	R	Memory card instruction in execution	
CR3215	R	Memory plugboard write protection (ON: write protection, OFF: without write protection)	
CR3300 to CR3415		Warning relay	
CR3500	R	Warning relay in operation (ON when one of CR3300 to CR3415 is ON)	
CR3501		When warning relay ON ➞ OFF, process the log (ON: reserve, OFF: without reserve)	
CR3502		When OFF turns to ON, clear Warning relay log	
CR3503 to CR3513	-	Reserved for the system	
CR3514		HKEY instruction	Multikey disable
CR3515	R		Scanning end
CR3600 to CR3615	R	Information storage area of HKEY instruction	

Device No.	Attributes R : Read only. Blank: R/W	Function	
CR3700		Operator panel KV-D30	F1 customized switch 1
CR3701			F2 customized switch 2
CR3702			F3 customized switch 3
CR3703			F4 customized switch 4
CR3704			LED1 customized indicator 1
CR3705			LED2 customized indicator 2
CR3706			LED3 customized indicator 3
CR3707			LED4 customized indicator 4
CR3708			Green backlit lamp (highlighted during ON)
CR3709			Red backlit lamp (highlighted during ON)
CR3710			System information display languag e(ON: Japanese, OFF: English)
CR3711			Buzzer ON
CR3712			Screen display setting (ON: positive display, OFF: reverse display)
CR3713			Warning interruption permit (ON: enable, OFF: disable)
CR3714			Page switching (rise switching to the page of CM0401, forced OFF)
CR3715			" ↑ "" ↓ " Page switching is disabled (ON: enable, OFF: disable)
CR3800			External output 0 is also in ON state in the case of ON state
CR3801			External output 1 is also in ON state in the case of ON state
CR3802 to CR3815	-	Reserved for the system	
CR3900		OFF→ON, severe error log is cleared	
CR3901		OFF→ON, slight error log is cleared	
CR3902	R	Severe error log represents maximum number	
CR3903	R	Slight error log represents maximum number	
CR3904	R	RUN/PRG switching state (ON: RUN side, OFF: PROG side)	
CR3905	R	In the event of occurrence of severe error	
CR3906	R	In the event of occurrence of slight error	
CR3907		OFF→ON, power ON log is cleared	
CR3908		OFF→ON, power OFF log is cleared	
CR3909		Current error is cleared (Only SET/RES instruction)	
CR3910 to CR3915	-	Reserved for the system	

Control Memory CM (For KV-5500/5000/3000)

Control memory (CM0000 - CM5999) may realize confirmation of CPU unit action state, settings or monitoring of built-in functions etc.
Control memory reversed in the system cannot be used.

Device No.	Attributes R : Read only. Blank: R/W	Function	
CM0000, CM0001		Operation panel KV-D30 0 Page	Set up 1st line display device
CM0002, CM0003			Set up 2nd line display device
CM0004, CM0005			Set up 3rd line display device
CM0006, CM0007	-		Set up 4th line display device
CM0008, CM0009			Reserved for system
CM0010 to CM0199		Page 1 to 19 (same as CM0000 to CM0009)	
CM0200, CM0201		Operation panel KV-D30 0 Pages	Set up 1st line display attribute
CM0202, CM0203			Set up 2nd line display attribute
CM0204, CM0205			Set up 3rd line display attribute
CM0206, CM0207			Set up 4th line display attribute
CM0208			Setup module 1
CM0209			Setup module 2
CM0210 to CM0399		Page 1 to 19 (same as CM0200 to CM0209)	
CM0400		Operation panel KV-D30	For conversion enable/inhibited setting
CM0401			For settings of page conversion
CM0402			Current display page
CM0403 to CM0409			Reserved for system
CM0410 to CM0415			Direct access function
CM0416 to CM0499	-		Reserved for system
CM0500 to CM0699	-	Reserved for system	
CM0700	R	Read from calendar timer	(year)
CM0701	R		(month)
CM0702	R		(day)
CM0703	R		(hour)
CM0704	R		(Minute)
CM0705	R		(s)
CM0706	R		(week)(0: (Sun), 1 (Mon), 2: (Tues) 3: (Wed) 4: (Thur) 5: (Fri), 6 (Sat))
CM0707	-	Reserved for system	
CM0708, CM0709	R	Free Run Counter (32-bit, 1ms)	
CM0710	R	Current memory No.of FM	
CM0711	R	If V2.0 CPU functional version is used, save "2", for other situations, save "0"	
CM0712	R	Execution time of constant cycle module (10μs unit)	
CM0713 to CM0715	-	Reserved for system	
CM0716		Maximum exeuction time (10μs unit) for Fixed Period Module	
CM0717 to CM0719	-	Reserved for system	
CM0720	R	Scan time measuring value (10μs unit)	
CM0721		Fixed scanning time run set value (10μs unit)	
CM0722	R	Exceeds scanning time set up (-fixed scanning time running/END processing time settings) scanning time (10μs unit)	
CM0723		END processing time measured value (10μs unit)	
CM0724 to CM0725	-	Reserved for system	
CM0726	R	Minimum fixed scanning time (10μs unit)	
CM0727	R	Maximum fixed scanning time (10μs unit)	
CM0728		END processing time set value (10μs unit)	
CM0729		Exceeds scanning time set up (-fixed scanning time running/END processing time settings) scanning time (10μs unit)	

Device No.	Attributes R : Read only. Blank: R/W	Function	
CM0730 to CM1479	-	Reserved for system	
CM1480, CM1481	R	Frequency counter function (CTH0)	Frequency measuring result(Hz or rpm)
CM1482			1Number of placed pulse
CM1483			Scan times
CM1484, CM1485		Specify frequency pulse output function.	Setup frequency (Hz)
CM1486			Duty ratio (0 to 100%)
CM1487 to CM1489	-	Reserved for system	
CM1490, CM1491	R	Frequency counter function (CTH1)	Frequency measuring result (Hz or rpm)
CM1492			Number of pulses per rotation
CM1493			Scan times
CM1494, CM1495		Specify frequency pulse output function.	Setup frequency (Hz)
CM1496			Duty ratio (0 to 100%)
CM1497 to CM1593	-	Reserved for system	
CM1594		CTH0 change direction detection constant	
CM1595		CTH01 change direction detection constant	
CM1596, CM1597		Lower limit of CTH0 ring counter	
CM1598, CM1599		Lower limit of CTH1 ring counter	
CM1600, CM1601		CTH0 ring counter upper limit	
CM1602, CM1603		CTH1 ring counter upper limit	
CM1604, CM1605		CTH0 preset value	
CM1606, CM1607		CTH1 preset value	
CM1608		CTH0 start, reset, polarity set up	
CM1609		CTH0 start, reset, polarity set up	
CM1610, CM1611	R	INT R00000 Input capture (high bit, low bit) when occurred	
CM1612, CM1613	R	INT R00001 Input capture (high bit, low bit) when occurred	
CM1614, CM1615	R	INT R00002 Input capture (high bit, low bit) when occurred	
CM1616, CM1617	R	INT R00003 Input capture (high bit, low bit) when occurred	
CM1618, CM1619	-	Reserved for system	
CM1620		Setup constant when inputting	
CM1621 to CM1649	-	Reserved for system	
CM1650	R	Executed number of sensor setting command	
CM1651 to CM1659	R	Reserved for system	
CM1660 to CM1669	R	Log/trace Ring Buffer free space (ID0 to ID10)	
CM1670 to CM1679	-	Reserved for system	
CM1680 to CM1689	R	Log/trace File Save Counter (ID0 to ID10)	
CM1690 to CM1699	-	Reserved for system	
CM1700, CM1701		Digital trimmer 0 upper limit	
CM1702, CM1703		Digital trimmer 1 upper limit	
CM1704, CM1705		Digital trimmer 2 upper limit	
CM1706, CM1707		Digital trimmer 3 upper limit	
CM1708, CM1709		Digital trimmer 4 upper limit	
CM1710, CM1711		Digital trimmer 5 upper limit	
CM1712, CM1713		Digital trimmer 6 upper limit	
CM1714, CM1715		Digital trimmer 7 upper limit	
CM1716 to CM1719		Reserved for system	
CM1720		Access Window	user message 1
CM1721 to CM1737			user message 2 (up to 24 characters + end character (00 H))
CM1738			Setting Inhibited Operation
CM1739	-		Reserved for system

Device No.	Attributes R : Read only. Blank: R/W	Function	
CM1740 to CM1756		LD project name (up to 32 characters +end character (00 H))	
CM1757 to CM1763		Access window display project name (up to 12 characters + end character (00 H))	
CM1764 to CM1774		Bluetooth device identification name (up to 20 characters + terminal code (00H)) *1	
CM1775 to CM1779		Reserved for system	
CM 1780		AW initial menu setting is enabled	
CM1781		AW initial menu setting device type	
CM1782,CM1783		AW initial menu setting menu ID	
CM1784,CM1785		AW initial menu setting device No.	
CM1786		AW initial menu setting display format	
CM1787		AW initial menu setting device name representation	
CM1788		AW initial menu setting key lock state	
CM1789		AW initial menu setting display language	
CM1790		Automatic loading folder No. When powered on (request)	
CM1791		Automatic loading folder No. When powered on (completed)	
CM1792		Automatic loading folder No. For PROG/RUN (request)	
CM1793		Automatic loading folder No. For PROG/RUN (completed)	
CM1794 to CM1799		Reserved for system	
CM1800		Serious error log (latest)	Year month (decimal number, lower two 2 digits represent month, digits above hundred digit represent year. Example: 707 -> June, 2007
CM1801			day
CM1802			hour(0 to 23)
CM1803			minute
CM1804			Error No.
CM1805 to CM1899		Error log (2 to 20 records, same as CM1800 to CM1804)	
CM1900		Power ON log (latest)	Year month (decimal number, lower two 2 digits represent month, digits above hundred digit represent year. Example: 707 -> June, 2007
CM1901			day
CM1902			hour(0 to 23)
CM1903			minute
CM1904			second
CM1905 to CM1999		Power ON log (2 to 20 records, same as CM1800 to CM1804)	

*1 Only CPU unit that supports Bluetooth communication is valid.

Device No.	Attributes R : Read only. Blank: R/W	Function	
CM2000		CPU Positioning function X axis	I/O setting
CM2001			Start sensor
CM2002, CM2003			Comparator 0
CM2004, CM2005			Comparator 1
CM2006			Origin return: starting speed
CM2007			Zero return: acceleration/deceleration time
CM2008, CM2009			Origin return: running speed
CM2010			JOG operation: starting speed
CM2011			JOG operation: acceleration/deceleration time
CM2012, CM2013			JOG operation: running speed
CM2014, CM2015			Current value change setting value
CM2016, CM2017			Operation speed change setting value
CM2018			Set up the origin return detail
CM2019			Operation in progress point No.(M code)
CM2020, CM2021	R		Current position
CM2022, CM2023	R		Current speed
CM2024			Error code
CM2025			Specify point No.
CM2026, CM2027			Resting position coordinate setpoint
CM2028, CM2029			Comparator 2
CM2030		CPU Positioning function Y axis	I/O setting
CM2031			Start sensor
CM2032, CM2033			Comparator 0
CM2034, CM2035			Comparator 1
CM2036			Origin return: starting speed
CM2037			Zero return: acceleration/deceleration time
CM2038, CM2039			Origin return: running speed
CM2040			JOG operation: starting speed
CM2041			JOG operation: acceleration/deceleration time
CM2042, CM2043			JOG operation: running speed
CM2044, CM2045			Current value change setting value
CM2046, CM2047			Operation speed change setting value
CM2048			Set up the origin return detail
CM2049			Operation in progress point No.(M code)
CM2050, CM2051	R		Current position
CM2052, CM2053	R		Current speed
CM2054			Error code
CM2055			Specify point No.
CM2056, CM2057			Resting position coordinate setpoint
CM2058, CM2059			Comparator 2
CM2060, CM2061		CPU Positioning function Point parameter 0	Target value/amount of movement
CM2062			Startup speed
CM2063			Acceleration/deceleration time
CM2064, CM2065			Operation speed
CM2066			Operation mode
CM2067			Specified pulse number for stop sensor
CM2068, CM2069			Reserved for system
CM2070 to CM2379		Point parameter 1 to 31 (The same with CM2060 to CM2069)	
CM2380 to CM2389	-	Reserved for system	
CM2390	R	Error code used for memory card instruction	

Device No.	Attributes R : Read only. Blank: R/W	Function	
CM2391		Power OFF when accessing memory plug-in board (when occurred: the value other than 0)	
CM2392		Memory card retry occurrence times (still hold in case of power off)	
CM2393		Memory card check error times (still hold in case of power off)	
CM2394		Memory card timeout occurrence times (still hold in case of power off)	
CM2395		When error occurs on the memory card reset delay time (in ms, upper limit is 10000[10 s], do not reset if set to 65535)	
CM2392 to CM2399		Reserved for system	
CM2400 to CM2416		Interrupt times (INT0 to INT3, CTC2, CTC3, CTC0, CTC1, positioning X axis, positioning Y axis, INT4 to INT9, fixed cycle module)	
CM2417 to CM2999	-	Reserved for system	
CM3000		Number of received texts	
CM3001 to CM3999		Text received data (1998-byte)	
CM4000		Number of sending texts	
CM4001 to CM4999		Text send data (1998-byte)	
CM5000		Power OFF log (latest)	Year month (decimal number, lower two 2 digits represent month, digits above hundred digit represent year. Example: 707 -> June, 2007
CM5001			day
CM5002			hour(0 to 23)
CM5003			minute
CM5004			second
CM5005 to CM5099		Power OFF log (2 to 20 units the same with CM5000 to CM5004)	
CM5100		Latest serious error detail information 1 unit*	Year month (decimal number, lower two 2 digits represent month, digits above hundred digit represent year. Example: 707 -> June, 2007
CM5101			day
CM5102			HH
CM5103			minute
CM5104			second
CM5105			Error No.
CM5106			Effective data number of expansion information
CM5107			Detailed information 1*
CM5108			Detailed information 2*
: :			: :
CM5126			Detailed information 20*
CM5150		Latest general warning detail information 1 unit*	Year month (decimal number, lower two 2 digits represent month, digits above hundred digit represent year. Example: 707 -> June, 2007
CM5151			day
CM5152			hour
CM5153			minute
CM5154			second
CM5155			Error No.
CM5156			Effective data number of expansion information
CM5157			Detailed information 1*
CM5158			Detailed information 2*
: :			: :
CM5176			Detailed information 20*
CM5177 to CM5999	-	Reserved for system	

* Detailed information error is coded as follows:

- When No.128 (calculation error)

Detailed information n	Description
Detailed information 1	Details of error 0 (not used) 1 Improper indirect specification 2 The indirectly specified target is T/C 3 Improper simple indirect specifying 4 Beyond operand range 5 Floating real number overflow 6 Improper device specifying 7 ROOT operand is negative 8 No ENCO instruction ON bit 9 Floating-point real number error 10 Trigonometric function operand error 11 data cannot be converted 12 Improper table specifying 13 Time comparison instruction operand error 14 Cam switch operand error 15 Frequency counter instruction operand error 16 0 division operation 17 Record ID illegal 18 Direct Processing Error 19 The units cannot direct processed 20 Subroutine does not exist for transfer target 21 Writing during RUN, cannot execute command 22 Expansion unit special command object error (unit No.) 23 Reserved for system 24 PIDAT operation setting out-of-range 25 PIDAT sampling period out-of-range 26 PIDAT constant of proportionality out-of-range 27 PIDAT integral constant out-of-range 28 PIDAT differential constant out-of-range 29 PIDAT constant of proportionality two degree freedom parameters are out of scope 30 PIDAT integral constant two degree freedom parameters are out of scope 31 PIDAT differential constant two degree freedom parameters are out of scope 32 PIDAT operation quantity upper limit value/lower limit value out-of-range 33 PIDAT operation quantity change amplitude critical value is out of scope 34 PIDAT measured value change amplitude is out of scope 35 PIDAT AT state transition exception 36 PIDAT operational control mark is out of scope 37 PIDAT operation period is out of scope 38 PIDAT AT adjusting parameters are out of scope 39 PIDAT AT lagging is out of scope 40 PIDAT AT timeout value is out of scope 41 PIDAT AT calculation control mode setting is out of range 42 PIDAT AT timeout 43 PIDAT AT constant of proportionality is out of scope 44 PIDAT AT integral constant is out of scope 45 PIDAT AT differential constant is out of scope 46 Root of negative number cannot be calculated 47 Independent variable of ABS command is incorrect 48 Operation overflow

Detailed information n	Description
Detailed information 2	**00H fixed**
Detailed information 3	Step number high bit error
Detailed information 4	Step number low bit error
Detailed information 5 to 20	**Use SJIS to store module name. (the part other than module name is 00H)**

· No.50,53,55,56,58,59 (unit ID related errors)

Detailed information n	Description
Detailed information 1	Unit No. (according to occurrence state of error, when unit No. Cannot be designated, it will not be saved.)
Detailed information 2 to 20	00H fixed

· When No.54 (error related with number of units)

Detailed information n	Description
Detailed information 1	Number of units on unit setting information
Detailed information 2	Actually connected number of units
Detailed information 3 to 20	00H fixed

· When No.127 (automatic read error)

Detailed information n	Description
Detailed information 1	Details of error For details, refer to 📖 "Errors during load/save execution".
Detailed information 2 to 20	00H fixed

· When No.129 (unit error)

Detailed information n	Description
Detailed information 1	Unit No.
Detailed information 2	Error No. of each unit
Detailed information 3 to 20	00H fixed

· In case of error numbers other than above

Detailed information n	Description
Detailed information 1 to 20	00H fixed

Control Relays CR (For KV Nano series)

Control relays (CR0000 to CR8915) can be used to perform operations such as checking the operation status of the base unit and setting and monitoring the built-in functions.
Control relays reserved for the system cannot be used.

CR	Attribute R: Read only Blank: Read/write	Function	
CR0000		Logging or tracing function (ID0)	Logging and tracing execution (ON: enabled, OFF: disabled)
CR0001	R		Logging or tracing operation in progress
CR0002	R		Logging or tracing file writing completion
CR0003	R		Logging buffer less than 50% warning
CR0004	R		Logging buffer overflow
CR0005	R		Data to be written in RUN mode during logging present
CR0006, CR0007	-		Reserved for the system
CR0008	R		Logging or tracing error
CR0009	R		Logging or tracing insufficient memory card capacity error
CR0010	R		Logging or tracing device or trigger setting error
CR0011	R		Tracing file save trigger monitor
CR0012	R		Tracing data capture completion
CR0013	R		Logging or tracing setting to be written present
CR0014, CR0015	-		Reserved for the system
CR0100 to CR0915		Logging or tracing function (ID1 to ID9)	(The same as CR0000 to CR0015)
CR1000 to CR1015		Real-time chart monitor (ID10)	(The same as CR0000 to CR0015)
CR1100 to CR2001	-	Reserved for the system	
CR2002	R	Always turned ON	
CR2003	R	Always turned OFF	
CR2004	R	10 ms clock pulse (duty cycle: 50%)	
CR2005	R	100 ms clock pulse (duty cycle: 50%)	
CR2006	R	1 s clock pulse (duty cycle: 50%)	
CR2007	R	Turned OFF for just one scan when operation starts	
CR2008	R	Turned ON for just one scan when operation starts	
CR2009		Turned ON when the calculation result is negative or when an overflow occurs	
CR2010	R	Turned ON when the calculation result is zero	
CR2011	R	Turned ON when the calculation result is positive	
CR2012	R	Turned ON when a calculation execution error occurs	
CR2013 to CR2100	-	Reserved for the system	
CR2101	R	Indicates the presence of extension unit 1 (ON: connected, OFF: not connected)	
CR2102	R	Indicates the presence of extension unit 2 (ON: connected, OFF: not connected)	
CR2103 to CR2200	-	Reserved for the system	
CR2201	R	Indicates the presence of expansion unit 1 (ON: connected, OFF: not connected)	
CR2202	R	Indicates the presence of expansion unit 2 (ON: connected, OFF: not connected)	
CR2203	R	Indicates the presence of expansion unit 3 (ON: connected, OFF: not connected)	
CR2204	R	Indicates the presence of expansion unit 4 (ON: connected, OFF: not connected)	
CR2205	R	Indicates the presence of expansion unit 5 (ON: connected, OFF: not connected)	
CR2206	R	Indicates the presence of expansion unit 6 (ON: connected, OFF: not connected)	
CR2207	R	Indicates the presence of expansion unit 7 (ON: connected, OFF: not connected)	
CR2208	R	Indicates the presence of expansion unit 8 (ON: connected, OFF: not connected)	

CR	Attribute R: Read only Blank: Read/write	Function
CR2209 to CR2215	-	Reserved for the system
CR2300		External output disabled (ON: execute, OFF: do not execute)
CR2301		External input refresh disabled (ON: execute, OFF: do not execute)
CR2302	-	Reserved for the system
CR2303		Fixed scan time operation (ON: enabled, OFF: disabled)
CR2304	R	Turned ON for just one scan when the scan time setting (the fixed scan time operation and END processing time settings) is exceeded
CR2305		Sets the input time constant of all inputs of the base unit (ON: the value is set according to CM1620, OFF: the value is not set)
CR2306	-	Reserved for the system
CR2307		Clears the minimum and maximum scan time values
CR2308		Sets the END processing time (ON: enabled, OFF: disabled; if CR2303 is turned ON at the same time, CR2303 is given priority)
CR2309		Reserved for the system
CR2310	R	Indicates the presence of a calendar timer (ON: present, OFF: not present)
CR2311	R	This is turned ON when a clock data lost error occurs (it is turned OFF when the clock data is written)
CR2312	-	Reserved for the system
CR2313		Forces switch of the VT sensor application screen (ON: switch, OFF: do not switch)
CR2314		Project password authentication status (ON: authentication disabled, OFF: authentication enabled)
CR2315 to CR2515	-	Reserved for the system
CR2600, CR2601		INT0 interrupt polarity Rising Edge / Falling Edge / Both Rising and Falling Edges CR2600: OFF / ON / OFF / ON CR2601: OFF / OFF / ON / ON
CR2602, CR2603		INT1 interrupt polarity Rising Edge / Falling Edge / Both Rising and Falling Edges CR2602: OFF / ON / OFF / ON CR2603: OFF / OFF / ON / ON
CR2604, CR2605		INT2 interrupt polarity Rising Edge / Falling Edge / Both Rising and Falling Edges CR2604: OFF / ON / OFF / ON CR2605: OFF / OFF / ON / ON
CR2606, CR2607		INT3 interrupt polarity Rising Edge / Falling Edge / Both Rising and Falling Edges CR2606: OFF / ON / OFF / ON CR2607: OFF / OFF / ON / ON
CR2608 to CR2813		Reserved for the system
CR2814		Zero suppression during execution of DASC and FASC instructions (all suffixes; ON: execute, OFF: do not execute)
CR2815		Omission of "+" during execution of DASC.S, DASC.L, and FASC instructions (ON: execute, OFF: do not execute)
CR2900		Access window: User message display 1 (ON: displayed, OFF: hidden)
CR2901		Access window: User message display 2 (ON: displayed, OFF: hidden)
CR2902 to CR2914	-	Reserved for the system
CR2915	R	Indicates the presence of an access window cassette (ON: connected, OFF: not connected)
CR3000		Project load execution request
CR3001	-	Reserved for the system
CR3002		Project save execution request
CR3003	-	Reserved for the system
CR3004		Project load execution completion

CR	Attribute R: Read only Blank: Read/write	Function	
CR3005		Project load execution failure	
CR3006		Project save execution completion	
CR3007		Project save execution failure	
CR3008 to CR3209	-	Reserved for the system	
CR3210	R	Optional logging or tracing memory card is in use	
CR3211	R	Memory card is in use	
CR3212	R	Memory card recognition complete	
CR3213	R	Memory card present	
CR3214	R	Memory card executing instructions	
CR3215	R	Memory card write protection	
CR3300 to CR3415	-	Alarm relays	
CR3500	R	Alarm operation relay (ON: one of CR3300 to CR3415 is turned ON)	
CR3501		Record an entry in the log when an alarm relay switches from ON to OFF (ON: record, OFF: do not record)	
CR3502		The alarm relay log is cleared when this switches from OFF to ON	
CR3503 to CR3513		Reserved for the system	
CR3514		HKEY instruction	Disables multiple keys
CR3515	R		Scan completion
CR3600 to CR3615	R	HKEY information storage area	
CR3700		Operator panel KV-D30	F1 customize switch 1
CR3701			F2 customize switch 2
CR3702			F3 customize switch 3
CR3703			F4 customize switch 4
CR3704			LED1 customize indicator 1
CR3705			LED2 customize indicator 2
CR3706			LED3 customize indicator 3
CR3707			LED4 customize indicator 4
CR3708			Green back light (lights when this is turned ON)
CR3709			Red back light (lights when this is turned ON)
CR3710			System message display language (ON: Japanese, OFF: English)
CR3711			Buzzer ON
CR3712			Screen display setting (ON: positive display, OFF: negative display)
CR3713			Alarm interrupt enable (ON: enabled, OFF: disabled)
CR3714			Page switching (switches to the page indicated by the value of CM0401 on a rising edge; forced OFF)
CR3715			Disables page switching using the "▲" and "▼" keys (ON: disabled, OFF: enabled)
CR3800			Sets external output 0 to ON when this is turned ON
CR3801			Sets external output 1 to ON when this is turned ON
CR3802 to CR3815	-	Reserved for the system	
CR3900		The serious error log is cleared when this switches from OFF to ON	
CR3901		The minor error log is cleared when this switches from OFF to ON	
CR3902	R	Indicates that the serious error log is full	
CR3903	R	Indicates that the minor error log is full	
CR3904	R	RUN-PROG selector switch status (ON: RUN, OFF: PROG)	
CR3905	R	A serious error has occurred	
CR3906	R	A minor error has occurred	
CR3907		The power on log is cleared when this switches from OFF to ON	
CR3908		The power off log is cleared when this switches from OFF to ON	
CR3909		Clears the current error	

CR	Attribute R: Read only Blank: Read/write	Function
CR3910 to CR4115	-	Reserved for the system
CR4200	R	CTH0 dedicated internal clock (50 ns)
CR4201	R	CTH0 dedicated internal clock (1 μs)
CR4202	R	CTH0 dedicated internal clock (10 μs)
CR4203	R	CTH0 dedicated internal clock (100 μs)
CR4204	-	Reserved for the system
CR4205	R	Indicates the presence of CTH0 (ON: present, OFF: not present)
CR4206		Detects overflows and underflows of CTH0 (ON: detected, OFF: not detected)
CR4207	R	Indicates the change direction of CTH0 (ON: down counting, OFF: up counting)
CR4208, CR4209		CTH0 external signal preset [table 1]
CR4210		CTH0 count mode selection (ON: ring, OFF: linear)
CR4211		CTH0 preset disabled (ON: execute, OFF: do not execute)
CR4212		CTH0 internal enable relay (ON: counting enabled, OFF: counting disabled)
CR4213 to CR4302	-	Reserved for the system
CR4303		Action when high-speed counter comparator CTC0 is turned ON: Automatic reset of CTH0 (ON: execute, OFF: do not execute)
CR4304		Comparator matching output disabled when comparator CTC0 is turned ON (ON: execute, OFF: do not execute)
CR4305		Comparator matching output turned OFF when comparator CTC0 is turned ON (ON: execute, OFF: do not execute)
CR4306		Comparator matching output turned ON when comparator CTC0 is turned ON (ON: execute, OFF: do not execute)
CR4307		Comparator matching output inversion (ON/OFF) each time that comparator CTC0 is turned ON (ON: execute, OFF: do not execute)
CR4308		Action when high-speed counter comparator CTC1 is turned ON: Comparator matching output disabled when comparator CTC1 is turned ON (ON: execute, OFF: do not execute)
CR4309		Comparator matching output turned OFF when comparator CTC1 is turned ON (ON: execute, OFF: do not execute)
CR4310		Comparator matching output turned ON when comparator CTC1 is turned ON (ON: execute, OFF: do not execute)
CR4311		Comparator matching output inversion (ON/OFF) each time that comparator CTC1 is turned ON (ON: execute, OFF: do not execute)
CR4312		Automatic reset of CTH0 (ON: execute, OFF: do not execute)
CR4313 to CR4315		High-speed counter CTH0 count input mode selection [table 2]
CR4400	R	CTH1 dedicated internal clock (50 ns)
CR4401	R	CTH1 dedicated internal clock (1 μs)
CR4402	R	CTH1 dedicated internal clock (10 μs)
CR4403	R	CTH1 dedicated internal clock (100 μs)
CR4404	-	Reserved for the system
CR4405	R	Indicates the presence of CTH1 (ON: present, OFF: not present)
CR4406		Detects overflows and underflows of CTH1 (ON: detected, OFF: not detected)
CR4407	R	Indicates the change direction of CTH1 (ON: down counting, OFF: up counting)

[table 1]

	Not Used	Rising Edge	Falling Edge	Level
CR4208	OFF	ON	OFF	ON
CR4209	OFF	OFF	ON	ON

[table 2]

	Single Phase		1x	2x	4x	2-Pulse
	W/O Direction	W/ Direction				
CR4313	OFF	ON	OFF	ON	OFF	ON
CR4314	OFF	OFF	ON	ON	OFF	OFF
CR4315	OFF	OFF	OFF	OFF	ON	ON

<table>
<tr><th>CR</th><th>Attribute
R: Read only
Blank: Read/write</th><th colspan="2">Function</th></tr>
<tr><td>CR4408,
CR4409</td><td></td><td colspan="2">CTH1 external signal preset
<table>
<tr><th></th><th>Not Used</th><th>Rising Edge</th><th>Falling Edge</th><th>Level</th></tr>
<tr><td>CR4408</td><td>OFF</td><td>ON</td><td>OFF</td><td>ON</td></tr>
<tr><td>CR4409</td><td>OFF</td><td>OFF</td><td>ON</td><td>ON</td></tr>
</table></td></tr>
<tr><td>CR4410</td><td></td><td colspan="2">CTH1 count mode selection (ON: ring, OFF: linear)</td></tr>
<tr><td>CR4411</td><td></td><td colspan="2">CTH1 preset disabled (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4412</td><td></td><td colspan="2">CTH1 internal enable relay (ON: counting enabled, OFF: counting disabled)</td></tr>
<tr><td>CR4413 to CR4502</td><td>-</td><td colspan="2">Reserved for the system</td></tr>
<tr><td>CR4503</td><td></td><td rowspan="5">Action when high-speed counter comparator CTC2 is turned ON</td><td>Automatic reset of CTH1 (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4504</td><td></td><td>Comparator matching output disabled when comparator CTC2 is turned ON (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4505</td><td></td><td>Comparator matching output turned OFF when comparator CTC2 is turned ON (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4506</td><td></td><td>Comparator matching output turned ON when comparator CTC2 is turned ON (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4507</td><td></td><td>Comparator matching output inversion (ON/OFF) each time that comparator CTC2 is turned ON (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4508</td><td></td><td rowspan="5">Action when high-speed counter comparator CTC3 is turned ON</td><td>Comparator matching output disabled when comparator CTC3 is turned ON (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4509</td><td></td><td>Comparator matching output turned OFF when comparator CTC3 is turned ON (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4510</td><td></td><td>Comparator matching output turned ON when comparator CTC3 is turned ON (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4511</td><td></td><td>Comparator matching output inversion (ON/OFF) each time that comparator CTC3 is turned ON (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4512</td><td></td><td>Automatic reset of CTH1 (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4513 to CR4515</td><td></td><td colspan="2">CTH1 count input mode selection
<table>
<tr><th></th><th colspan="2">Single Phase</th><th rowspan="2">1x</th><th rowspan="2">2x</th><th rowspan="2">4x</th><th rowspan="2">2-Pulse</th></tr>
<tr><th></th><th>W/O Direction</th><th>W/ Direction</th></tr>
<tr><td>CR4513</td><td>OFF</td><td>ON</td><td>OFF</td><td>ON</td><td>OFF</td><td>ON</td></tr>
<tr><td>CR4514</td><td>OFF</td><td>OFF</td><td>ON</td><td>ON</td><td>OFF</td><td>OFF</td></tr>
<tr><td>CR4515</td><td>OFF</td><td>OFF</td><td>OFF</td><td>OFF</td><td>ON</td><td>ON</td></tr>
</table></td></tr>
<tr><td>CR4600</td><td>R</td><td colspan="2">CTH2 dedicated internal clock (50 ns)</td></tr>
<tr><td>CR4601</td><td>R</td><td colspan="2">CTH2 dedicated internal clock (1 μs)</td></tr>
<tr><td>CR4602</td><td>R</td><td colspan="2">CTH2 dedicated internal clock (10 μs)</td></tr>
<tr><td>CR4603</td><td>R</td><td colspan="2">CTH2 dedicated internal clock (100 μs)</td></tr>
<tr><td>CR4604</td><td>-</td><td colspan="2">Reserved for the system</td></tr>
<tr><td>CR4605</td><td>R</td><td colspan="2">Indicates the presence of CTH2 (ON: present, OFF: not present)</td></tr>
<tr><td>CR4606</td><td></td><td colspan="2">Detects overflows and underflows of CTH2 (ON: detected, OFF: not detected)</td></tr>
<tr><td>CR4607</td><td>R</td><td colspan="2">Indicates the change direction of CTH2 (ON: down counting, OFF: up counting)</td></tr>
<tr><td>CR4608,
CR4609</td><td></td><td colspan="2">CTH2 external signal preset
<table>
<tr><th></th><th>Not Used</th><th>Rising Edge</th><th>Falling Edge</th><th>Level</th></tr>
<tr><td>CR4608</td><td>OFF</td><td>ON</td><td>OFF</td><td>ON</td></tr>
<tr><td>CR4609</td><td>OFF</td><td>OFF</td><td>ON</td><td>ON</td></tr>
</table></td></tr>
<tr><td>CR4610</td><td></td><td colspan="2">CTH2 count mode selection (ON: ring, OFF: linear)</td></tr>
<tr><td>CR4611</td><td></td><td colspan="2">CTH2 preset disabled (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4612</td><td></td><td colspan="2">CTH2 internal enable relay (ON: counting enabled, OFF: counting disabled)</td></tr>
<tr><td>CR4613 to CR4702</td><td>-</td><td colspan="2">Reserved for the system</td></tr>
</table>

CR	Attribute R: Read only Blank: Read/write	Function	
CR4703		Action when high-speed counter comparator CTC4 is turned ON	Automatic reset of CTH2 (ON: execute, OFF: do not execute)
CR4704			Comparator matching output disabled when comparator CTC4 is turned ON (ON: execute, OFF: do not execute)
CR4705			Comparator matching output turned OFF when comparator CTC4 is turned ON (ON: execute, OFF: do not execute)
CR4706			Comparator matching output turned ON when comparator CTC4 is turned ON (ON: execute, OFF: do not execute)
CR4707			Comparator matching output inversion (ON/OFF) each time that comparator CTC4 is turned ON (ON: execute, OFF: do not execute)
CR4708		Action when high-speed counter comparator CTC5 is turned ON	Comparator matching output disabled when comparator CTC5 is turned ON (ON: execute, OFF: do not execute)
CR4709			Comparator matching output turned OFF when comparator CTC5 is turned ON (ON: execute, OFF: do not execute)
CR4710			Comparator matching output turned ON when comparator CTC5 is turned ON (ON: execute, OFF: do not execute)
CR4711			Comparator matching output inversion (ON/OFF) each time that comparator CTC5 is turned ON (ON: execute, OFF: do not execute)
CR4712			Automatic reset of CTH2 (ON: execute, OFF: do not execute)
CR4713 to CR4715		CTH2 count input mode selection	

	Single Phase		1x	2x	4x	2-Pulse
	W/O Direction	W/ Direction				
CR4713	OFF	ON	OFF	ON	OFF	ON
CR4714	OFF	OFF	ON	ON	OFF	OFF
CR4715	OFF	OFF	OFF	OFF	ON	ON

CR4800	R	CTH3 dedicated internal clock (50 ns)
CR4801	R	CTH3 dedicated internal clock (1 μs)
CR4802	R	CTH3 dedicated internal clock (10 μs)
CR4803	R	CTH3 dedicated internal clock (100 μs)
CR4804	-	Reserved for the system
CR4805	R	Indicates the presence of CTH3 (ON: present, OFF: not present)
CR4806		Detects overflows and underflows of CTH3 (ON: detected, OFF: not detected)
CR4807	R	Indicates the change direction of CTH3 (ON: down counting, OFF: up counting)
CR4808, CR4809		CTH3 external signal preset

	Not Used	Rising Edge	Falling Edge	Level
CR4808	OFF	ON	OFF	ON
CR4809	OFF	OFF	ON	ON

CR4810		CTH3 count mode selection (ON: ring, OFF: linear)
CR4811		CTH3 preset disabled (ON: execute, OFF: do not execute)
CR4812		CTH3 internal enable relay (ON: counting enabled, OFF: counting disabled)
CR4813 to CR4902	-	Reserved for the system

CR4903		Action when high-speed counter comparator CTC6 is turned ON	Automatic reset of CTH3 (ON: execute, OFF: do not execute)
CR4904			Comparator matching output disabled when comparator CTC6 is turned ON (ON: execute, OFF: do not execute)
CR4905			Comparator matching output turned OFF when comparator CTC6 is turned ON (ON: execute, OFF: do not execute)
CR4906			Comparator matching output turned ON when comparator CTC6 is turned ON (ON: execute, OFF: do not execute)
CR4907			Comparator matching output inversion (ON/OFF) each time that comparator CTC6 is turned ON (ON: execute, OFF: do not execute)

<table>
<tr><th>CR</th><th>Attribute
R: Read only
Blank: Read/write</th><th colspan="2">Function</th></tr>
<tr><td>CR4908</td><td></td><td rowspan="5">Action when high-speed counter comparator CTC7 is turned ON</td><td>Comparator matching output disabled when comparator CTC7 is turned ON (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4909</td><td></td><td>Comparator matching output turned OFF when comparator CTC7 is turned ON (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4910</td><td></td><td>Comparator matching output turned ON when comparator CTC7 is turned ON (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4911</td><td></td><td>Comparator matching output inversion (ON/OFF) each time that comparator CTC7 is turned ON (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4912</td><td></td><td>Automatic reset of CTH3 (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR4913 to CR4915</td><td></td><td colspan="2">Multiplication mode selection of CTH3
<table>
<tr><th></th><th colspan="2">Single Phase</th><th rowspan="2">1x</th><th rowspan="2">2x</th><th rowspan="2">4x</th><th rowspan="2">2-Pulse</th></tr>
<tr><th>W/O Direction</th><th>W/ Direction</th></tr>
<tr><td>CR4913</td><td>OFF</td><td>ON</td><td>OFF</td><td>ON</td><td>OFF</td><td>ON</td></tr>
<tr><td>CR4914</td><td>OFF</td><td>OFF</td><td>ON</td><td>ON</td><td>OFF</td><td>OFF</td></tr>
<tr><td>CR4915</td><td>OFF</td><td>OFF</td><td>OFF</td><td>OFF</td><td>ON</td><td>ON</td></tr>
</table></td></tr>
<tr><td>CR5000 to CR5315</td><td>-</td><td colspan="2">Reserved for the system</td></tr>
<tr><td>CR5400</td><td></td><td rowspan="5">Frequency counter (CTH0)</td><td>Start operation (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR5401, CR5402</td><td></td><td>CTH0 input source setting
<table>
<tr><th></th><th colspan="2">Single Phase</th><th rowspan="2">Double Phase 1x</th><th rowspan="2">2-Pulse</th></tr>
<tr><th>W/O Direction</th><th>W/ Direction</th></tr>
<tr><td>CR5401</td><td>OFF</td><td>ON</td><td>OFF</td><td>ON</td></tr>
<tr><td>CR5402</td><td>OFF</td><td>OFF</td><td>ON</td><td>ON</td></tr>
</table></td></tr>
<tr><td>CR5403</td><td></td><td>Switches between Hz and rpm (ON: rpm, OFF: Hz)</td></tr>
<tr><td>CR5404</td><td></td><td>Function in which one rotation is performed to update the rotation speed (ON: use, OFF: do not use)</td></tr>
<tr><td>CR5405</td><td></td><td>Input source switching function (ON: use, OFF: do not use)</td></tr>
<tr><td>CR5406, CR5407</td><td>-</td><td colspan="2">Reserved for the system</td></tr>
<tr><td>CR5408</td><td></td><td colspan="2">Specified frequency pulse output (CTH0; ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR5409</td><td></td><td colspan="2">Specified frequency pulse output (CTH0) error</td></tr>
<tr><td>CR5410 to CR5415</td><td>-</td><td colspan="2">Reserved for the system</td></tr>
<tr><td>CR5500</td><td></td><td rowspan="5">Frequency counter (CH1)</td><td>Start operation (ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR5501, CR5502</td><td></td><td>CTH1 input source setting
<table>
<tr><th></th><th colspan="2">Single Phase</th><th rowspan="2">Double Phase 1x</th><th rowspan="2">2-Pulse</th></tr>
<tr><th>W/O Direction</th><th>W/ Direction</th></tr>
<tr><td>CR5501</td><td>OFF</td><td>ON</td><td>OFF</td><td>ON</td></tr>
<tr><td>CR5502</td><td>OFF</td><td>OFF</td><td>ON</td><td>ON</td></tr>
</table></td></tr>
<tr><td>CR5503</td><td></td><td>Switches between Hz and rpm (ON: rpm, OFF: Hz)</td></tr>
<tr><td>CR5504</td><td></td><td>Function in which one rotation is performed to update the rotation speed (ON: use, OFF: do not use)</td></tr>
<tr><td>CR5505</td><td></td><td>Input source switching function (ON: use, OFF: do not use)</td></tr>
<tr><td>CR5506, CR5507</td><td>-</td><td colspan="2">Reserved for the system</td></tr>
<tr><td>CR5508</td><td></td><td colspan="2">Specified frequency pulse output (CTH1; ON: execute, OFF: do not execute)</td></tr>
<tr><td>CR5509</td><td></td><td colspan="2">Specified frequency pulse output (CTH1) error</td></tr>
<tr><td>CR5510 to CR5515</td><td>-</td><td colspan="2">Reserved for the system</td></tr>
</table>

CR	Attribute R: Read only Blank: Read/write	Function	
CR5600		Frequency counter (CH2)	Start operation (ON: execute, OFF: do not execute)
CR5601, CR5602		Frequency counter (CH2)	CTH2 input source setting

	Single Phase		Double Phase 1x	2-Pulse
	W/O Direction	W/ Direction		
CR5601	OFF	ON	OFF	ON
CR5602	OFF	OFF	ON	ON

CR5603		Frequency counter (CH2)	Switches between Hz and rpm (ON: rpm, OFF: Hz)
CR5604		Frequency counter (CH2)	Function in which one rotation is performed to update the rotation speed (ON: use, OFF: do not use)
CR5605		Frequency counter (CH2)	Input source switching function (ON: use, OFF: do not use)
CR5606, CR5607	-	Reserved for the system	
CR5608		Specified frequency pulse output (CTH2; ON: execute, OFF: do not execute)	
CR5609		Specified frequency pulse output (CTH2) error	
CR5610 to CR5615	-	Reserved for the system	
CR5700		Frequency counter (CTH3)	Start operation (ON: execute, OFF: do not execute)
CR5701, CR5702		Frequency counter (CTH3)	CTH3 input source setting

	Single Phase		Double Phase 1x	2-Pulse
	W/O Direction	W/ Direction		
CR5701	OFF	ON	OFF	ON
CR5702	OFF	OFF	ON	ON

CR5703		Frequency counter (CTH3)	Switches between Hz and rpm (ON: rpm, OFF: Hz)
CR5704		Frequency counter (CTH3)	Function in which one rotation is performed to update the rotation speed (ON: use, OFF: do not use)
CR5705		Frequency counter (CTH3)	Input source switching function (ON: use, OFF: do not use)
CR5706, CR5707	-	Reserved for the system	
CR5708		Specified frequency pulse output (CTH3; ON: execute, OFF: do not execute)	
CR5709		Specified frequency pulse output (CTH3) error	
CR5710 to CR5715	-	Reserved for the system	
CR5800 to CR5915	-	Reserved for the system	
CR6000 to CR6515		For built-in serial assignment	
CR6600 to CR7115		For extension function assignment 1	
CR7200 to CR7715		For extension function assignment 2	
CR7800 to CR7915	-	Reserved for the system	
CR8000		CPU positioning function, axis 1	Forced stop
CR8001		CPU positioning function, axis 1	Deceleration stop
CR8002		CPU positioning function, axis 1	Error clear
CR8003		CPU positioning function, axis 1	Warning clear
CR8004		CPU positioning function, axis 1	Current coordinate change request (level detection)
CR8005		CPU positioning function, axis 1	Operating speed change request (level detection)
CR8006		CPU positioning function, axis 1	Target coordinate change request (level detection)
CR8007 to CR8011	-	CPU positioning function, axis 1	Reserved for the system
CR8012		CPU positioning function, axis 1	Limit switch CW direction input
CR8013		CPU positioning function, axis 1	Limit switch CCW direction input
CR8014		CPU positioning function, axis 1	Origin sensor input
CR8015		CPU positioning function, axis 1	Stop sensor input

CR	Attribute R: Read only Blank: Read/write	Function	
CR8100		CPU positioning function, axis 2	Forced stop
CR8101			Deceleration stop
CR8102			Error clear
CR8103			Warning clear
CR8104			Current coordinate change request (level detection)
CR8105			Operating speed change request (level detection)
CR8106			Target coordinate change request (level detection)
CR8107 to CR8111	-		Reserved for the system
CR8112			Limit switch CW direction input
CR8113			Limit switch CCW direction input
CR8114			Origin sensor input
CR8115			Stop sensor input
CR8200		CPU positioning function, axis 3	Forced stop
CR8201			Deceleration stop
CR8202			Error clear
CR8203			Warning clear
CR8204			Current coordinate change request (level detection)
CR8205			Operating speed change request (level detection)
CR8206			Target coordinate change request (level detection)
CR8207 to CR8211			Reserved for the system
CR8212			Limit switch CW direction input
CR8213			Limit switch CCW direction input
CR8214			Origin sensor input
CR8215			Stop sensor input
CR8300		CPU positioning function, axis 4	Forced stop
CR8301			Deceleration stop
CR8302			Error clear
CR8303			Warning clear
CR8304			Current coordinate change request (level detection)
CR8305			Operating speed change request (level detection)
CR8306			Target coordinate change request (level detection)
CR8307 to CR8311			Reserved for the system
CR8312			Limit switch CW direction input
CR8313			Limit switch CCW direction input
CR8314			Origin sensor input
CR8315			Stop sensor input
CR8400	R	CPU positioning function, axis 1	Turned ON during pulse output
CR8401	R		Positioning complete relay
CR8402	R		Error
CR8403	R		Warning
CR8404	R		Origin returning operation in progress
CR8405	R		Origin returning operation complete
CR8406 to CR8413	-		Reserved for the system
CR8414	R		Indicates the presence of axis 1 (ON: present, OFF: not present)
CR8415	R		Comparator 2 matching relay

CR	Attribute R: Read only Blank: Read/write	Function	
CR8500	R	CPU positioning function, axis 2	Turned ON during pulse output
CR8501	R		Positioning complete relay
CR8502	R		Error
CR8503	R		Warning
CR8504	R		Origin returning operation in progress
CR8505	R		Origin returning operation complete
CR8506 to CR8513	-		Reserved for the system
CR8514	R		Indicates the presence of axis 2 (ON: present, OFF: not present)
CR8515	R		Comparator 2 matching relay
CR8600	R	CPU positioning function, axis 3	Turned ON during pulse output
CR8601	R		Positioning complete relay
CR8602	R		Error
CR8603	R		Warning
CR8604	R		Origin returning operation in progress
CR8605	R		Origin returning operation complete
CR8606 to CR8613	-		Reserved for the system
CR8614	R		Indicates the presence of axis 3 (ON: present, OFF: not present)
CR8615	R		Comparator 2 matching relay
CR8700	R	CPU positioning function, axis 4	Turned ON during pulse output
CR8701	R		Positioning complete relay
CR8702	R		Error
CR8703	R		Warning
CR8704	R		Origin returning operation in progress
CR8705	R		Origin returning operation complete
CR8706 to CR8713	R		Reserved for the system
CR8714	R		Indicates the presence of axis 4 (ON: present, OFF: not present)
CR8715	R		Comparator 2 matching relay
CR8800 to CR8915	-	Reserved for the system	

Control Memory Entries CM (For KV Nano series)

Control memory entries (CM0000 to CM8999) can be used to perform operations such as checking the operation status of the base unit and setting and monitoring the built-in functions.
Control memory entries reserved for the system cannot be used.

CM	Attribute R: Read only Blank: Read/write	Function		Hold O: Hold Set: Holding can be set
CM0000, CM0001		Operator panel KV-D30 Page 0	Device setting displayed on the first line	Set
CM0002, CM0003			Device setting displayed on the second line	Set
CM0004, CM0005			Device setting displayed on the third line	Set
CM0006, CM0007			Device setting displayed on the fourth line	Set
CM0008, CM0009			Reserved for the system	Set
CM0010 to CM0199		Page 1 to 19 (the same as CM0000 to CM0009)		Set
CM0200, CM0201		Operator panel KV-D30 Page 0	Attribute setting displayed on the first line	Set
CM0202, CM0203			Attribute setting displayed on the second line	Set
CM0204, CM0205			Attribute setting displayed on the third line	Set
CM0206, CM0207			Attribute setting displayed on the fourth line	Set
CM0208			Template 1 setting	Set
CM0209			Template 2 setting	Set
CM0210 to CM0399		Page 1 to 19 (the same as CM0200 to CM0209)		Set
CM0400		Operator panel KV-D30	For enabling and disabling transitions	Set
CM0401			For the page switching setting	Set
CM0402			Current page displayed	Set
CM0403 to CM409			Reserved for the system	Set
CM0410 to CM415			Direct access function	Set
CM0416 to CM0499			Reserved for the system	Set
CM0500 to CM699	-	Reserved for the system		
CM0700	R	Read from RTC (year)		
CM0701	R	Read from RTC (month)		
CM0702	R	Read from RTC (day)		
CM0703	R	Read from RTC (hour)		
CM0704	R	Read from RTC (minute)		
CM0705	R	Read from RTC (second)		
CM0706	R	Read from RTC (day of the week)		
CM0707	-	Reserved for the system		
CM0708, CM709	R	32-bit, 1 ms, free-run counter		
CM0710		Reserved for the system		
CM0711	R	CPU function version		

CM	Attribute R: Read only Blank: Read/write	Function	Hold ○: Hold Set: Holding can be set
CM0712 to CM719	-	Reserved for the system	
CM0720	R	Scan time measured value (in units of 10 μs)	
CM0721		Set value during fixed scan time operation (in units of 10 μs)	
CM0722	R	Scan time (in units of 10 μs) when the scan time setting (the fixed scan time operation and END processing time settings) is exceeded	
CM0723	R	END processing time measured value (in units of 10 μs)	
CM0724, CM725	-	Reserved for the system	
CM0726	R	Minimum scan time value (in units of 10 μs)	
CM0727	R	Maximum scan time value (in units of 10 μs)	
CM0728		END processing time set value (in units of 10 μs)	
CM0729	R	END processing time (in units of 10 μs) when the scan time setting (the fixed scan time operation and END processing time settings) is exceeded	
CM0730 to CM1599	-	Reserved for the system	
CM1600, CM1601	R	Input capture when INT0 occurs	
CM1602, CM1603	R	Input capture when INT1 occurs	
CM1604, CM1605	R	Input capture when INT2 occurs	
CM1606, CM1607	R	Input capture when INT3 occurs	
CM1608 to CM1619	-	Reserved for the system	
CM1620		Input time constant setting	○
CM1621 to CM1629	-	Reserved for the system	
CM1630, CM1631	R	Analog volume 0	
CM1632, CM1633	R	Analog volume 1	
CM1634 to CM1649	-	Reserved for the system	
CM1650	R	Number of sensor setting instruction executions	
CM1651 to CM1657	-	Reserved for the system	
CM1658	R	Number of consecutive project password authentication failures	○
CM1659	R	Total number of project password authentication failures	○
CM1660 to CM1669	R	Free space in the logging or tracing ring buffer (in units of kB; ID0 to ID9)	
CM1670 to CM1679	-	Reserved for the system	
CM1680	R	Logging or tracing 0 file save counter	○
CM1681	R	Logging or tracing 1 file save counter	○
CM1682	R	Logging or tracing 2 file save counter	○
CM1683	R	Logging or tracing 3 file save counter	○
CM1684	R	Logging or tracing 4 file save counter	○
CM1685	R	Logging or tracing 5 file save counter	○
CM1686	R	Logging or tracing 6 file save counter	○
CM1687	R	Logging or tracing 7 file save counter	○
CM1688	R	Logging or tracing 8 file save counter	○
CM1689	R	Logging or tracing 9 file save counter	○

CM	Attribute R: Read only Blank: Read/write	Function		Hold ○: Hold Set: Holding can be set
CM1690 to CM1699	-	Reserved for the system		
CM1700	R	AutoLoad folder No. (request)		○
CM1701	R	RunLoad folder No. (request)		○
CM1702	R	Folder No. during project load (request)		○
CM1703	R	Folder No. during project save (request)		○
CM1704 to CM1709		Reserved for the system		
CM1710	R	Completion code of automatic loading when power is turned on		
CM1711	R	Automatic loading folder No. during power-on (complete)		
CM1712	R	Automatic loading completion code during PROG→RUN		○
CM1713	R	Automatic loading folder No. during PROG→RUN (complete)		○
CM1714	R	Completion code during project load		○
CM1715	R	Folder No. during project load (complete)		○
CM1716	R	Completion code during project save		○
CM1717	R	Folder No. during project save (complete)		○
CM1718, CM1719	-	Reserved for the system		
CM1720		Access window	User message 1	○
CM1721 to CM1737			User message 2 (up to 24 characters + end code (00H))	○
CM1738			Access window operation disabled setting	○
CM1739			Reserved for the system	
CM1740 to CM1756			Ladder project name (up to 32 characters + end code (00H))	
CM1757 to CM1763			Project name to be displayed in the access window (up to 12 characters + end code (00H))	
CM1764 to CM1779	-		Reserved for the system	
CM1780			Access window initial screen setting, enabled	○
CM1781			Access window initial screen setting, device type	○
CM1782			Access window initial screen setting, screen ID	○
CM1783			Access window initial screen setting, screen ID	○
CM1784			Access window initial screen setting, device number	○
CM1785			Access window initial screen setting, device number	○
CM1786			Access window initial screen setting, display format	○
CM1787			Access window initial screen setting, device name representation	○
CM1788			Access window initial screen setting, key lock status	○
CM1789			Access window initial screen setting, display language	○
CM1790 to CM2199	-		Reserved for the system	
CM2200		Detailed information of the latest serious error, 1 unit	Year and month	○
CM2201			Day	○
CM2202			Hour	○
CM2203			Minute	○
CM2204			Seconds	○
CM2205			Number	○
CM2206			Detailed information, number of valid pieces of data	○
CM2207			Detailed information, piece 01	○
CM2208			Detailed information, piece 02	○
:				○
CM2226			Detailed information, piece 20	○
CM2227 to CM2249	-		Reserved for the system	

CM	Attribute R: Read only Blank: Read/write	Function		Hold O: Hold Set: Holding can be set
CM2250		Detailed information of the latest minor error, 1 unit*1	Year and month	○
CM2251			Day	○
CM2252			Hour	○
CM2253			Minute	○
CM2254			Seconds	○
CM2255			Number	○
CM2256			Detailed information, number of valid pieces of data	○
CM2257			Detailed information, piece 01	○
CM2258			Detailed information, piece 02	○
:				○
CM2276			Detailed information, piece 20	○
CM2277 to CM2388	-	Reserved for the system		
CM2389	R	Number of connected expansion units		
CM2390	R	Error code for memory card instructions		
CM2391		Power OFF when accessing memory card		○
CM2392		Memory card retry occurrence times		○
CM2393		Memory card check error times		○
CM2394		Memory card timeout occurrence times		○
CM2395	-	Reset delay time during memory card errors		○
CM2396 to CM2399	-	Reserved for the system		
CM2400 to CM2415		Number of interrupts (INT0 to INT3, CTC0 to CTC7, and positioning axis 1 to axis 4)		
CM2416 to CM4799	-	Reserved for the system		
CM4800		High-speed counter function (CTH0)	Constant when rotation direction is detected	○
CM4801			Enable, clear, and sign setting	○
CM4802, CM4803			Ring counter lower limit	○
CM4804, CM4805			Ring counter upper limit	○
CM4806, CM4807			Preset input	○
CM4808, CM4809	-		Reserved for the system	
CM4810		High-speed counter function (CTH1)	Constant when rotation direction is detected	○
CM4811			Enable, clear, and sign setting	○
CM4812, CM4813			Ring counter lower limit	○
CM4814, CM4815			Ring counter upper limit	○
CM4816, CM4817			Preset input	○
CM4818, CM4819	-		Reserved for the system	
CM4820		High-speed counter function (CTH2)	Constant when rotation direction is detected	○
CM4821			Enable, clear, and sign setting	○
CM4822, CM4823			Ring counter lower limit	○
CM4824, CM4825			Ring counter upper limit	○
CM4826, CM4827			Preset input	○

CM	Attribute R: Read only Blank: Read/write	Function		Hold ○: Hold Set: Holding can be set
CM4828, CM4829	-	High-speed counter function (CTH3)	Reserved for the system	
CM4830			Constant when rotation direction is detected	○
CM4831			Enable, clear, and sign setting	○
CM4832, CM4833			Ring counter lower limit	○
CM4834, CM4835			Ring counter upper limit	○
CM4836, CM4837			Preset input	○
CM4838, CM4839	-		Reserved for the system	○
CM4840 to CM4899	-	Reserved for the system		
CM4900, CM4901	R	Frequency counter function (CTH0)	Frequency measured result (Hz or rpm)	
CM4902			Number of pulses per rotation	○
CM4903			Number of scans	○
CM4904			External input source	○
CM4905	-		Reserved for the system	
CM4906		Specified frequency pulse output function (CH0)	Set frequency (Hz)	○
CM4908			Duty cycle (0 to 100%)	○
CM4909	-		Reserved for the system	
CM4910, CM4911	R	Frequency counter function (CTH1)	Frequency measured result (Hz or rpm)	
CM4912			Number of pulses per rotation	○
CM4913			Number of scans	○
CM4914			External input source	○
CM4915	-		Reserved for the system	
CM4916		Specified frequency pulse output function (CH1)	Set frequency (Hz)	○
CM4918			Duty cycle (0 to 100%)	○
CM4919	-		Reserved for the system	
CM4920, CM4921	R	Frequency counter function (CTH2)	Frequency measured result (Hz or rpm)	
CM4922			Number of pulses per rotation	○
CM4923			Number of scans	○
CM4924			External input source	○
CM4925	-		Reserved for the system	
CM4926		Specified frequency pulse output function (CH2)	Set frequency (Hz)	○
CM4928			Duty cycle (0 to 100%)	○
CM4929	-		Reserved for the system	
CM4930, CM4931	R	Frequency counter function (CTH3)	Frequency measured result (Hz or rpm)	
CM4932			Number of pulses per rotation	○
CM4933			Number of scans	○
CM4934			External input source	○
CM4935	-		Reserved for the system	
CM4936		Specified frequency pulse output function (CH3)	Set frequency (Hz)	○
CM4938			Duty cycle (0 to 100%)	○
CM4939			Reserved for the system	
CM4940 to CM4999	-	Reserved for the system		
CM5000 to CM5999		For built-in serial assignment		

CM	Attribute R: Read only Blank: Read/write	Function		Hold ○: Hold Set: Holding can be set
CM6000 to CM6999		For extension function assignment 1		
CM7000 to CM7999		For extension function assignment 2		
CM8000, CM8001		CPU positioning function, axis 1 Point parameter 1	Target value or travel	Set
CM8002			Acceleration rate	Set
CM8003			Deceleration rate	Set
CM8004, CM8005			Operation frequency	Set
CM8006			Operation mode	Set
CM8007			Reserved for the system	Set
CM8008			Travel after stop sensor input	Set
CM8009	-		Reserved for the system	
CM8010 to CM8199		CPU positioning function, axis 1	Point parameters 2 to 20 (the same as CM8000 to CM8009)	Set
CM8200, CM8201		CPU positioning function, axis 2 Point parameter 1	Target value or travel	Set
CM8202			Acceleration rate	Set
CM8203			Deceleration rate	Set
CM8204, CM8205			Operation frequency	Set
CM8206			Operation mode	Set
CM8207			Reserved for the system	Set
CM8208			Travel after stop sensor input	Set
CM8209	-		Reserved for the system	
CM8210 to CM8399		CPU positioning function, axis 2	Point parameters 2 to 20 (the same as CM8200 to CM8209)	Set
CM8400, CM8401		CPU positioning function, axis 3 Point parameter 1	Target value or travel	Set
CM8402			Acceleration rate	Set
CM8403			Deceleration rate	Set
CM8404, CM8405			Operation frequency	Set
CM8406			Operation mode	Set
CM8407			Reserved for the system	Set
CM8408			Travel after stop sensor input	Set
CM8409	-		Reserved for the system	
CM8410 to CM8599		CPU positioning function, axis 3	Point parameters 2 to 20 (the same as CM8400 to CM8409)	Set
CM8600, CM8601		CPU positioning function, axis 4 Point parameter 1	Target value or travel	Set
CM8602			Acceleration rate	Set
CM8603			Deceleration rate	Set
CM8604, CM8605			Operation frequency	Set
CM8606			Operation mode	Set
CM8607			Reserved for the system	Set
CM8608			Travel after stop sensor input	Set
CM8609	-		Reserved for the system	
CM8610 to CM8799		CPU positioning function, axis 4	Point parameters 2 to 20 (the same as CM8600 to CM8609)	Set

CM	Attribute R: Read only Blank: Read/write	Function		Hold ○: Hold Set: Holding can be set
CM8800		CPU positioning function, axis 1	I/O setting	
CM8801			Sensor enable	
CM8802, CM8803			Comparator 0, current value	
CM8804, CM8805			Comparator 1, current value	
CM8806, CM8807			Comparator 2, positioning	
CM8808			Starting speed	
CM8809			Origin returning operation: starting frequency	
CM8810			Origin returning operation: acceleration rate	
CM8811			Origin returning operation: deceleration rate	
CM8812, CM8813			Origin returning operation: operation frequency	
CM8814			Origin returning operation: detailed settings	
CM8815			JOG operation: starting frequency	
CM8816			JOG operation: acceleration rate	
CM8817			JOG operation: deceleration rate	
CM8818, CM8819			JOG operation: operation frequency	
CM8820, CM8821			Home position	
CM8822, CM8823	-		Reserved for the system	
CM8824, CM8825			Current coordinate change value	
CM8826, CM8827			Operation speed change value during speed control mode	
CM8828, CM8829			Target coordinate change value	
CM8830, CM8831	R		Output pulse, current coordinate	
CM8832, CM8833	R		Output frequency, current value	
CM8834	R		Error code	
CM8835	R		Execution point number	
CM8836 to CM8839	-		Reserved for the system	

CM	Attribute R: Read only Blank: Read/write	Function		Hold ○: Hold Set: Holding can be set
CM8840		CPU positioning function, axis 2	I/O setting	
CM8841			Sensor enable	
CM8842, CM8843			Comparator 0, current value	
CM8844, CM8845			Comparator 1, current value	
CM8846, CM8847			Comparator 2, positioning	
CM8848			Starting speed	
CM8849			Origin returning operation: starting frequency	
CM8850			Origin returning operation: acceleration rate	
CM8851			Origin returning operation: deceleration rate	
CM8852, CM8853			Origin returning operation: operation frequency	
CM8854			Origin returning operation: detailed settings	
CM8855			JOG operation: starting frequency	
CM8856			JOG operation: acceleration rate	
CM8857			JOG operation: deceleration rate	
CM8858, CM8859			JOG operation: operation frequency	
CM8860, CM8861			Home position	
CM8862, CM8863	-		Reserved for the system	
CM8864, CM8865			Current coordinate change value	
CM8866, CM8867			Operation speed change value during speed control mode	
CM8868, CM8869			Target coordinate change value	
CM8870, CM8871	R		Output pulse, current coordinate	
CM8872, CM8873	R		Output frequency, current value	
CM8874	R		Error code	
CM8875	R		Execution point number	
CM8876 to CM8879	-		Reserved for the system	

CM	Attribute R: Read only Blank: Read/write	Function		Hold ○: Hold Set: Holding can be set
CM8880		CPU positioning function, axis 3	I/O setting	
CM8881			Sensor enable	
CM8882, CM8883			Comparator 0, current value	
CM8884, CM8885			Comparator 1, current value	
CM8886, CM8887			Comparator 2, positioning	
CM8888			Starting speed	
CM8889			Origin returning operation: starting frequency	
CM8890			Origin returning operation: acceleration rate	
CM8891			Origin returning operation: deceleration rate	
CM8892, CM8893			Origin returning operation: operation frequency	
CM8894			Origin returning operation: detailed settings	
CM8895			JOG operation: starting frequency	
CM8896			JOG operation: acceleration rate	
CM8897			JOG operation: deceleration rate	
CM8898, CM8898			JOG operation: operation frequency	
CM8900, CM8901			Home position	
CM8902, CM8903			Reserved for the system	
CM8904, CM8905			Current coordinate change value	
CM8906, CM8907			Operation speed change value during speed control mode	
CM8908, CM8909			Target coordinate change value	
CM8910, CM8911	R		Output pulse, current coordinate	
CM8912, CM8913	R		Output frequency, current value	
CM8914	R		Error code	
CM8915	R		Execution point number	
CM8916 to CM8919			Reserved for the system	

CM	Attribute R: Read only Blank: Read/write	Function		Hold ○: Hold Set: Holding can be set
CM8920		CPU positioning function, axis 4	I/O setting	
CM8921			Sensor enable	
CM8922, CM8923			Comparator 0, current value	
CM8924, CM8925			Comparator 1, current value	
CM8926, CM8927			Comparator 2, positioning	
CM8928			Starting speed	
CM8929			Origin returning operation: starting frequency	
CM8930			Origin returning operation: acceleration rate	
CM8931			Origin returning operation: deceleration rate	
CM8932, CM8933			Origin returning operation: operation frequency	
CM8934			Origin returning operation: detailed settings	
CM8935			JOG operation: starting frequency	
CM8936			JOG operation: acceleration rate	
CM8937			JOG operation: deceleration rate	
CM8938, CM8939			JOG operation: operation frequency	
CM8940, CM8941			Home position	
CM8942, CM8943			Reserved for the system	
CM8944, CM8945			Current coordinate change value	
CM8946, CM8947			Operation speed change value during speed control mode	
CM8948, CM8949			Target coordinate change value	
CM8950, CM8951	R		Output pulse, current coordinate	
CM8952, CM8953	R		Output frequency, current value	
CM8954	R		Error code	
CM8955	R		Execution point number	
CM8956 to CM8959			Reserved for the system	

- Error number 128 (calculation error)

Detailed Information, Piece n	Explanation
Detailed information, piece 1	Error details 0 (Not used.) 1 Incorrect indirect specification. 2 The indirectly specified target is T/C. 3 Incorrect simple indirect specification. 4 Outside of operand range. 5 Floating-point real number overflow. 6 Incorrect device specification. 7 The ROOT instruction operand is negative. 8 No bits turned ON with the ENCO instruction. 9 Floating-point real number format error. 10 Incorrect trigonometric function instruction operand. 11 Data could not be converted. 12 Incorrect table specification. 13 Incorrect time comparison instruction operand. 14 Incorrect cam switch operand. 15 Incorrect frequency counter instruction operand. 16 Divide by zero. 17 Illegal logging ID. 18 Direct processing error. 19 Unit on which direct processing cannot be performed. 20 Jump destination subroutine does not exist. 21 Data being written during RUN mode, so instruction cannot be executed. 22 Error for expansion unit-dedicated instructions (unit number). 23 Reserved for the system. 24 PIDAT, outside of operation setting range. 25 PIDAT, outside of sampling period range. 26 PIDAT, outside of comparison constant range. 27 PIDAT, outside of integral constant range. 28 PIDAT, outside of differential constant range. 29 PIDAT comparison constant, two-degree freedom parameters outside of range. 30 PIDAT integral constant, two-degree freedom parameters outside of range. 31 PIDAT differential constant, two-degree freedom parameters outside of range. 32 PIDAT, outside of operation quantity upper limit or lower limit range. 33 PIDAT, outside of operation quantity change width limit range. 34 PIDAT, outside of measured value change width range. 35 PIDAT, AT state transition error. 36 PIDAT, outside of operation control flag range. 37 Operation period exceeded. 38 PIDAT, outside of AT adjustment parameter range. 39 PIDAT, outside of AT hysteresis range. 40 PIDAT, outside of AT timeout range. 41 PIDAT, outside of AT calculation control mode setting range. 42 PIDAT, AT timeout. 43 PIDAT, AT comparison constant exceeded. 44 PIDAT, AT integral constant exceeded. 45 PIDAT, AT differential constant exceeded. 46 Root of negative number cannot be calculated. 47 Incorrect ABS instruction operand. 48 Calculation overflow. 49 UR, UM, or UV unit not connected. 50 Unit device (R) not assigned. 51 Unit device (DM) not assigned.

Detailed Information, Piece n	Explanation
Detailed information, piece 2	Fixed to 00H
Detailed information, piece 3	Upper bits of the step number at which the error occurred
Detailed information, piece 4	Lower bits of the step number at which the error occurred
Detailed information, pieces 5 to 20	The module name is saved in SJIS format (any parts not occupied with the module name are "00H")

- Error numbers 50, 53, 55, 56, 58, and 59 (unit number errors)

Detailed Information, Piece n	Explanation
Detailed information, piece 1	Unit number (the unit number will not be stored if it cannot be determined according to the occurrence conditions of the error)
Detailed information, pieces 2 to 20	Fixed to 00H

- Error number 54 (number of units error)

Detailed Information, Piece n	Explanation
Detailed information, piece 1	The number of units as specified in the unit setup information
Detailed information, piece 2	The number of units that are actually connected
Detailed information, pieces 3 to 20	Fixed to 00H

- Error number 60 (number of expansion I/O points error)

Detailed Information, Piece n	Explanation
Detailed information, piece 1	Upper limit of the number of base unit expansion I/O points
Detailed information, piece 2	Actually connected number of expansion I/O points
Detailed information, pieces 3 to 20	Fixed to 00H

- Error number 61 (assigned device overlap error)

Detailed Information, Piece n	Explanation
Detailed information, piece 1	Unit number of the expansion unit whose assignment is being overlapped
Detailed information, piece 2	Type of device that is being overlapped (R—relay: 1, DM: 2)
Detailed information, piece 3	Number of the device that is being overlapped (channel number when the device type is R—relay, example: R1000 → 10)
Detailed information, pieces 4 to 20	Fixed to 00H

- Error numbers 96, 97, 98, 99, and 101 (extension adapter error)

Detailed Information, Piece n	Explanation
Detailed information, piece 1	Extension adapter number (left: 1, right: 2) (0 is stored when the slot cannot be determined)
Detailed information, piece 2	Reserved for the system
Detailed information, pieces 3 to 20	Fixed to 00H

- Error number 127 (automatic loading failure error)

Detailed Information, Piece n	Explanation
Detailed information, piece 1	Details of error For details, refer to "Errors during load/save execution".
Detailed information, pieces 2 to 20	00H Fixed

- Error number 129 (unit error)

Expansion Information, Piece n	Explanation
Expansion information, piece 1	Unit number
Expansion information, piece 2	Error number of each unit
Detailed information, pieces 3 to 20	Fixed to 00H

- Error numbers other than those listed above

Expansion Information, Piece n	Explanation
Detailed information, pieces 1 to 20	Fixed to 00H

ASCII Code Table

List of ASCII codes is shown below.

Low 4-bit \ High 4-bit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DLE	SP	0	@	P	`	p				ー	タ	ミ		
1	SOH	DC1	!	1	A	Q	a	q			。	ア	チ	ム		
2	STX	DC2	”	2	B	R	b	r			「	イ	ツ	メ		
3	ETX	DC3	#	3	C	S	c	s			」	ウ	テ	モ		
4	EOT	DC4	$	4	D	T	d	t			、	エ	ト	ヤ		
5	ENQ	NAK	%	5	E	U	e	u			・	オ	ナ	ユ		
6	ACK	SYN	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7	BEL	ETB	’	7	G	W	g	w			ァ	キ	ヌ	ラ		
8	BS	CAN	(	8	H	X	h	x			ィ	ク	ネ	リ		
9	HT	EM	)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
A	LF	SUB	*	:	J	Z	j	z			ェ	コ	ハ	レ		
B	VT	ESC	+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C	FF	→	,	<	L	\	l	|			ャ	シ	フ	ワ		
D	CR	←	—	=	M	]	m	}			ュ	ス	ヘ	ン		
E	SO	↑	.	>	N	^	n	~			ョ	セ	ホ	゛		
F	SI	↓	／	?	O	_	o	DEL			ッ	ソ	マ	゜		

List of Characters That Cannot Be Used

This appendix describes the characters that cannot be used in project names, module names, macro names, labels, and device comments.

● Characters that cannot be used in project names

Characters that cannot be used	! ″ # $ % & ‘ * , . / : ; < = > ?
	@ ¥ ^ ` | ~ (space) _

● Characters that cannot be used in program names

Characters that cannot be used	! ″ # $ % & ’ ( ) * , . / : ; < = > ?
	@ [¥] ^ ` { | } ~ (space) + *1 – *1
Characters that cannot be used as the first character	+ *2 – *2 0 1 2 3 4 5 6 7 8 9 _
Character and character strings that cannot be used as the last character or characters	_ _0 _1 _2 _3 _4 _5 _6 _7 _8 _9
Character string that cannot be used	Global グローバル Global 全局

*1 String that cannot be used only with KV-8000 with CPU function version 2.0 or later.

*2 String that cannot be used at positions other than the start when using KV-8000 with CPU function version 2.0 or later.

● Characters that cannot be used in function / function block / structure names

Characters that cannot be used	! ″ # $ % & ‘ * , . / : ; < = > ? @ ¥
	^ ‘ | ~ (スペース) (__) + *1 – *1
Characters that cannot be used at the start	+ *2 – *2 0 1 2 3 4 5 6 7 8 9 _
Characters or strings that cannot be used at the end	_ _0 _1 _2 _3 _4 _5 _6 _7 _8 _9
Strings that cannot be used	ｸﾞﾛｰﾊﾞﾙ グローバル Global 全局 全域 글로벌 The following are strings that cannot be used only with KV-8000 with CPU function version 2.0 or later • TIMER 定时器 定時器 타이머 COUNTER 计数器 計數器 카운터 FUN_ENO • Keywords specified in IEC61131-3 (For the contents of keyword, refer to "Characters that cannot be used in variable names".)

*1 This string cannot be used only with KV-8000 with CPU function version 2.0 or later.

*2 String that cannot be used at positions other than the start when using KV-8000 with CPU function version 2.0 or later.

● Characters that cannot be used in variable

Characters that cannot be used	! ″ # $ % & ’ ( ) * + , – . / : ; < =
	> ? @ [¥] ^ ` { | } ~ (space)
Characters that cannot be used as the first character	0 1 2 3 4 5 6 7 8 9 _
Character strings that cannot be used	Global グローバル Global 全局
	The same names as instructions
	Device names
	• (1) One alphabet character + R or (2) M + a number • One alphabet character + a number • X + (A to F) • Y + (A to F)

Character strings that cannot be used	• The following keywords, which are prescribed in KV script !!, #PRAGMA, ', (,), *, *=, +, +=, -, -=, .B, .D, .DF, .F, .L, .S, .T, .U, /, /=, <, <=, <>, =, >, >=, ABS, ACNT※1, ACOS, ADRADD, ADRDEC, ADRINC, ADRSET, ADRSUB, AJST, AND, ANDA, APR, AS, ASC, ASIN, ASLA, ASRA, ATAN, ATAN2, AVG, AWHIDE, AWSHOW, B, BANDC, BCMP, BCMPI, BCNT, BLD, BLDB, BMOV, BOUB, BOUT, BREAK, BRES, BSET, BSL, BSR, BSUM, BSWAP, BYBMOV, BYLMOV, C, CALL, CASE, CHR, CM, CNT, COM, COS, CPMGET, CPMSET, CPSASC, CR, CRC, CTC, CTH, DASC, DCNT, DEC, DECO, DEG, DELETE, DFLOAT, DI, DIC, DIM, DISB, DISF, DISN, DISS, DM, DMX, DO, DOUBLE, DR, DSER, ECALL, EI, ELSE, ELSE IF, EM, ENCO, END, END IF, END SELECT, END TYPE, END WHILE, ENRA, EORA, EXP, FALSE, FASC, FB, FBCALL※1, FBSTRT※1, FDEL, FEND, FIFOR, FIFOW, FIND, FINS, FLOAT, FM, FMOV, FOR, FRLDM, FRSET, FRSTM, FUN, FWRIT, GRY, HASC, IEDGE, IF, INC, INSERT, INSTR, INT, INTG, IS, IS_7500, IS_8000, IS_AD40, IS_AD40G, IS_AD40V, IS_AM40V, IS_B16R, IS_B16T, IS_B16X, IS_B8RC, IS_B8XTD, IS_C16XTD, IS_C32T, IS_C32X, IS_C32XTD, IS_C64T, IS_C64X, IS_CA02, IS_CL20, IS_DA40, IS_DA40V, IS_DN20, IS_EP21V, IS_FL20V, IS_H20G, IS_H20S, IS_H40S, IS_L21V, IS_LE21V, IS_LH20V, IS_LM21V, IS_MC20V, IS_MC40V, IS_ML16V, IS_MVL40, IS_N20V, IS_SAD04, IS_SC20, IS_SDA04, IS_SFP24XT, IS_SFP32XT, IS_SH04PL, IS_SIR32XT, IS_SSC02, IS_TF40, IS_TP40, IS_XCA02, IS_XCM02※1, IS_XH04ML, IS_XH16EC, IS_XH16ML, IS_XL202, IS_XL402, IS_XLE02, LDF, LDFB, LDP, LDPB, LEFT, LEN, LIFOR, LIFOW, LIMIT, LLFLT, LOG, LOG10, LOGD, LOGE, LR, LTRIM, MAX, MC, MCALL, MCOPY, MCR, MDEL, MDSTOP, MDSTRT, MEND, MFREE, MFREEK, MID, MIN, MMKDIR, MMOV, MOD, MPRINT, MPX, MR, MREAD, MREADL, MREN, MRMDIR, MSTAT, MSTRT, MWRIT, NEG, NEXT, NOT, OFF, ON, OR, ORA, P, PIDAT, PMOV, R, RAD, RAMP, RASC, RCPSASC, RDASC, REM, REPLACE, RES, RFASC, RFSCI, RFSCO, RFSFRC, RFSX, RFSY, RGRY, RHASC, RIGHT, RLA, RLNCA, RND, ROOT, RRA, RRNCA, RSEC, RTRIM, SDEL, SEC, SEG, SELECT, SELECT CASE, SER, SET, SFIND, SFINDN, SIN, SINS, SLA, SLEFT, SMID, SORT, SORTN, SPLIT, SPRD, SPWR, SQRT, SRA, SRGHT, SRPLC, SSVC, STEP, STR, STRIM, STRWORK, SWAP, T, TAN, TBCD, TBIN, THEN, TM, TMH, TMR, TMS, TMU, TO, TOD, TODF, TOF, TOL, TOS, TOU, TPOUT, TRGD, TRGR, TRIM, TRUE, TYPE, UFILL, UFRSM, UFSUS, UM, UMALLOC, UNIB, UNIF, UNIN, UNIS, UNTIL, UPSTOP, UPSTRT, UR, UREAD, UV, UWRIT, U_CCTRCV, U_CCTSND, U_CCTST, U_CCTTO, U_CERR, U_CREG, U_CSTOP, U_CSTRT, U_DISOCP, U_DISOVP, U_ELCYC, U_ELSTAT, U_FCSTAT, U_MLATTA, U_MLSTAT, U_MLSUB, U_MLTEXT, U_MLTO, U_MLTOGR, U_MSGRCV, U_MSGSND, U_MSGST, U_MSGTO, U_NDSTAT, U_NDTO, U_PLCYC, U_PLSTAT, U_PRDBUF, U_RDAD, U_RDADB, U_RDAERC, U_RDBF, U_RDCNT, U_RDCPOS, U_RDCPT0, U_RDCPT1, U_RDCVEL, U_RDFPOS, U_RDFTRQ, U_RDFVEL, U_RDHZ, U_RDMD, U_RDMERC, U_RDPMC, U_RDPPB, U_RDPPNT, U_RDPSB, U_RDRPM, U_RDSERC, U_RDSPB, U_RDSVPB, U_RDSYSB, U_SBMON, U_SBNAME, U_SELSP, U_SLREG, U_SOPEN, U_SRCVNP, U_SRDBUF, U_SSELPT, U_SSTAT, U_STMON, U_SUDPTO, U_SWRBUF, U_WRAVG, U_WRBNK, U_WRCNT,

Character strings that cannot be used	• The following keywords, which are prescribed in KV script U_WRCPOS, U_WRCTC0, U_WRCTC1, U_WRDA, U_WRFBNO, U_WRFNO, U_WRFRNO, U_WRFVR, U_WRLMT, U_WRMD, U_WROFST, U_WRPOR, U_WRPPB, U_WRPPNT, U_WRPPS, U_WRPPSS, U_WRPSB, U_WRPST, U_WRPVEL, U_WRSCAM, U_WRSCL, U_WRSCOF, U_WRSCON, U_WRSPB, U_WRSVPB, U_WRSYSB, U_WRTTRQ, U_WRTVL, U_WRVTL, U_WRVVEL, V, VAL, W, WHILE, WSIZE*1, WSL, WSR, WSUM, WTIME, XOR, Z, ZF, ZONE, ZPOP, ZPUSH, ZRES, [, \\, \\A, \\B, \\F, \\N, \\R, \\T, \\V,]
	• The following keywords, which are prescribed in IEC61131-3 TRUE, FALSE, TIME, DATE,TIME_OF_DAY, TOD, DATE_AND_TIME, DT, BOOL, SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT, REAL, LREAL, STRING, WSTRING, BYTE, WORD, DWORD, LWORD, ANY, ANY_DERIVED, ANY_ELEMENTARY, ANY_MAGNITUDE, ANY_NUM, ANY_REAL, ANY_INT, ANY_BIT, ANY_STRING, ANY_DATE, ACTION, END_ACTION, ARRAY, OF, AT, CASE, END_CASE, CONFIGURATION, END_CONFIGURATION, CONSTANT, EN, ENO, EXIT, • F_EDGE, FOR, TO, BY, DO, END_FOR, FUNCTION, END_FUNCTION, FUNCTION_BLOCK, END_FUNCTION_BLOCK, IF, THEN, ELSIF, ELSE, END_IF, INITIAL_STEP, END_STEP, NOT, MOD, AND, XOR, OR, PROGRAM, WITH, END_PROGRAM, R_EDGE, READ_ONLY, READ_WRITE, REPEAT, UNTIL, END_REPEAT, RESOURCE, END_RESOURCE, RETAIN, NON_RETAIN, RETURN, STEP, STRUCT, END_STRUCT, TASK, TRANSITION, FROM, END_TRANSITION, TYPE, END_TYPE, VAR,VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR_TEMP, VAR_EXTERNAL, VAR_ACCESS, VAR_CONFIG,VAR_GLOBAL, END_VAR, WHILE, DO, END_WHILE • Other character strings (see the IEC61131-3 specifications) • A variable name cannot have two or more consecutive underscore characters. The string "a_b_c" is acceptable, but "a__bc" cannot be used.

*1 String that cannot be used only with KV-8000 with CPU function version 2.0 or later.

● Characters that cannot be used in structure member names

Characters that cannot be used	( ) * / + - < > = & ! ¥ ″ # $ % % & | ' ~ ` @ [ ] { } : ; , . ? ^ (Half-width space) (Full-width space) (__)
Characters that cannot be used at the start	0 1 2 3 4 5 6 7 8 9 _
Characters or strings that cannot be used at the end	_ _0 _1 _2 _3 _4 _5 _6 _7 _8 _9

Strings that cannot be used	ｸﾞﾛｰﾊﾞﾙ グローバル Global 全局 全域 글로벌
	• Strings that cannot be distinguished from numeric values
	  • K
	  • K + value
	  • K + value + suffix(U/S/D/L/F/DF)
	  • K + value + E
	  • K + value + E + value
	  • K + value + E + value + suffix(F/DF)
	  • H + hexadecimal value (decimal value or ABCDEF string)
	• The following key words, which are prescribed in KV script
	(For the contents of keyword, refer to "Characters that cannot be used in variable names".)
	• Keywords specified in IEC61131-3
	(For the contents of keyword, refer to "Characters that cannot be used in variable names".)

● Character that cannot be used in device comments

Character that cannot be used	,

List of Unusable Functions

The functions which cannot be used in the initialization module, function block, macro, or interrupt program are explained in this section.

● Initialization module, interrupt program

Type of function	Function	
Contact functions	LDP/LDF	Load pulse/load pulse fall
	LDPB/LDFB	Load pulse bar/load pulse fall bar
Timer counter functions	TMR/TMH/TMS/ TMU	Timer / high-speed timer / high-speed 1ms timer / high-speed 10μs timer
	C	Counter
File register functions	FRSTM/FRLDM	File register batch save/ file register batch read
Storage device functions	MWRIT/MREAD/ MFREE/MMKDIR	Write storage device / read storage device / get storage device open space / create storage device folder
	MRMDIR/MDEL	Delete storage device folder / delete storage device file
	MPRINT/MREADL/ MCOPY/MMOV/ MREN/MFREEK/ MSTAT	Write storage device text / read one line of storage device / copy storage device file / move storage device file / rename storage device file / get storage device open space / get storage device status
Access window functions	AWNUM/AWMSG AWSHOW/AWHIDE	User message 1 / user message 2 User message display/hide user message

● Function block

Type of function	Function	
Macro functions	MCALL/MSTRT	Macro call / macro start

* The FEND (function block end) function is written in the function block. It cannot be used in other types of programs such as functions or modules.

● Function

Type of function	Function	
Macro functions	MCALL/MSTRT	Macro call / macro start
Function block functions	FB/FBSTRT/ FBCALL	Function block

● Macros

Type of function	Function	
Macro functions	MCALL/MSTRT	Macro call / macro start
Function block functions	FB/FBSTRT/ FBCALL/FUN	Function block / function

* The MEND (macro end) function is written in the self-holding type macro. It cannot be used in other types of programs such as sub-routine type macros or modules.

Index

Symbol

(#) DEC constant 3-52

A

About assignment statement 1-10
Bit assignment 1-10
Numerical value assignment 1-10
Text string assignment 1-10
About Comments Function 2-69
About control statement 1-12
Conditional transfer control statements 1-12
Loop control statement 1-13
About control statements nesting 4-32
About infinite loop 4-37
About Input Auxiliary Function 2-64
About label suffix 1-16, 2-44
About label type 2-44
About Operation Type 3-71
About Parallel Connection of Box Script 2-51
About processing time 4-35
About script function 1-17
About Script Work Memory (Work Devices) 2-74
About suffix 1-14
About the bit of global label 2-45
About timer actions in a loop control statement 4-37
About warning level 2-53
Actual nesting actions 4-34
AND 4-7
Application area of work devices 2-75
Application method of indirect specifying 3-59
Area Script 2-16
Area script 1-9
Area script is used in the following cases 2-16, 2-18
Array 2-46
ASCII Code Table A-53

B

Batch register of local label 2-43
Bit assignment 1-10
Bit assignment statement 4-2
Bit Device 3-2, 3-6, 3-24
Bit status 3-21, 3-24
Box script 1-8, 2-4
BREAK Processing 4-31

C

Call function list 2-64
Call script function format hints 2-65
Calling functions 2-58
Calling self-hold type macro 2-60
Calling subroutine macro 2-60
Change priority 4-8
Characters that cannot be used in labels A-54
Characters that cannot be used in program names A-54
Characters that cannot be used in project names A-54
Combination of index modification and indirect specifying 3-62
Comment writing method 2-69
Comparison operator 4-6
Comparison operator priority 4-8
Conditional transfer control statements 1-12
Considerations when using the counter 4-40
Timer/ Counter contact 4-41
Timer/ Counter current value reading 4-40
Const 3-4, 3-8, 3-14, 3-19, 3-52
Constant 3-52
Control memory CM 3-45
Control relay (contact) 3-35
Control Relays A-10, A-18
Control Statement List 4-10
Copy area script 2-24
Copy box script 2-12
Counter (contact) 3-34
Counter (current value) C 3-43
Customizing script window 2-73

D

Data memory/Extended data memory DM/EM/FM 3-36
Data processing 3-1
Data range processed by operation formula 3-71
Data Type (suffix) 3-63
About Type Declaration (TYPE) 3-70
Device type declaration 1-16
Value change when exceeding max./min. value 3-69
Delete area script 2-23
Delete box script 2-11
Device name 3-2
Device notation method 3-21
Device number list A-2
Device table (for KV Nano) 3-17
Device table (for KV-5500 / 5000/3000/1000) 3-10
Device type designation 3-14
Differences in CPU function version 5
Differences in CPU functions according to CPU function version 5
Differences in Ethernet functions according to CPU function version 6
Restrictions for using extended input/output unit 6
Direct input (default) 2-3
Display help of the specified function on Script Edit Screen 2-67
DO ～ UNTIL 4-29
About difference between WHILE statement between DO statement 4-30

E

Error display in output window 2-55
Error Message 2-54
Error message when script conversion 2-54
Example for indirect specifying 3-61

Example for programming using box scripts 2-5
Example for programming using global label ..2-40
Example for programming with area scripts 2-17

F

Features of Area Script 2-16
Features of Box Script 2-4
Features of KV script 1-2
File register ZF/FM 3-38
Fixed text string (text string type constant) 3-54
Flow Functions 6-2, 6-4, 6-18
FM
when File Register is processed in Bank switching mode 3-39
FOR ～ NEXT (specified loop control times) ...4-24
When FOR ~ NEXT statements are imperatively ended 4-26
Precautions on Using FOR ~ NEXT statement 4-26
The range that can be specified with STEP 4-26
When specifying single precision floating point type constant with STEP 4-26
Function 1-17
About script function 1-17
Function argument error 2-54
Function block 2-58
Function block call 2-58
Function block function 6-10
Function Version .....-3, -4
Functions that cannot be executed with the current model 2-52

G

Global label 2-38
Global label array 2-47

H

HELP Function 2-66
HEX constant 3-53
High-speed counter (current value) CTH 3-43
High-speed counter comparator (contact) 3-35
High-speed counter comparator (setting value) CTC 3-44
How to Input Scripts 2-3
How to process setting/current value of the counter 3-34
How to process setting/current value of the timer 3-33
How to Program Script 2-2
How to program the counter when counter function is operated via scripts 3-34
How to use 2-58, 2-60
How to Use An Assignment Statement 4-2
Bit assignment statement 4-2
Numerical value assignment statement 4-2
Operation assignment statement 4-3
String assignment statement 4-2
When it can not be assigned to left side 4-4
When the type on right side is different from that on left side 4-4
How to Use KV Scripts 1-3
How to use output window 2-56

I

I/O relay R 3-24
IF ～ 4-12
IF ～ ELSE 4-14
IF ～ ELSE IF 4-16
Index modification 3-56
Index Modification and Indirect Specifying 3-55
Index modification programming 3-19
Index register Z 3-44
Indirect specifying principle 3-58
Indirect specifying programming 3-58
Inline Mnemonics 2-72
Internal auxiliary relay
B 3-29
LR (Latch relay) 3-31
MR 3-30
R 3-28
Internal auxiliary relay R/B/MR/LR 3-28
Internal register 3-4, 3-8, 3-19

K

KV Nano Series bit device 3-17
KV Nano series of word devices 3-18
KV-1000 bit device 3-12
KV-1000 series of word devices 3-13
KV-5500/5000/3000 bit device 3-10
KV-5500/5000/3000 word device 3-11

L

Label array 2-46
Label Programming 2-37
Ladder program equivalent to OR connection 2-51
Link register W 3-40
Link relay B 3-26
List of Characters That Cannot Be Used A-54
Character that cannot be used in device comments A-57
Characters that cannot be used in function / function block / structure names A-54
Characters that cannot be used in labels A-54
Characters that cannot be used in program names A-54
Characters that cannot be used in project names A-54
Characters that cannot be used in structure member names A-57
List of control relay control memory A-2
Local device list 3-5, 3-9, 3-20
Local device list (At KV-1000) 3-16
Local device list (When KV-5500 / 5000/3000) 3-15
Local label 2-41
Local label array 2-49

Loop control statement 1-13, 4-11
Loop control statement and processing time ... 4-35

M

Macro 2-60
Macro argument device 3-14
Macro Call 2-58, 2-59, 2-60
Macro function 6-6
Macro operand device 3-4, 3-8
Making Area Script 2-18
Making Box Script 2-6
MC ～ MCR 4-22
MC ~ MCR When using timer /
counter function 4-45
MOD 4-6
Monitor Function 2-62
Monitoring with tool tip 2-63

N

New a area script 2-19
New a box script 2-7
NOT 4-7

O

Omitting suffix 1-15
Operation assignment statement..... 4-3
Operation flag 1-18
Operation Flow 2-6
Operator 1-11
Operator List 1-11, 4-6
Arithmetic operator 4-6
Change priority 4-8
Comparison operator 4-6
Comparison operator priority 4-8
Differences between assignments
statements and compare operators 4-9
Logic operator 4-7
Logic operator priority 4-8
String Connection Operator 4-6
The priority between multiplication,
division, addition, and subtraction 4-8
Operator priority 4-7
Operator type 4-6
OR 4-7
Order of prionity low 4-6
Other precautions 2-45
Others 2-55

P

PRAGMA Directive 3-73
Precautions about the use of
timer function 3-33, 4-41
Specify timer function using Box Script 4-43
Use timer function in IF statement of
area script 4-42
Precautions about use of timer function in
Box Script and IF statement 4-42
Precautions for using the timer 4-40
Timer/ Counter contact 4-41
Timer/ Counter current value reading 4-40
Precautions on Edit Screen and
Customization 2-73
Precautions on Script Programming 4-38
Status of Hold Control Bit Device 4-38
Precautions on using assignment statement 4-4
Precautions on Using Control Statement 4-32
About timer actions in a loop
control statement 4-37
Loop control statement and
processing time 4-35
Precautions when programming script .. 2-51, 2-73
Precautions When Using Box Script 2-13
Precautions when using label 2-44
Processing Bit Device in Word Unit 3-46
Processing Word Device as Bit Device 3-47
Programming control statements with
KV Scripts 1-7
For multi-branch control 1-7
Programming example for
device using suffix 1-14
Programming flow using global label 2-38
Programming flow using local label 2-41
Programming Macro by Using Script 2-61
Programming operation processing with
KV scripts 1-3
PROGRAMMING SCRIPT 2-1, 2-2
Programming Script 2-2
Programming text string processing with
KV Script 1-5

R

Reedit area script 2-22
Reedit Box Script 2-11
Register global label 2-38
Register local label 2-41
Register monitor 2-62
Reserved word 2-66

S

Script 1-2
Script Conversion Output Message 2-52
Script function 1-10
Script function format hints 2-65
Script Function List 2-64
Script program via XYM mark. 2-77
SCRIPT SYNTAX AND PRECAUTIONS 4-1
Script type 1-8
Script warning level setting 2-53
SELECT CASE 4-18
Setting method of warning level 2-53
Simple indirect specifying #TM 3-62
Single precision floating point type constant 3-53
Specify invalid type 2-54
Specify timer function using Box Script 4-43
Start script function help 2-66
String assignment 1-10
String assignment statement 4-2
String Connection Operator 4-6

Suffix 3-63
About .B 3-68
About .D and .L 3-65
About .DF 3-67
About .F 3-66
About .T 3-68
About .U and .S 3-64
Suffix (Data Type (suffix)) 3-63
Suffix type 1-14
Syntax error 2-54

T

Temporary data memory TM 3-42
Timer (contact) T 3-32
Timer (current value) T 3-42
Timer error 3-33
TYPE 3-70
Type conversion leads to data loss 2-52
Type of indirect specifying device 3-60
Typing error 2-54

U

Use timer function in IF statement of
area script 4-42

V

Variable suffix 1-15
Variables 3-50

W

Warning Message 2-52
Watch window 2-63
What is indirect specifying(＊) 3-58
When conversion error occurs 2-55
When conversion fails 2-9, 2-21
When conversion successful 2-9, 2-20
When looking up notes on indirect
specifying and index modification 1-18
Every function operation table
when invalid address is specified 1-18
When processing 16-bit data 3-37, 3-39, 3-41
When processing 32-bit data 3-37, 3-40, 3-41
When processing as the bit device
at any bit. 3-47
When processing as the bit device
at the lowerest bit 3-47
When using box scripts in the
following cases 2-4
When using differential execution
type functions 2-13
When using KV-7000 ..... 3-2, 3-6
When using operation functions 1-4
When using operators 1-3
When using string operator 1-5
When using text string processing function 1-6
When using timer function 2-14
When WHILE statement is imperatively
terminated 4-28
When work devices are insufficient 2-75
WHILE ～ 4-27
Window input 2-3
Word device 3-3, 3-7, 3-36
Work memory (work device) 2-74
Write Easy-reading Script 2-70
Writing index modification
Invalid index modification 3-57
Modify by index register 3-56
Modify the device with suffix 3-57
Modify via index constant 3-57
Modify via the device and expression 3-57
Writing indirect specification (*) 3-21
Writing local devices 3-22

X

XOR 4-7
XYM Marking 2-77

Z

ZF
when file register is processed with consecutive
number method 3-39

Functions Index

A

ABS 7-36
ACNT 6-60
ACOS 7-148
ADRADD 6-32
ADRDEC 6-30
ADRINC 6-28
ADRSET 6-26
ADRSUB 6-34
AJST 8-76
ANDA 7-22
APR 7-68
ASC 7-154
ASIN 7-146
ASLA 7-44
ASRA 7-42
ATAN 7-150
ATAN2 7-152
AVG 8-22
AWHIDE 8-123
AWMSG 8-120
AWNUM 8-118
AWSHOW 8-122

B

BANDC 7-64
BCMP 8-36
BCMPI 8-38
BCNT 8-6
BLD 5-8
BLDB 5-10
BMOV 7-2
BOUB 5-16
BOUT 5-14
BRES 5-20
BSET 5-18
BSL 7-60
BSR 7-58
BSUM 8-30
BSWAP 7-102
BYBMOV 7-10
BYLMOV 7-12

C

CALL 6-2
CHR 7-232
CNT 5-32
COM 7-30
COS 7-142
CPMGET 7-112
CPMSET 7-110
CPSASC 7-234
CRC 8-32

D

DASC 7-158
DCNT 8-8
DEC 7-16
DECO 7-104
DEG 7-138
DELETE 7-206
DFLOAT 7-118
DI 8-78
DIC 8-80
DISB 7-96
DISF 7-126
DISN 7-92
DISS 7-228
DMX 7-86
DSER 8-12

E

ECALL 6-4
EI 8-78
ENCO 7-106
ENRA 7-28
EORA 7-26
EXP 7-130

F

FASC 7-174
FB 6-10
FBCALL 6-11
FBSTRT 6-13
FDEL 8-68
FEND 6-15
FIFOR 8-52
FIFOW 8-48
FIND 7-226
FINS 8-66
FLOAT 7-114
FMOV 7-4
FRLDM 6-40
FRSET 6-36
FRSTM 6-38
FUN 6-16
FWRIT 8-64

G

GRY 7-88

H

HASC 7-168

I

IEDGE 8-82
INC 7-14
INSERT 7-202
INSTR 7-222
INT 7-124
INTG 7-122
IS_ □□□ 6-58

L

LDF 5-4
LDFB 5-7
LDP 5-2
LDPB 5-6
LEFT 7-190
LEN 7-182
LIFOR 8-60
LIFOW 8-56
LIMIT 7-62
LLFLT 7-78
LOG 7-132
LOG10 7-134
LOGD 8-86
LOGE 8-86
LTRIM 7-210

M

MAX 8-14
MCALL 6-6
MCOPY 8-110
MDEL 8-104
MDSTOP 6-19
MDSTRT 6-18
MEND 6-8
MFREE 8-94
MFREEK 8-96
MID 7-194
MIN 8-18
MMKDIR 8-98
MMOV 8-112
MPRINT 8-106
MPX 7-84
MREAD 8-92
MREADL 8-108
MREN 8-114
MRMDIR 8-102
MSTAT 8-116
MSTRT 6-7
MWRIT 8-90

N

NEG 7-32

O

ORA 7-24

P

PIDAT 8-84
PMOV 7-6

R

RAD 7-136
RAMP 7-74
RASC 7-156
RCPSASC 7-236
RDASC 7-166
REPLACE 7-199
RES 5-13
RFASC 7-178
RFSCI 8-132
RFSCO 8-134
RFSCTH 8-79
RFSFRC 8-124
RFSX 5-36
RFSY 5-37
RGRY 7-90
RHASC 7-172
RIGHT 7-186
RLA 7-48
RLNCA 7-52
RND 8-40
ROOT 7-18
RRA 7-46
RRNCA 7-50
RSEC 8-74
RTRIM 7-212

S

SDEL 7-204
SEC 8-72
SEG 8-2
SER 8-10
SET 5-12
SFIND 7-218
SFINDN 7-220
SIN 7-140
SINS 7-200
SLA 7-40
SLEFT 7-188
SMID 7-192
SORT 8-42
SORTN 8-44
SPLIT 7-216
SPRD 8-126
SPWR 8-128
SQRT 7-20
SRA 7-38
SRGHT 7-184
SRPLC 7-196
SSVC 8-130
STR 7-162
STRIM 7-214
SWAP 7-100

T

TAN 7-144
TBCD 7-80
TBIN 7-82
TMH 5-26
TMR 5-22
TMS 5-28
TMU 5-30
TOD 7-108
TODF 7-120
TOF 7-116
TOL 7-108
TOS 7-108
TOU 7-108
TPOUT 7-76
TRGD 8-87
TRGR 8-88
TRIM 7-208

U

U_*****(Unit specific function)
KV-XLE02/EP21V
U_CSTOP *1
U_CSTRT *1
U_CREG *1
U_CERR *1
U_MSGTO *1
U_MSGSND *1
U_MSGRCV *1
U_MSGST *1
U_NDTO *1
U_NDSTAT *1
U_SBMON *1
U_SBNAME *1
U_STMON *1
U_SLREG *1
KV-XLE02/EP21V/LE21V/LE20V
U_ELCYC(KV-EP21V/LE21V) *1
U_ELSTAT(KV-EP21V/LE21V) *1
U_FCSTAT(KV-XLE02/EP21V/LE21V) *1
U_MLATTA *1
U_MLSTAT *1
U_MLSUB *1
U_MLTEXT *1
U_MLTO *1
U_MLTOGR *1
U_SOPEN *1
U_SRCVNP *1
U_SRDBUF *1
U_SSTAT *1
U_SUDPTO *1
U_SWRBUF *1
KV-XLE02/XL202/XL402
U_PLSTAT *1
U_PLCYC *1
U_PRDBUF *1
KV-XLE02
U_CCTTO *1
U_CCTSND *1
U_CCTRCV *1
U_CCTST *1
U_SSELPT *1
KV-SC20V/SSC02
U_RDCNT *1
U_RDCPT0 ..... *1
U_RDCPT1 ..... *1
U_RDHZ *1
U_RDRPM *1
U_WRCNT *1
U_WRCTC0 ..... *1
U_WRCTC1 ..... *1
U_WRPST *1
KV-SSC02
U_DISOCP *1
U_RDBF *1
KV-AD40V/KV-SAD04
U_RDAD *1
U_RDADB *1
U_WRAVG *1
U_WRLMT *1
U_WROFST *1
U_WRSCL *1
KV-SAD04
U_RDBF *1
KV-DA40V/KV-SDA04
U_WRDA *1
U_WRLMT *1
U_WROFST *1
U_WRSCL *1
KV-ML16V/MC40V/MC20V
U_WRPPNT *1
U_RDPPNT *1
U_RDPMC *1
U_RDFPOS *1
U_RDFVEL *1
U_RDFTRQ(KV-ML16V) *1
U_RDCPOS *1
U_RDCVEL *1
U_RDCNT.D *1
U_WRCPOS *1
U_WRPPS *1
U_WRPVEL *1
U_WRPOR *1
U_RDMD *1
U_WRMD(KV-ML16V) *1
U_WRVVEL(KV-ML16V) *1
U_WRVTL(KV-ML16V) *1
U_WRTTRQ(KV-ML16V) *1
U_WRTVL(KV-ML16V) *1
U_WRFRNO *1
U_WRFBNO *1
U_WRFNO *1
U_WRFVR *1
U_RDMERC *1
U_RDSERC *1
U_RDAERC *1
U_WRBNK *1
U_WRPPB *1
U_RDPPB *1
U_WRSVPB(KV-ML16V) *1
U_RDSVPB(KV-ML16V) *1
U_WRSYSB *1
U_RDSYSB *1
U_WRPSB *1
U_RDPSB *1
U_WRSPB *1
U_RDSPB *1
U_WRSCON *1
U_WRSCOF *1

U_WRSCAM *1
KV-SIR32XT
U_DISOCP *1
U_DISOVP *1
U_RDBF *1
KV-XH16ML/XH04ML/SH04PL
U_RDAERC.....*1
U_RDFPOS.....*1
U_RDFVEL.....*1
U_RDMD *1
U_RDPMC.....*1
U_RDPPNT *1
U_SELSP *1
U_WRCPOS.....*1
U_WRPOR *1
U_WRPPNT *1
U_WRPPS.....*1
U_WRPPSS *1
U_WRPVEL.....*1
KV-XH16ML/XH04ML
U_RDCPOS *1
U_RDCVEL *1
U_RDFTRQ.....*1
U_WRMD *1
U_WRTTRQ.....*1
U_WRTVL *1
U_WRVTL *1
U_WRVVEL.....*1
KV-SH04PL
U_DISOCP *1
UFILL 6-56
UFRSM 6-48
UFSUS 6-46
UMALLOC 6-50
UNIB 7-98
UNIF 7-128
UNIN 7-94
UNIS 7-230
UPSTOP 6-44
UPSTRT 6-42
UREAD 6-52
UWRIT 6-54

V

VAL 7-181

W

WSIZE 6-62
WSL 7-56
WSR 7-54
WSUM 8-26
WTIME 8-70

Z

ZONE 7-66
ZPOP 6-24
ZPUSH 6-20
ZRES 8-34

*1 See the user's manual of the corresponding unit.

*2 See the user's manual of the corresponding base unit.

Revision History

Printing Date	Version	Details of Revision
Apr 2015	Second version	
Jan 2016	Third version	
Aug 2016	Fourth version	
Sep 2017	Sixth version	
Aug 2019	First version	
May 2020	1st revision 1st version	
Jul 2021	3rd revision 1st version	
Sep 2021	4th revision 1st version	

WARRANTIES AND DISCLAIMERS

(1) KEYENCE warrants the Products to be free of defects in materials and workmanship for a period of one (1) year from the date of shipment. If any models or samples were shown to Buyer, such models or samples were used merely to illustrate the general type and quality of the Products and not to represent that the Products would necessarily conform to said models or samples. Any Products found to be defective must be shipped to KEYENCE with all shipping costs paid by Buyer or offered to KEYENCE for inspection and examination. Upon examination by KEYENCE, KEYENCE, at its sole option, will refund the purchase price of, or repair or replace at no charge any Products found to be defective. This warranty does not apply to any defects resulting from any action of Buyer, including but not limited to improper installation, improper interfacing, improper repair, unauthorized modification, misapplication and mishandling, such as exposure to excessive current, heat, coldness, moisture, vibration or outdoors air. Components which wear are not warranted.

(2) KEYENCE is pleased to offer suggestions on the use of its various Products. They are only suggestions, and it is Buyer's responsibility to ascertain the fitness of the Products for Buyer’s intended use. KEYENCE will not be responsible for any damages that may result from the use of the Products.

(3) The Products and any samples ("Products/Samples") supplied to Buyer are not to be used internally in humans, for human transportation, as safety devices or fail-safe systems, unless their written specifications state otherwise. Should any Products/Samples be used in such a manner or misused in any way, KEYENCE assumes no responsibility, and additionally Buyer will indemnify KEYENCE and hold KEYENCE harmless from any liability or damage whatsoever arising out of any misuse of the Products/Samples.

(4) **OTHER THAN AS STATED HEREIN, THE PRODUCTS/SAMPLES ARE PROVIDED WITH NO OTHER WARRANTIES WHATSOEVER. ALL EXPRESS, IMPLIED, AND STATUTORY WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF PROPRIETARY RIGHTS, ARE EXPRESSLY DISCLAIMED. IN NO EVENT SHALL KEYENCE AND ITS AFFILIATED ENTITIES BE LIABLE TO ANY PERSON OR ENTITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, PUNITIVE, SPECIAL OR CONSEQUENTIAL DAMAGES (INCLUDING, WITHOUT LIMITATION, ANY DAMAGES RESULTING FROM LOSS OF USE, BUSINESS INTERRUPTION, LOSS OF INFORMATION, LOSS OR INACCURACY OF DATA, LOSS OF PROFITS, LOSS OF SAVINGS, THE COST OF PROCUREMENT OF SUBSTITUTED GOODS, SERVICES OR TECHNOLOGIES, OR FOR ANY MATTER ARISING OUT OF OR IN CONNECTION WITH THE USE OR INABILITY TO USE THE PRODUCTS, EVEN IF KEYENCE OR ONE OF ITS AFFILIATED ENTITIES WAS ADVISED OF A POSSIBLE THIRD PARTY’S CLAIM FOR DAMAGES OR ANY OTHER CLAIM AGAINST BUYER.** In some jurisdictions, some of the foregoing warranty disclaimers or damage limitations may not apply.

BUYER'S TRANSFER OBLIGATIONS:

If the Products/Samples purchased by Buyer are to be resold or delivered to a third party, Buyer must provide such third party with a copy of this document, all specifications, manuals, catalogs, leaflets and written information provided to Buyer pertaining to the Products/Samples.

Specifications are subject to change without notice.

KEYENCE CORPORATION

www.keyence.com/glb

1-3-14, Higashi-Nakajima, Higashi-Yodogawa-ku, Osaka, 533-8555, Japan PHONE: +81-6-6379-2211

AUSTRIA
Phone: +43 (0)2236 378266 0

BELGIUM
Phone: +32 (0)15 281 222

BRAZIL
Phone: +55-11-3045-4011

CANADA
Phone: +1-905-366-7655

CHINA
Phone: +86-21-3357-1001

CZECH REPUBLIC
Phone: +420 220 184 700

FRANCE
Phone: +33 1 56 37 78 00

GERMANY
Phone: +49-6102-3689-0

HONG KONG
Phone: +852-3104-1010

HUNGARY
Phone: +36 1 802 7360

INDIA
Phone: +91-44-4963-0900

INDONESIA
Phone: +62-21-2966-0120

ITALY
Phone: +39-02-6688220

KOREA
Phone: +82-31-789-4300

MALAYSIA
Phone: +60-3-7883-2211

MEXICO
Phone: +52-55-8850-0100

NETHERLANDS
Phone: +31 (0)40 206 6100

PHILIPPINES
Phone: +63-(0)2-8981-5000

POLAND
Phone: +48 71 368 61 60

ROMANIA
Phone: +40 (0)269 232 808

SINGAPORE
Phone: +65-6392-1011

SLOVAKIA
Phone: +421 (0)2 5939 6461

SLOVENIA
Phone: +386 (0)1 4701 666

SWITZERLAND
Phone: +41 (0)43 455 77 30

TAIWAN
Phone:+886-2-2721-8080

THAILAND
Phone: +66-2-369-2777

UK & IRELAND
Phone: +44 (0)1908-696-900

USA
Phone: +1-201-930-0100

VIETNAM
Phone: +84-24-3772-5555

B5WW1-MAN-2090

Copyright (c) 2021 KEYENCE CORPORATION. All rights reserved. 084659GB 2091-1 B89GB Printed in Japan

B89GB-1