MITSUBISHI ELECTRIC

TECHNICAL BULLETIN

FA-A-0001-N

Method of replacing High Performance model QCPU with Universal model QCPU

■Date of Issue

January 2008 (Ver. N: March 2019)

■Relevant Models

Q02CPU, Q02HCPU, Q06HCPU, Q12HCPU, Q25HCPU, Q02UCPU, Q03UDCPU, Q03UDVCPU, Q03UDECPU, Q04UDHCPU, Q04UDVCPU, Q04UDEHCPU, Q06UDHCPU, Q06UDVCPU, Q06UDEHCPU, Q10UDHCPU, Q10UDEHCPU, Q13UDHCPU, Q13UDVCPU, Q13UDEHCPU, Q20UDHCPU, Q20UDEHCPU, Q26UDHCPU, Q26UDVCPU, Q26UDEHCPU, Q50UDEHCPU, Q100UDEHCPU

Thank you for your continued support of Mitsubishi Electric programmable controllers, MELSEC-Q series.

This bulletin provides detailed information on how to replace the High Performance model QCPU with the Universal model QCPU.

When considering the replacement, refer to the technical bulletin "Method of replacing High Performance model QCPU with Universal model QCPU (Introduction) (FA-A-0209)" before read this bulletin and check the products and functions required to be replaced.

In addition, for the method of replacing the Basic model QCPU with the Universal model QCPU, refer to the latest version of the technical bulletin "FA-A-0054".

When replacing the High Performance model QCPU with the Universal model QCPU, products and functions not described in this technical bulletin are not especially restricted.

Note that the reference manuals or the references described in this bulletin are information as of March 2019.

CONTENTS

1 GENERIC TERMS

Unless otherwise specified, this technical bulletin uses the following terms.

Generic term	Description
High Performance model QCPU	A generic term for the Q02CPU, Q02HCPU, Q06HCPU, Q12HCPU, and Q25HCPU
Universal model QCPU	A generic term for the Q02UCPU, Q03UDCPU, Q03UDVCPU, Q03UDECPU, Q04UDHCPU, Q04UDVCPU, Q04UDEHCPU, Q06UDHCPU, Q06UDVCPU, Q06UDEHCPU, Q10UDHCPU, Q10UDEHCPU, Q13UDHCPU, Q13UDVCPU, Q13UDEHCPU, Q20UDHCPU, Q20UDEHCPU, Q26UDHCPU, Q26UDVCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU
Built-in Ethernet port QCPU	A generic term for the Q03UDVCPU, Q03UDECPU, Q04UDVCPU, Q04UDEHCPU, Q06UDVCPU, Q06UDEHCPU, Q10UDEHCPU, Q13UDVCPU, Q13UDEHCPU, Q20UDEHCPU, Q26UDVCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU
High-speed Universal model QCPU	A generic term for the Q03UDVCPU, Q04UDVCPU, Q06UDVCPU, Q13UDVCPU, and Q26UDVCPU
QnUD(H)CPU	A generic term for the Q03UDCPU, Q04UDHCPU, Q06UDHCPU, Q10UDHCPU, Q13UDHCPU, Q20UDHCPU, and Q26UDHCPU
QnUDE(H)CPU	A generic term for the Q03UDECPU, Q04UDEHCPU, Q06UDEHCPU, Q10UDEHCPU, Q13UDEHCPU, Q20UDEHCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU
QnUD(E)(H)CPU	A generic term for the Q03UDCPU, Q03UDECPU, Q04UDHCPU, Q04UDEHCPU, Q06UDHCPU, Q06UDEHCPU, Q10UDHCPU, Q10UDEHCPU, Q13UDHCPU, Q13UDEHCPU, Q20UDHCPU, Q20UDEHCPU, Q26UDHCPU, Q26UDEHCPU, Q50UDEHCPU, and Q100UDEHCPU
QnUDVCPU	A generic term for the Q03UDVCPU, Q04UDVCPU, Q06UDVCPU, Q13UDVCPU, and Q26UDVCPU

2 PRECAUTIONS FOR REPLACEMENT

This chapter describes the precautions for replacing the High Performance model QCPU with the Universal model QCPU and the replacement methods.

System configuration

■Precautions and replacement methods

No.	Item	Precaution	Replacement method	Reference
1	Use of AnS/A series module	The Universal model QCPU whose serial number (first five digits) is "13102" or later must be used. Since the Universal model QCPU whose serial number (first five digits) is "13101" or earlier cannot be mounted with AnS/A series modules, consider configuring a system using Q series modules.	—	—
2	GOT	GOT900 series cannot be connected.	Use GOT1000 or GOT2000 series.	—
3	Programming tool connection	Applicable USB cables are different. • High Performance model QCPU: A-B type • Universal model QCPU: A-miniB type	Use USB cables of A-miniB type. Or, use USB conversion adapters of B-miniB type.	—
4	Applicable products and software	Products and software compatible with the Universal model QCPU must be used.	Check products need to be replaced for the compatibility with the Universal model QCPU and software need to be upgraded for the communication with the Universal model QCPU.	• Page 11 Products needed to be replaced for the compatibility with the Universal model QCPU • Page 13 CPU modules that can configure a multiple CPU system with the Universal model QCPU
5	Multiple CPU system	To configure a multiple CPU system, CPU modules compatible with the Universal model QCPU must be used.	Check CPU modules compatible with the Universal model QCPU.	Page 13 CPU modules that can configure a multiple CPU system with the Universal model QCPU
		In a multiple CPU system using the Motion CPU, an existing auto refresh area and user setting area cannot be used for data communication with the Motion CPU.	For data communication with the Motion CPU, use an auto refresh area and user setting area in the multiple CPU high-speed transmission area.	Chapter 4 in the QCPU User's Manual (Multiple CPU System)
6	Redundant power supply system	In a redundant power supply system, to check the status of the power supply module using SM1780 to SM1783/SD1780 to SD1783 or the system monitor window, the Universal model QCPU whose serial number (first five digits) is "10042" or later must be used. If the Universal model QCPU whose serial number (first five digits) is "10041" or earlier is used, check the status of the power supply module by the LED on the front of the module. (The status of the power supply module in the redundant power supply system cannot be stored in SM1780 to SM1783/SD1780 to SD1783 nor cannot be displayed in the system monitor window.)	—	Section 7.1 in the QCPU User's Manual (Hardware Design, Maintenance and Inspection)
7	MELSECNET/H	There is no special relay for the simple dual-structured network.	When the simple dual-structured network is used, modify programs and parameters.	• Section 7.7 in the Q Corresponding MELSECNET/H Network System Reference Manual (PLC to PLC network) • Page 56 Alternative Methods for the Simple Dual-structured Network of MELSECNET/H

No.	Item	Precaution	Replacement method	Reference
8	MELSECNET/H, CC-Link IE Controller Network	Interlink transmission timing differs.	Add a handshake program to the send side and receive side so that the module does not receive data while sending data.	Section 6.2 in the Q Corresponding MELSECNET/H Network System Reference Manual (PLC to PLC network) Section 4.1 in the MELSEC-Q CC-Link IE Controller Network Reference Manual

Program

■Precautions and replacement methods

No.	Item	Precaution	Replacement method	Reference
1	Language and instruction	Some instructions are not supported.	When the instructions not supported in the Universal model QCPU are used, replace them with alternative methods.	Page 15 Instructions not Supported in the Universal Model QCPU and Replacing Methods
2	Floating-point operation	The Universal model QCPU performs program operations of floating-point data in single-precision.	Instructions for floating-point double-precision operation are added for the Universal model QCPU. When floating-point double-precision operations are required, replace the instructions with double-precision floating-point operation instructions.	• Appendix 4.4 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals) • Page 30 Floating-point Operation Instructions
		When using the floating-point data comparison instructions, LDE□, ANDE□, ORE□, LDED□, ANDED□, and ORED□, if the comparison source data are -0, nonnumeric, unnormalized number, or ±∞, "OPERATION ERROR" (error code: 4101) is detected.*2 (□ indicates one of the following: =, <>, <=, >=, <, >)	When the floating-point data comparison instructions are used, modify the program.	Page 37 Error Check Processing for Floating-point Data Comparison Instructions (excluding High-speed Universal model QCPU)
3	Device range check at index modification	When a device number exceeds a setting range due to index modification, "OPERATION ERROR" (error code: 4101) is detected.	Deselect the "Check device range at indexing" checkbox in the PLC RAS tab of the PLC parameter dialog box so that checking is not performed.	• Section 3.17 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals) • Page 41 Range Check Processing for Index-modified Devices
4	Program execution type	Low-speed execution type programs are not supported.	Use scan execution type programs or fixed scan execution type programs.	Section 2.10 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)
		A program execution type cannot be changed by remote operation. However, in the QnUDVCPU and QnUDPVCPU whose serial number (first five digits) is "18112" or later, the program execution type can be changed by remote operation when the program execution type is the scan execution type or the stand-by type.	Use instructions for switching program execution types, such as PSTOP, POFF, and PSCAN.	Section 2.10.5 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)
5	Latch setting	If latch ranges of internal user devices are specified, the processing time is added in proportion to the device points set to be latched. (For example, if 8K points are latched for the latch relay (L) with the QnUD(E)(H)CPU, the processing time is 28.6μs.)	The latch function of the Universal model QCPU is enhanced. • Large-capacity file register (R, ZR) • Writing/reading device data to/from the standard ROM (SP.DEVST/S(P).DEVLD instructions) • Latch range specification of internal devices • "Time Setting" specification in the latch interval setting parameter*3 Change the latch method to the one described above according to the application.	• Section 3.3 and 3.3 (5) (b) in the QnUCPU User's Manual (Function Explanation, Program Fundamentals) • Page 45 Device Latch Function
6	Interrupt program	The interrupt pointer (I49) for the high-speed interrupt function is not supported.*2	Consider the use of interrupt pointers for fixed scan interrupt (I28 to I31).	Section 3.13.2 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)
		Interrupt counter is not supported.	Check the number of interrupt program executions on the Interrupt program monitor list window.	
		The interrupt pointer (I32 to I40) for an error is not supported.	—	Section 4.11 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)

No.	Item	Precaution	Replacement method	Reference
7	SCJ instruction	When the SCJ instruction is used in the Universal model QCPU, the AND SM400 (or NOP instruction) needs to be inserted immediately before the SCJ instruction.*2	Insert the AND SM400 (or NOP instruction) immediately before the SCJ instruction when the SCJ instruction is used.	Section 6.5 in the MELSEC-Q/L Programming Manual (Common Instruction)
8	ZPUSH instruction	The number of index registers is increased to 20 for the Universal model QCPU. The area for saving the data in the index register with the ZPUSH instruction is increased as well.	Increase the save areas used for the ZPUSH instruction as needed.	Section 7.19 in the MELSEC-Q/L Programming Manual (Common Instruction)
9	File usability setting for each program	The following file usability setting for each program is not available.*1 • File register • Device initial value • Comment	When file usability is set, modify the program.	• Section 2.10 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals) • Page 48 File Usability Setting
10	I/O refresh setting for each program	I/O refresh setting for each program is not available.	Use the RFS instruction if I/O refresh setting for each program is required.	MELSEC-Q/L Programming Manual (Common Instruction)
11	SM/SD	Usage of a part of the special relay and special register is different.	Replace the corresponding special relay and special register using alternative methods.	• Page 62 Special Relay List • Page 64 Special Register List
		To use the A series-compatible special relay/ special register (SM1000 to SM1255/SD1000 to SD1255), the Universal model QCPU whose serial number (first five digits) is "10102" or later must be used. If the one whose serial number (first five digits) is "10101" or earlier is used, replace the special relay/special register with that for the Universal model QCPU using the conversion function of a programming tool. Note, however, that the ones which are not compatible with the Universal model QCPU are replaced with SM1255/ SD1255, modify programs as necessary.	—	QCPU User's Manual (Hardware Design, Maintenance and Inspection)
12	Processing time	Scan time and other processing times are different.	Modify programs as needed, checking the processing timing.	—
13	Number of steps	The number of steps increases by one when:*4 • Index modification is performed. • A leading or trailing edge instruction is used. • Bit devices are used as word data by specifying digits using K1, K2, K3, K5, K6, or K7, or by specifying a device number of other than multiples of 16.	If index modifications mentioned on the left are frequently used in the program, the program size may exceed the storage capacity of the replaced CPU module. After the program controller type is changed, check the program size using the confirm memory size function. If the program size exceeds the storage capacity, take the following actions or change the CPU module to that with larger program memory. • Move parameters and device comments to the standard ROM. • Reduce the reserved area for online change. • Use the file register, extended data register, and extended link register within 64K words because the number of steps decreases by one when used in that way.	MELSEC-Q/L Programming Manual (Common Instruction)

*1 The local device file usability setting is also not available for the Universal model QCPU if the serial number (first five digits) is "10011" or earlier.

*2 This will not apply when the High Performance model QCPU is replaced with the High-speed Universal model QCPU.

*3 Only the High-speed Universal model QCPU supports this setting.

*4 This will apply only when the High Performance model QCPU is replaced with the High-speed Universal model QCPU.

Drives and files

■Precautions and replacement methods

No.	Item	Precaution	Replacement method	Reference
1	Boot file setting	Files in the standard ROM cannot be booted to the program memory.	Since the Universal model QCPU holds the data in the program memory even when the battery voltage drops, the boot file setting is not necessary. Move files with the boot setting (from the standard ROM to the program memory) to the program memory.	• Section 2.11 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals) • Page 50 Parameter-valid Drive and Boot File Setting
		Booting operation is different.	When the parameter-valid drive and the boot file setting are set in the High Performance model QCPU, change the setting.	
		A memory card (SRAM card, ATA card, or Flash card) cannot be specified as a transfer source.*1	Specify an SD memory card as a transfer source.	
2	Automatic all data write from memory card to standard ROM	The setting method of this function is different.	In the Boot file tab of the PLC parameter dialog box, select "standard ROM" for the transfer destination. Note, however, that the transfer destination of "program" is fixed to "program memory". (Setting by DIP switches is not necessary.)	Section 2.11 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)
3	Device comment	A device comment file cannot be stored in an SRAM card.*1	Store the file in the standard RAM.	—
		A device comment file cannot be stored in an ATA card nor Flash card.*1	Store the file in an SD memory card.	—
4	Initial device value	An initial device value file cannot be stored in an SRAM card.*1	Store the file in the standard RAM or standard ROM.	Section 3.25 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)
		An initial device value file cannot be stored in an ATA card nor Flash card.*1	Store the file in an SD memory card.	
5	Local device	A local device file cannot be stored in an SRAM card.*1	• Store the file in the standard RAM. • If the size of the local device file exceeds the standard RAM capacity, consider the use of an extended SRAM cassette.	Section 6.2 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)
6	File register	A file register file cannot be stored in an SRAM card.*1	• Store the file in the standard RAM. • If the size of the file register file exceeds the standard RAM capacity, consider the use of an extended SRAM cassette.	Section 4.7.1 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)
		A file register file cannot be stored in a Flash card. (Sequence programs only can read file register data in a Flash card.)*1	Use the initial device value file in an SD memory card or the FREAD/FWRITE instructions.	• Section 3.25 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals) • MELSEC-Q/L Programming Manual (Common Instruction)
7	Sampling trace	A sampling trace file cannot be stored in an SRAM card.*1	• Store the file in the standard RAM. • If the size of the sampling trace file exceeds the standard RAM capacity, consider the use of an extended SRAM cassette.	Section 3.14 (2) in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)
8	CPU module change function with memory card	A memory card cannot be specified as a backup destination or restoration source.*1	Specify an SD memory card as a backup destination or restoration source.	Section 3.31 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)

*1 This applies when the High Performance model QCPU is replaced with the High-speed Universal model QCPU.

External communication

■Precautions and replacement methods

No.	Item	Precaution	Replacement method	Reference
1	Module service interval time read	The module service interval time cannot be read.	—	Section 3.24.1 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)
2	MC protocol	To access the CPU module using A-compatible 1C frame or A-compatible 1E frame, the Universal model QCPU whose serial number (first five digits) is "10102" or later must be used. If the one whose serial number is "10101" or earlier is used, use the following frame types. • QnA-compatible 2C/3C/4C frame • QnA-compatible 3E frame • 4E frame	—	MELSEC Communication Protocol Reference Manual
		The following commands cannot specify monitoring conditions. • Randomly reading data in units of word (Command: 0403) • Device memory monitoring (Command: 0801) The applicable frame types are as follows: • QnA-compatible 3C/4C frame • QnA-compatible 3E frame • 4E frame	—	

Diagnostic function

■Precautions and replacement methods

No.	Item	Precaution	Replacement method	Reference
1	Error history	Error history data cannot be stored in the memory card.	The Universal model QCPU stores all storable data (up to 100) in the built-in memory.	Section 3.18 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)
2	LED indication priority setting	LED indication priority cannot be set. Only LED indication setting at error occurrence is supported.	—	Section 3.20.2 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)

Debugging

■Precautions and replacement methods

No.	Item	Precaution	Replacement method	Reference
1	Monitor condition setting	To use the monitor condition setting function, the Universal model QCPU whose serial number (first five digits) is "10042" or later must be used. If the one whose serial number is "10041" or earlier is used, check device data under the specified monitoring condition using the sampling trace function.	—	Section 3.11.1 and 3.14 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)
2	Scan time measurement	To use the scan time measurement function, the Universal model QCPU whose serial number (first five digits) is "10042" or later must be used.*1 If the one whose serial number is "10041" or earlier is used, calculate the time using instruction processing time described in the manual.	—	• Section 3.13.3 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals) • Appendix 1 in the MELSEC-Q/L Programming Manual (Common Instruction)
3	External input/output forced on/off	To use the external input/output forced on/off function, the Universal model QCPU whose serial number (first five digits) is "10042" or later must be used.*2 If the one whose serial number is "10041" or earlier is used, the function can be replaced with alternative programs described in Section 4.7. Note, however, that replacement method described does not apply in the following cases: • Input and output targeted for forced on/off are referred to or changed using the direct input device (DX) and direct output device (DY). • Input and output targeted for forced on/off are referred to or changed within an interrupt program.	—	• Section 3.11.3 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals) • Page 53 External Input/ Output Forced On/Off Function

*1 Scan time of each program can be checked on the Program monitor list window.
*2 Device test can be performed with the CPU module (Q02UCPU, Q03UDCPU, Q04UDHCPU, Q06UDHCPU, Q13UDHCPU, Q26UDHCPU) whose serial number (first five digits) is "10041" or earlier.

Switch on the front of the CPU module

■Precautions and replacement methods

No.	Item	Precaution	Replacement method	Reference
1	Switch on the front of the CPU module	The operation method with the RESET/RUN/ STOP switch is modified.	The RESET/STOP/RUN switch of the Universal model QCPU can be used for the reset operation of the CPU module and switching between STOP and RUN status.	Section 6.1.3 in the QCPU User's Manual (Hardware Design, Maintenance and Inspection)
		Latch data cannot be cleared by the switch.	To clear latch data, perform a remote latch clear operation.	Section 2.7 (4) and 3.6.4 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)
		The system protect cannot be set by the switch.	Data in the files can be protected by setting a password for each file. Password can be registered using a programming tool.	Section 3.19 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals)
		The parameter-valid drive setting is not necessary.	The Universal model QCPU automatically determines the parameter-valid drive. When the parameter-valid drive is set to other than the program memory in the High Performance model QCPU, change the setting.	• Section 2.1.2 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals) • Page 50 Parameter-valid Drive and Boot File Setting

SFC

■Precautions and replacement methods

No.	Item	Precaution	Replacement method	Reference
1	Step transition monitoring timer	The step transition monitoring timer is not supported.	Change the program as described in Appendix 3 in the MELSEC-Q/L/QnA Programming Manual (SFC).	Section 4.6 and Appendix 3 in the MELSEC-Q/L/QnA Programming Manual (SFC)
2	SFC operation mode setting	The periodic execution block setting is not supported.	Change the program as described in Appendix 3 in the MELSEC-Q/L/QnA Programming Manual (SFC).	Section 4.7 and Appendix 3 in the MELSEC-Q/L/QnA Programming Manual (SFC)
		To select an operation mode at double block START, the Universal model QCPU whose serial number (first five digits) is "12052" or later must be used. If the Universal model QCPU whose serial number (first five digits) is "12051" or earlier is used, the operation mode at double block START is fixed to "WAIT".	—	Section 4.7 in the MELSEC-Q/L/QnA Programming Manual (SFC)
		An operation mode at transition to active step cannot be selected. (Fixed to "TRANSFER".)	Consider to execute an SFC program with the operation mode at transition to active step "TRANSFER" (Operation mode at double step START).	Section 4.7 in the MELSEC-Q/L/QnA Programming Manual (SFC)
3	SFC program for program execution management	SFC programs for program execution management are not supported.	Consider to execute a program with one normal SFC program.	Section 5.3 in the MELSEC-Q/L/QnA Programming Manual (SFC)
4	SFC control instruction	Some SFC control instructions are not supported.	—	• Section 4.4 in the MELSEC-Q/L/QnA Programming Manual (SFC) • Page 16 SFC control instructions not supported in the Universal model QCPU and alternative methods
5	SFC comment readout instruction	To execute the following SFC comment readout instructions, the Universal model QCPU whose serial number (first five digits) is "12052" or later must be used. • S(P).SFCSCOMR (SFC step comment readout instruction) • S(P).SFCTCOMR (SFC transition condition comment readout instruction)	—	Section 4.8 in the MELSEC-Q/L/QnA Programming Manual (SFC)
6	Method of SFC program change	SFC program files cannot be written to the running CPU module. (Programs in SFC Figure can be changed online.)	• Write program data to the CPU module after changing the Universal model QCPU status to STOP. • An inactive block in an SFC program can be changed by online change of inactive block.*1	Section 6.6 in the MELSEC-Q/L/QnA Programming Manual (SFC)

*1 This operation is available for the Universal model QCPU other than the Q02UCPU and whose serial number (first five digits) is "12052" or later.

3 APPLICABLE PRODUCTS AND SOFTWARE

Products needed to be replaced for the compatibility with the Universal model QCPU

The following tables show products needed to be replaced for the compatibility with the Universal model QCPU. (As for devices not listed in the tables below, replacement is not required.)

■Products needed to be replaced (Communication modules)

Product	Model	Serial number (first five digits) of the product compatible with the Universal model QCPU*2			
		Used with Q02U/ Q03UD/Q04UDH/ Q06UDHCPU	Used with Q13UDH/ Q26UDHCPU	Used with Q10UDH/ Q20UDHCPU, or QnUDE(H)CPU	Used with High-speed Universal model QCPU
Web server module*1	• QJ71WS96	"09042" or later	"10011" or later	"10012" or later	"14122" or later
MES interface module	• QJ71MES96				
High speed data logger module	• QD81DL96	No restrictions	No restrictions	No restrictions	"14122" or later

*1 The Universal model QCPU does not operate normally when the Web server module on which GX RemoteService-I or MX MESInterface-WS Version 1 are installed is used.

*2 The Universal model QCPU does not operate normally when an incompatible module version is used.

■Products needed to be replaced (PC interface boards)

Product		Model	Dedicated software package version compatible with the Universal model QCPU*1			
			Used with Q02U/ Q03UD/Q04UDH/ Q06UDHCPU	Used with Q13UDH/ Q26UDHCPU	Used with Q10UDH/ Q20UDHCPU, or QnUDE(H)CPU	Used with High-speed Universal model QCPU
CC-Link IE Field Network interface board		• Q81BD-J71GF11-T2	No restrictions	No restrictions	No restrictions	1.03D or later
CC-Link IE Controller Network interface board		• Q81BD-J71GP21-SX • Q81BD-J71GP21S-SX • Q80BD-J71GP21-SX • Q80BD-J71GP21S-SX	No restrictions	1.03D or later	1.06G or later	1.15R or later
MELSECNET/H interface board	SI/QSI/H-PCF optical cable	• Q80BD-J71LP21-25 • Q80BD-J71LP21S-25	15R or later	18U or later	20W or later	25B or later
		• Q81BD-J71LP21-25	19V or later	19V or later		
	GI optical cable	• Q80BD-J71LP21G	15R or later	18U or later		
	Coaxial cable	• Q80BD-J71BR11				
CC-Link system master/local interface board		• Q80BD-J61BT11N	1.02C or later	1.05F or later	1.07H or later	1.12N or later
		• Q81BD-J61BT11	1.06G or later	1.06G or later		

*1 No restrictions on the board itself. For the latest dedicated software package, please consult your local Mitsubishi representative.

■Products needed to be replaced (GOT)

Product	Model	GT Designer2 OS version compatible with the Universal model QCPU*1					GT Works3 OS version compatible with the Universal model QCPU*1
		Used with Q02U/Q03UD/ Q04UDH/ Q06UDHCPU	Used with Q13UDH/ Q26UDHCPU	Used with Q10UDH/ Q20UDHCPU	Used with Q03UDE/ Q04UDEH/ Q06UDEH/ Q13UDEH/ Q26UDEHCPU	Used with Q10UDEH/ Q20UDEHCPU	Used with High-speed Universal model QCPU
GOT1000	GT16□-□	No restrictions	No restrictions	2.91V or later	No restrictions	2.91V or later	1.64S or later
	GT15□-□	2.60N or later	2.76E or later	2.91V or later	2.81K or later	2.91V or later	1.64S or later
	GT14□-□	No restrictions	No restrictions	No restrictions	No restrictions	No restrictions	1.64S or later
	GT12□-□	No restrictions	No restrictions	No restrictions	No restrictions	No restrictions	1.67V or later
	GT11□-□	2.60N or later	2.76E or later	2.91V or later	2.81K or later	2.91V or later	1.64S or later
	GT10□-□	2.76E or later	2.76E or later	2.91V or later	2.81K or later	2.91V or later	1.64S or later

*1 No restrictions on GOT itself. For the latest GT Designer2 or GT Works3, please consult your local Mitsubishi representative.

■Products needed to be replaced (Network modules and serial communication modules)

Product	Model	Module version compatible with the Universal model QCPU		
		Used with Q02U/Q03UD/ Q04UDH/Q06UDH/ Q10UDH/Q13UDH/ Q20UDH/Q26UDHCPU	Used with QnUDE(H)CPU	Used with High-speed Universal model QCPU
MELSECNET/H module	• QJ71LP21-25 • QJ71LP21S-25 • QJ71LP21G • QJ71LP21GE • QJ71BR11	No restrictions	Some restrictions depending on use conditions*1	
Serial communication module	• QJ71C24N • QJ71C24N-R2 • QJ71C24N-R4		Serial number (first five digits) "10042" or later	No restrictions

*1 The serial number (first five digits) of the MELSECNET/H module must be "10042" or later if all conditions described below are satisfied.
·A multiple CPU system including Built-in Ethernet port QCPU is configured.
·A programming tool or GOT is connected to an Ethernet port of Built-in Ethernet port QCPU.
·A programming tool or GOT accesses the CPU module on another station via the MELSECNET/H module controlled by another CPU.
·The access target on another station is A/QnA series CPU module.

CPU modules that can configure a multiple CPU system with the Universal model QCPU

CPU modules that can configure a multiple CPU system with the Universal model QCPU are shown below.

■For the QnUD(H)CPU or Built-in Ethernet port QCPU

• CPU modules that can configure a multiple CPU system with the QnUD(H)CPU or Built-in Ethernet port QCPU

CPU module	Model	Applicable version				Restrictions
		Configured with Q03UD/ Q04UDH/ Q06UDHCPU	Configured with Q13UDH/ Q26UDH/ Q03UDE/ Q04UDEH/ Q06UDEH/ Q13UDEH/ Q26UDEHCPU	Configured with Q10UDH/ Q20UDH/ Q10UDEH/ Q20UDEHCPU	Used with High-speed Universal model QCPU	
Motion CPU	• Q172DCPU • Q173DCPU • Q172DCPU-S1 • Q173DCPU-S1 • Q172DSCPU • Q173DSCPU	No restrictions				Use only the multiple CPU high-speed main base unit (Q3□DB) as a main base unit.
PC CPU module	• PPC-CPU852(MS)	Driver S/W (PPC-DRV-02) version 1.01 or later	Driver S/W (PPC-DRV-02) version 1.02 or later	Driver S/W (PPC-DRV-02) version 1.03 or later	N/A	—
C Controller module	• Q06CCPU-V • Q06CCPU-V-B	No restrictions	Serial number (first five digits) "10012" or later	Serial number (first five digits) "10102" or later	N/A	—
	• Q12DCCPU-V • Q24DHCCPU-V	No restrictions			Serial number (first five digits) "14122" or later	—
High Performance model QCPU	• Q02CPU • Q02HCPU • Q06HCPU • Q12HCPU • Q25HCPU	Function version B or later				—
Process CPU	• Q02PHCPU • Q06PHCPU • Q12PHCPU • Q25PHCPU	No restrictions				—

■For the Q02UCPU

• CPU modules that can configure a multiple CPU system with Q02UCPU

CPU module	Model	Applicable version	Restrictions
Motion CPU	• Q172CPUN(-T) • Q173CPUN(-T) • Q172HCPU(-T) • Q173HCPU(-T)	No restrictions	The multiple CPU high-speed main base unit (Q3□DB) cannot be used as a main base unit.
PC CPU module	• PPC-CPU852(MS)	Driver S/W (PPC-DRV-02) version 1.01 or later	—
C Controller module	• Q06CCPU-V • Q06CCPU-V-B	No restrictions	—
	• Q12DCCPU-V • Q24DHCCPU-V	No restrictions	—

Software needed to be upgraded for the compatibility with the Universal model QCPU

The following table shows software needed to be upgraded for the communication with the Universal model QCPU. (As for software not listed in the table below, version upgrade is not required.)

■Software needs to be upgraded

Software	Model	Version compatible with the Universal model QCPU				
		Used with Q02U/Q03UD/ Q04UDH/ Q06UDHCPU	Used with Q13UDH/ Q26UDHCPU	Used with Q03UDE/ Q04UDEH/ Q06UDEH, Q13UDEH/ Q26UDEHCPU	Used with Q10UDH/ Q20UDH/ Q10UDEH/ Q20UDEHCPU	Used with High-speed Universal model QCPU
GX Works2	• SW1DND-GXW2-E • SW1DNC-GXW2-E	No restrictions				1.98C or later
GX Developer	SW8D5C-GPPW-E	8.48A or later	8.62Q or later	8.68W or later	8.78G or later	N/A
GX Configurator-AD	SW2D5C-QADU-E	2.05F or later*1	2.05F or later*2	2.05F or later*3	2.05F or later*4	N/A
GX Configurator-DA	SW2D5C-QDAU-E	2.06G or later*1	2.06G or later*2	2.06G or later*3	2.06G or later*4	N/A
GX Configurator-SC	SW2D5C-QSCU-E	2.12N or later*1	2.12N or later*2	2.17T or later*3	2.17T or later*4	N/A
GX Configurator-CT	SW0D5C-QCTU-E	1.25AB or later*1	1.25AB or later*2	1.25AB or later*3	1.25AB or later*4	N/A
GX Configurator-TI	SW1D5C-QTIU-E	1.24AA or later*1	1.24AA or later*2	1.24AA or later*3	1.24AA or later*4	N/A
GX Configurator-TC	SW0D5C-QTCU-E	1.23Z or later*1	1.23Z or later*2	1.23Z or later*3	1.23Z or later*4	N/A
GX Configurator-FL	SW0D5C-QFLU-E	1.23Z or later*1	1.23Z or later*2	1.23Z or later*3	1.23Z or later*4	N/A
GX Configurator-QP	SW2D5C-QD75P-E	2.25B or later	2.29F or later	2.30G or later*5	2.32J or later	N/A
GX Configurator-PT	SW1D5C-QPTU-E	1.23Z or later*1	1.23Z or later*2	1.23Z or later*3	1.23Z or later*4	N/A
GX Configurator-AS	SW1D5C-QASU-E	1.21X or later*1	1.21X or later*2	1.21X or later*3	1.21X or later*4	N/A
GX Configurator-MB	SW1D5C-QMBU-E	1.08J or later*1	1.08J or later*2	1.08J or later*3	1.08J or later*4	N/A
GX Configurator-DN	SW1D5C-QDNU-E	1.23Z or later*1	1.23Z or later*2	1.24AA or later*3	1.24AA or later*4	N/A
MX Component	SW3D5C-ACT-E	3.09K or later	3.10L or later	3.11M or later	3.12N or later	4.02C or later
GX Simulator	SW7D5C-LLT-E	7.23Z or later*4	7.23Z or later*4	7.23Z or later*4	7.23Z or later*4	N/A
MESInterface	IT VN-SWMIT1-E	No restrictions				1.12N or later

*1 The software can be used by installing GX Developer Version 8.48A or later.
*2 The software can be used by installing GX Developer Version 8.62Q or later.
*3 The software can be used by installing GX Developer Version 8.68W or later.
*4 The software can be used by installing GX Developer Version 8.78G or later.
*5 GX Configurator-QP Version 2.29F can be used when connected via USB.

Software not supported by the Universal model QCPU

The following table shows software not supported by the Universal model QCPU.

■Software not supported by the Universal model QCPU

Product	Model
GX Explorer	SW□D5C-EXP-E
GX Converter	SW□D5C-CNVW-E

4 INSTRUCTIONS

4.1 Instructions not Supported in the Universal Model QCPU and Replacing Methods

Replace the instructions not supported in the Universal model QCPU using alternative methods described in the tables. (For other instructions, replacement is not required.)

Instructions not supported in the Universal model QCPU and alternative methods

Symbol	Instruction	Replacing method	Reference
IX IXEND	Index modification of entire ladder	Use alternative programs.	Page 17 Replacement example of the IX and IXEND instructions
IXDEV IXSET	Modification value specification in index modification of entire ladder	Change the program so that the device offset values specified by the IXSET instruction are directly set to the index modification table using the MOV instruction.	Page 19 Replacement example of the IXDEV and IXSET instructions
PR	Print ASCII code instruction	• It is recommended to use GOT as an ASCII code display device. ASCII codes stored in devices are directly displayed as characters on GOT. • Instructions can be replaced using a replacement program.	Page 21 Replacement example of the PR instruction
PRC	Print comment instruction	• It is recommended to use GOT as an ASCII code display device. Device comments can be displayed on GOT. • Comment data can be output to a display device in the replacement program of the PR instruction after reading data using the reading device comment data instruction (COMRD(P)).	
CHKST CHK	Specific format failure check instruction	Instructions can be replaced using a replacement program.	Page 24 Replacement example of the CHKST and CHK instructions
CHKCIR CHKEND	Format change instruction for CHK instruction	Failure detection ladder patterns can be changed in a replacement program.	
PLOW	Program low-speed execution registration instruction	• Use the PSCAN instruction instead of this instruction when low-speed execution type programs are replaced with scan execution type programs. • No instruction can be used if low-speed execution type programs are replaced with fixed scan execution type programs.	—
PCHK	Program execution status check instruction	Check the execution status of a program on the Program monitor list window. For details, refer to Section 3.13.1 in the QnUCPU User's Manual (Function Explanation, Program Fundamentals).	—
KEY	Numerical key input instruction	• It is recommended to use GOT as a numeral input device. • Instructions can be replaced using a replacement program.	Page 27 Replacement example of the KEY instruction
PLOADP PUNLOADP PSWAPP	Load program from memory card Unload program from memory card Load + Unload	Store all programs to be executed in the program memory. The Universal model QCPU can neither add programs to the program memory nor change them with other programs during RUN. If the capacity of the program memory is not enough, store parameters, device comments, and device initial values in the program memory into the standard ROM or memory card instead.	—

SFC control instructions not supported in the Universal model QCPU and alternative methods

Symbol	Instruction	Alternative method
LD TRn	Forced transition check instruction	When the programmable controller type is changed, these instructions are converted into SM1255. Modify programs as needed.
AND TRn		
OR TRn		
LDI TRn		
ANDI TRn		
ORI TRn		
LD BLm\TRn		
AND BLm\TRn		
OR BLm\TRn		
LDI BLm\TRn		
ANDI BLm\TRn		
ORI BLm\TRn		
SCHG(D)	Active step change instruction	Refer to Appendix 3 "Restrictions on Basic Model QCPU, Universal Model QCPU, and LCPU and Alternative Methods" in the MELSEC-Q/L/QnA Programming Manual (SFC).
SET TRn	Transition control instruction	Refer to Appendix 3 "Restrictions on Basic Model QCPU, Universal Model QCPU, and LCPU and Alternative Methods" in the MELSEC-Q/L/QnA Programming Manual (SFC).
SET BLm\TRn		
RST TRn		
RST BLm\TRn		
BRSET(S)*1	Block switching instruction	When the programmable controller type is changed, these instructions are converted into SM1255. Modify programs as needed.

*1 Usable for the Universal model CPU whose serial number (first five digits) is "13102" or later.

4.2 Replacing Programs Using Multiple CPU Transmission Dedicated Instructions

Replacing the module with the QnUD(H)CPU or Built-in Ethernet port QCPU

If the instructions listed below are used, replace them with the alternative instructions in the table.
For the specifications of each instruction, refer to the manuals for the Motion CPU.

■Instructions not supported in the QnUD(H)CPU and Built-in Ethernet port QCPU and alternative instructions

Symbol	Instruction description	Symbol of alternative instruction
S(P).DDWR	Write other CPU device data into host CPU	D(P).DDWR
S(P).DDRD	Read other CPU device data into host CPU	D(P).DDRD
S(P).SFCS	Request of motion SFC program startup	D(P).SFCS
S(P).SVST	Request of servo program startup	D(P).SVST
S(P).CHGA	Current value change of halted axis/synchronized encoder/cam axis	D(P).CHGA
S(P).CHGV	Axis speed change during positioning and JOG operation	D(P).CHGV
S(P).CHGT	Torque control value change during operation and suspension in real mode	D(P).CHGT
S(P).GINT	Request of other CPU interrupt program startup	D(P).GINT

Replacing the module with the Q02UCPU

The Q02UCPU supports the same multiple CPU transmission dedicated instructions used in the Basic model QCPU.
The alternative instructions for the QnUD(H)CPU and Built-in Ethernet port QCPU are not available for the Q02UCPU. For the alternative instructions for the QnUD(H)CPU and Built-in Ethernet port QCPU, refer to the following.
☞ Page 17 Instructions not supported in the QnUD(H)CPU and Built-in Ethernet port QCPU and alternative instructions

4.3 Program Replacement Examples

This section shows program replacement examples for the instructions that are not supported in the Universal model QCPU and can be replaced with replacement programs. Skip this section if instructions not supported in the Universal model QCPU are not used. For the instructions not supported in the Universal model QCPU, refer to the following.
☞ Page 15 Instructions not Supported in the Universal Model QCPU and Replacing Methods

Replacement example of the IX and IXEND instructions

A replacement example of program using the IX and IXEND instructions is shown below.
To save index register data using the ZPUSH instruction, a 23-word index register save area is required.

■Example of device assignment

(Before replacement)

Application	Device
Index modification table	D100 to D115

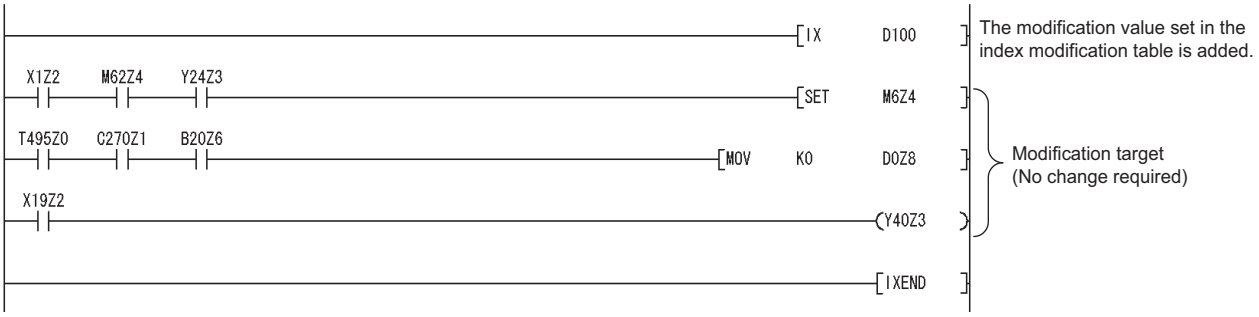
⇒

(After replacement)

Application	Device
Index modification table	D100 to D115
Index register save area	D200 to D222

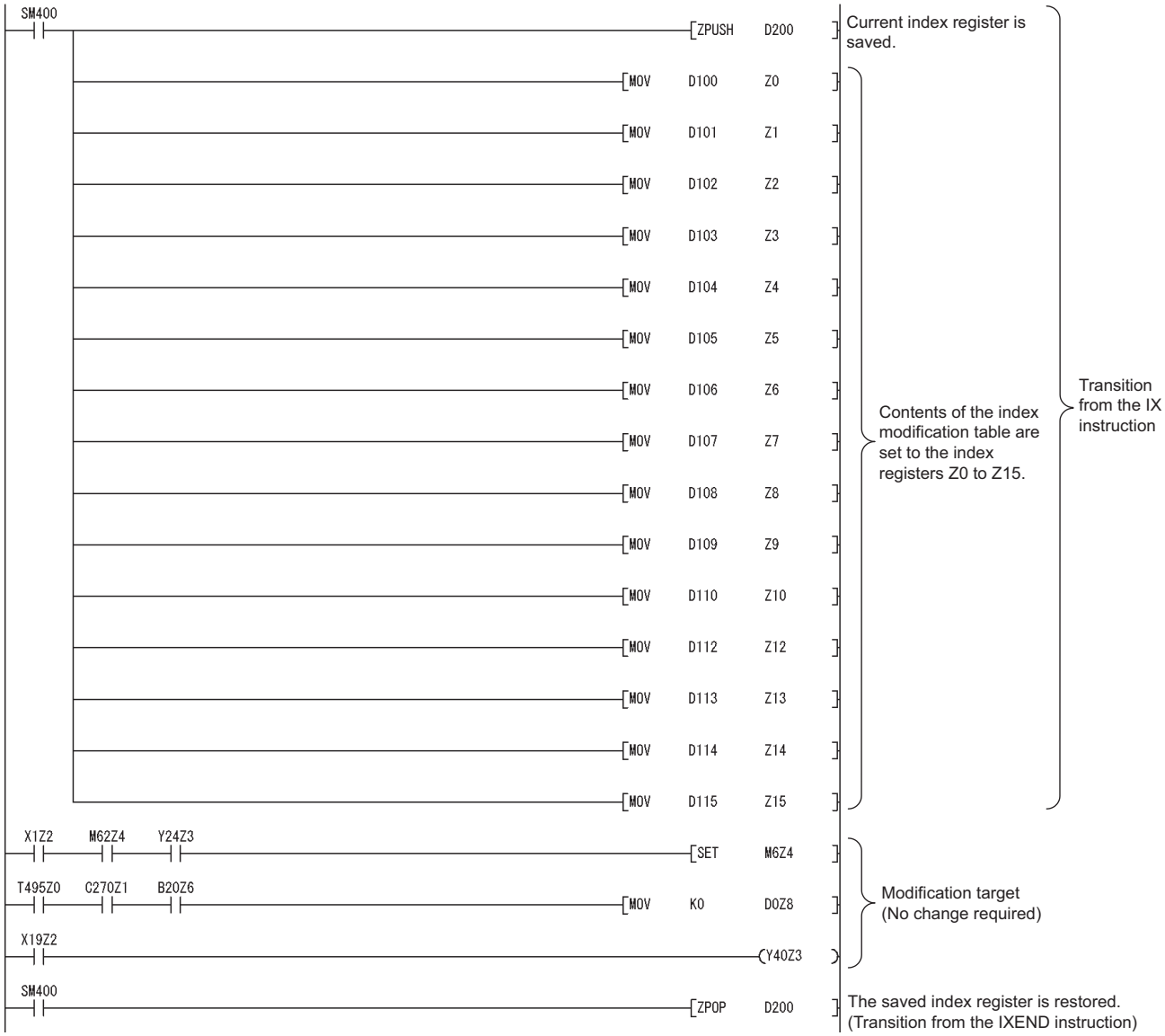
If the device numbers in the example above are used for other applications, assign unused device numbers instead.

■Program before replacement



■Program after replacement

- Replace the IX instruction with the ZPUSH instruction and set the contents of index modification table in the to index register.
- Replace the IXEND instruction with the ZPOP instruction.

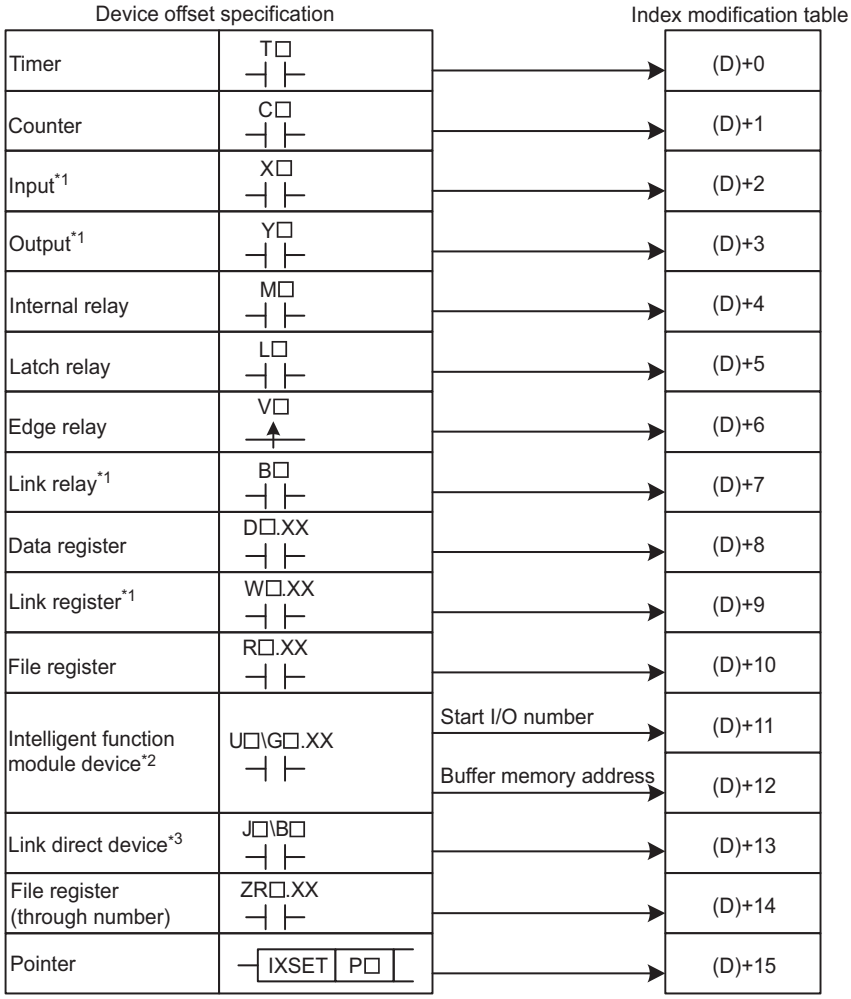


Replacement example of the IXDEV and IXSET instructions

Change the program so that the device offset values specified for the contacts between the IXDEV and IXSET instructions are directly set to the index modification table using the MOV instruction.

For the devices whose device offset value is not specified by the IXDEV and IXSET instructions, set the device offset value to 0 in the program after replacement.

The following figure shows how the device offset value is set in the program before and after replacement by the IXDEV and IXSET instructions.

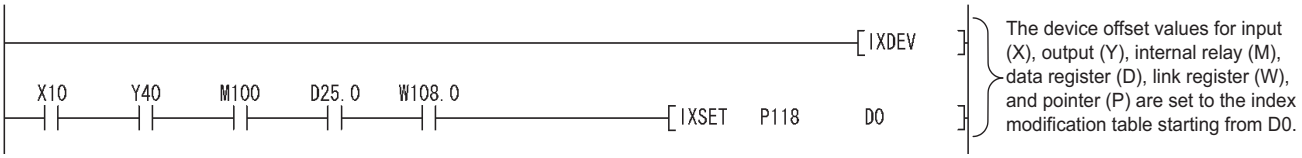


*1 Device numbers are represented in hexadecimal. Use hexadecimal constants (H□) when setting values in the index modification table.

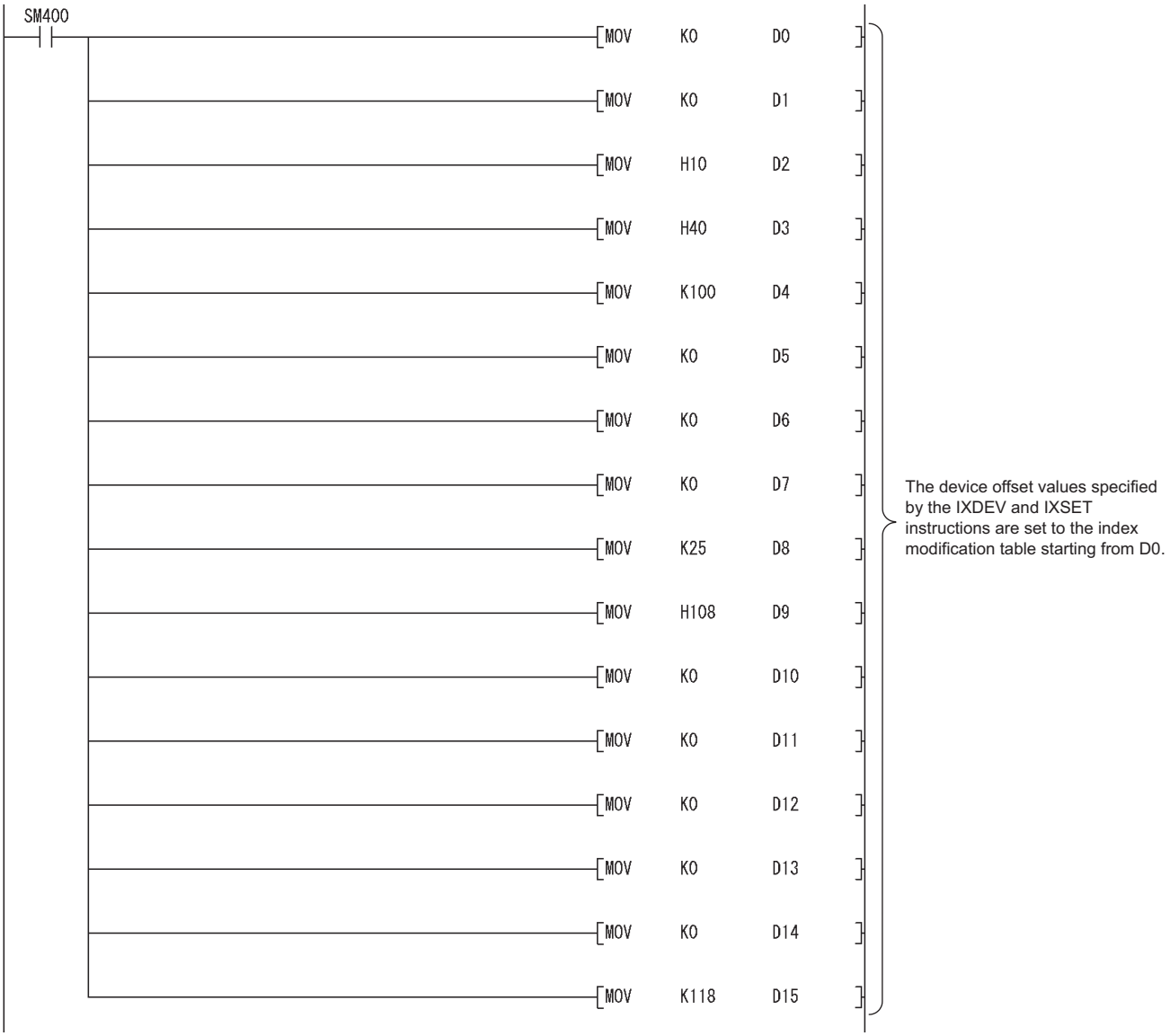
*2 Start I/O numbers (U□) are represented in hexadecimal. Use hexadecimal constants (H□) when setting values in the index modification table.

*3 Devices B, W, X, or Y can be specified following J□\. Set device numbers for B, W, X, and Y as device offset values of each device in the index modification table.
For example, if 'J10\Y220' is specified by the IXDEV or IXSET instruction, set 'K10' in (D)+13 and 'H220' in (D)+3 in the replacement program. ((D) indicates the start device in the index modification table.)

■Program before replacement



■Program after replacement



Replacement example of the PR instruction

The number of output characters can be switched by the on/off status of SM701.

■Example of device assignment

(Before replacement)

Application	Device
Output string	D0 to D13
ASCII code output signal	Y100 to Y107
Strobe signal	Y108
In-execution flag	Y109

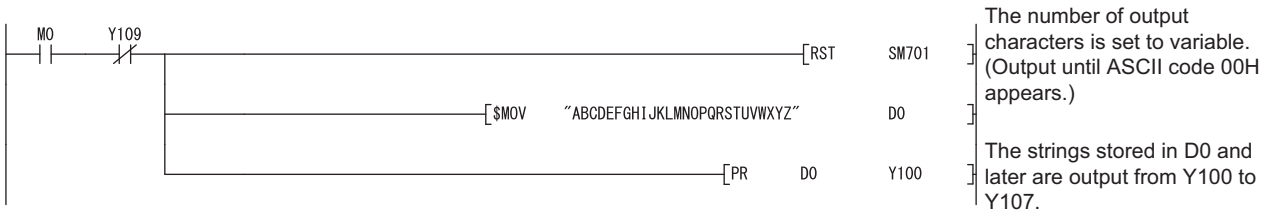
⇒

(After replacement)

Application	Device
Output string	D0 to D13
ASCII code output signal	Y100 to Y107
Strobe signal	Y108
In-execution flag	Y109
Output string storage address (BIN32)	D20 to D21
Output string storage address (BIN32) (Used for sub-routine programs and interrupt programs)	D200 to D201
Number of output characters	D202
Output module start Y number	D203
Character extraction position	D204
Number of extracted characters	D205
String output status value	D206
Result of string extraction by the MIDR instruction	D207
String output in-execution flag	M200
For index modification	Z0

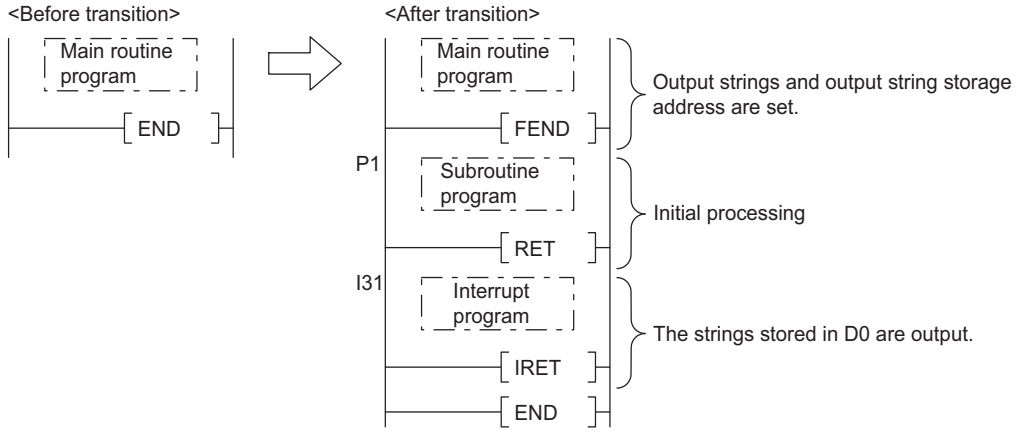
If the device numbers in the example above are used for other applications, assign unused device numbers instead.

■Program before replacement



■Program after replacement

In the sequence program after replacement, three programs are required as shown below.

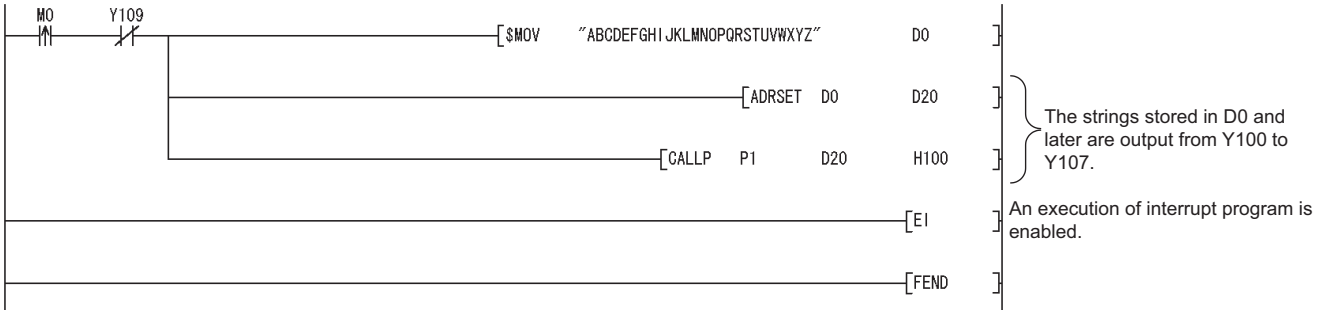


• Main routine program

Replace the PR instruction with the CALL instruction so that a subroutine program is called.

Output string storage device ('D0' in the program below) cannot be specified directly with the CALL instruction. Use the ADRSET instruction to acquire the indirect address for the CALL instruction. Y device ('Y100' in the program before replacement) cannot be specified directly as output Y number with the CALL instruction. Specify the output Y number in integer.

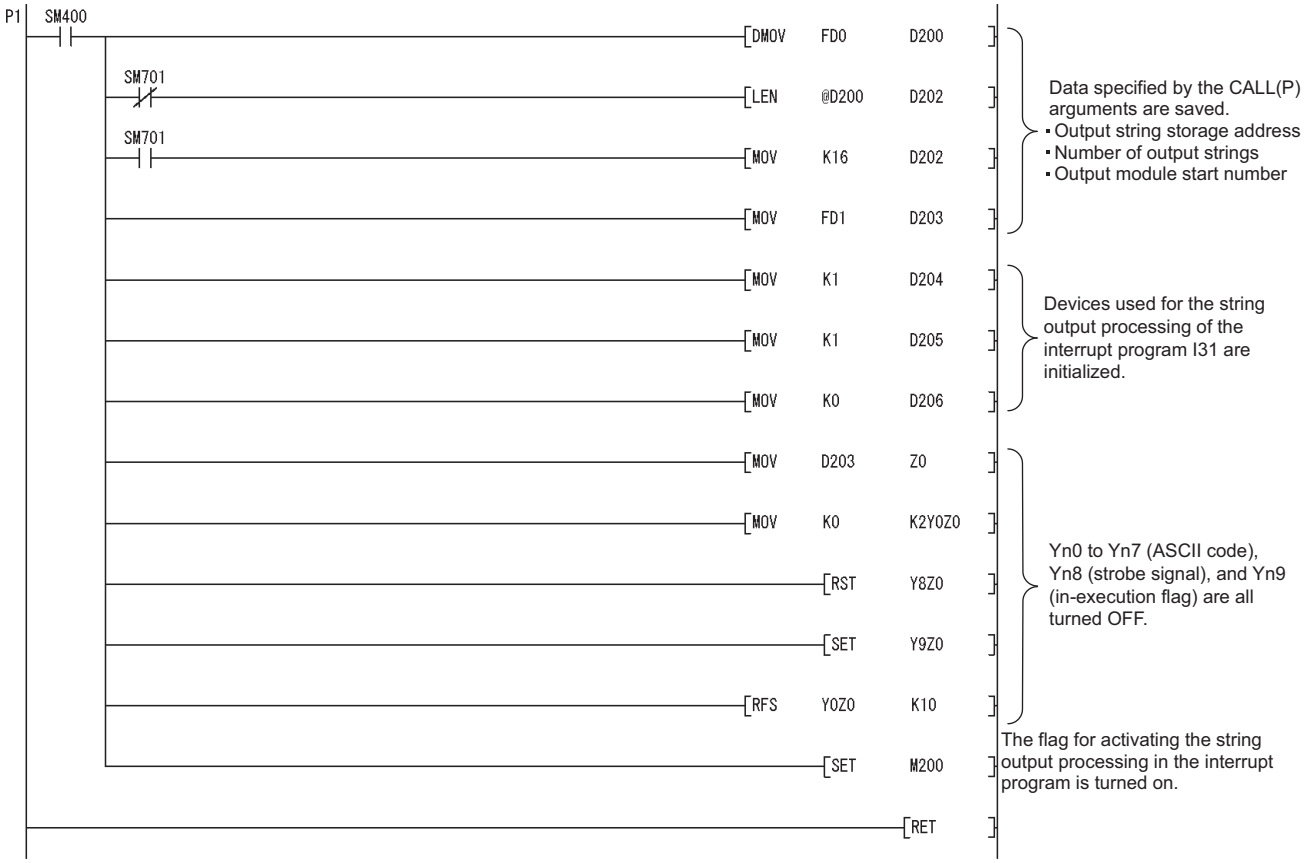
The program is used as an interrupt program to output character codes via the output module. Enable the execution of interrupt program using the EI instruction.



• Subroutine program

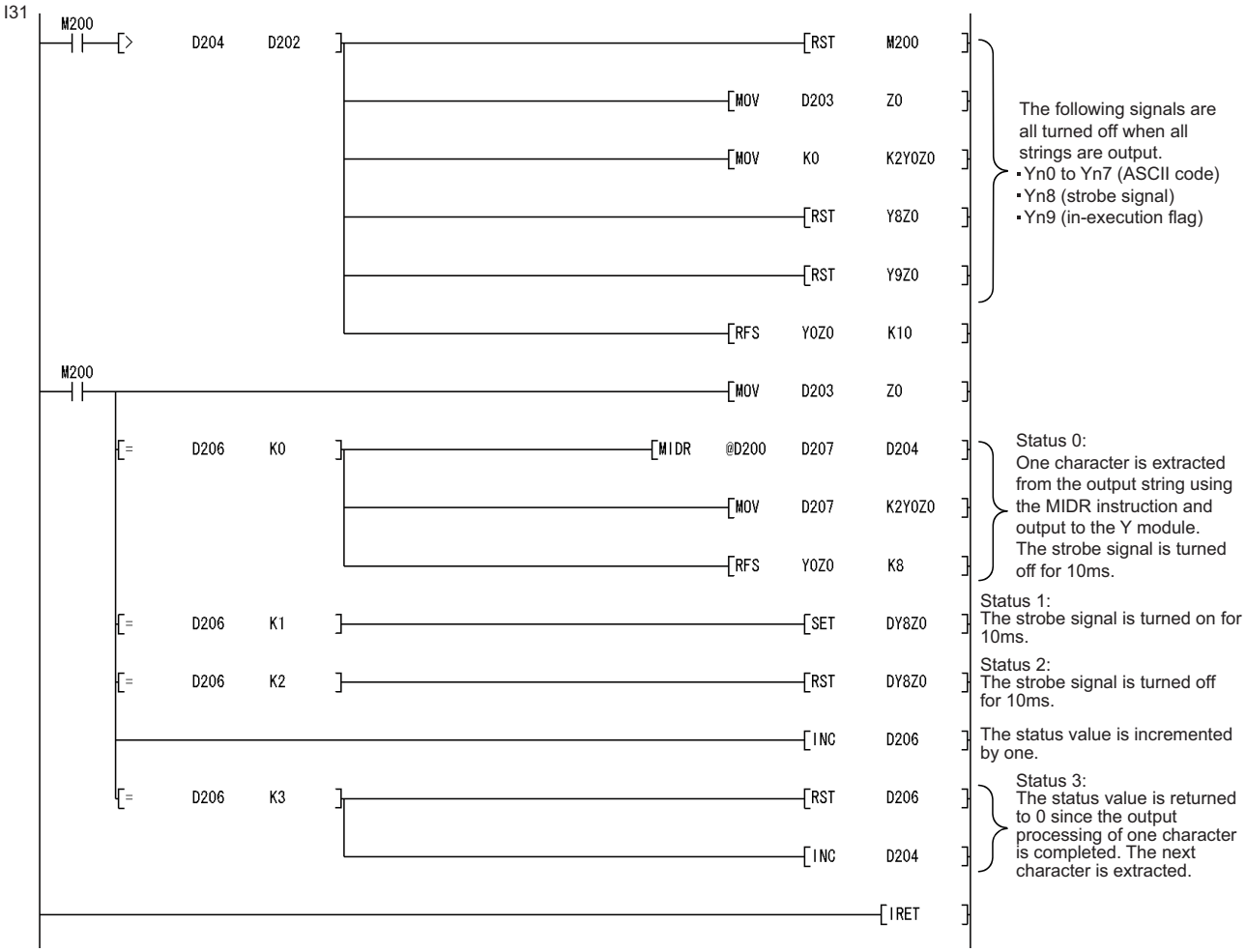
In the subroutine program, the data for outputting ASCII codes using a fixed scan interrupt program (10ms) are set to work devices. Also, the flag for activating the processing in the fixed scan interrupt program is turned on. Specify the following arguments for the subroutine program.

First argument	Output string storage address	(Input)
Second argument	Output module start Y number	(Input)



• Interrupt program

The following processing is added to a fixed scan interrupt program (10ms). The fixed scan interrupt program outputs ASCII codes from the output module and controls the strobe signal.



Replacement example of the CHKST and CHK instructions

In the example below, if the replacement program for the CHKST and CHK instructions detects a failure, a failure number (contact number + coil number) is stored in D200 and the annunciator F200 is turned on.

■Example of device assignment

(Before replacement)

Application	Device
Advance end detection sensor input 1	X100
Retract end detection sensor input 1	X101
Advance end detection sensor input 2	X102
Retract end detection sensor input 2	X103
Advance end detection sensor input 3	X104
Retract end detection sensor input 3	X105
Advance end detection sensor input 4	X106
Retract end detection sensor input 4	X107
Failure detection output 1	Y100
Failure detection output 2	Y102
Failure detection output 3	Y104
Failure detection output 4	Y106

⇒

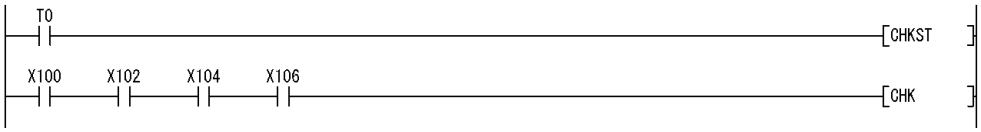
(After replacement)

Application	Device
Advance end detection sensor input 1	X100
Retract end detection sensor input 1	X101
Advance end detection sensor input 2	X102
Retract end detection sensor input 2	X103
Advance end detection sensor input 3	X104
Retract end detection sensor input 3	X105
Advance end detection sensor input 4	X106
Retract end detection sensor input 4	X107
Failure detection output 1	Y100
Failure detection output 2	Y102
Failure detection output 3	Y104
Failure detection output 4	Y106
Coil number (failure type detected)	D100
Contact number	D101
Failure number	D200
Failure detection display	F200
For index modification	Z0

If the device numbers in the example above are used for other applications, assign unused device numbers instead.
When the advance end detection sensor input performs a failure detection of Xn, assign device numbers for the retract end detection sensor input and the failure detection output as described below.

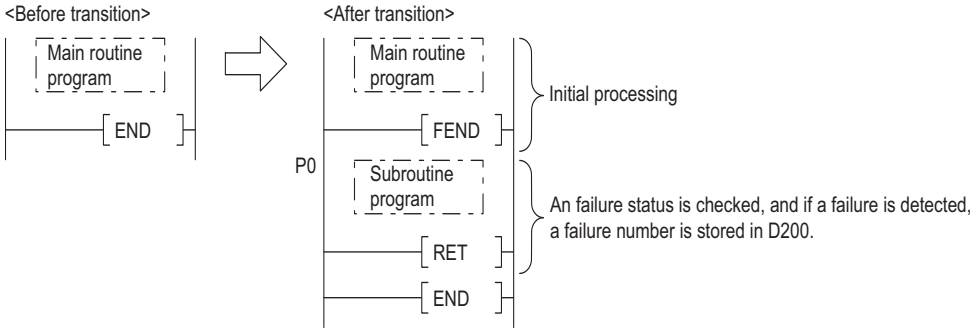
Advance end detection sensor input	Xn
Retract end detection sensor input	Xn+1
Failure detection output	Yn

■Program before replacement



■Program after replacement

In the sequence program after replacement, two programs are required as shown below.



• Main routine program

Replace the CHKST and CHK instructions with the CALL instructions so that a subroutine program is called.

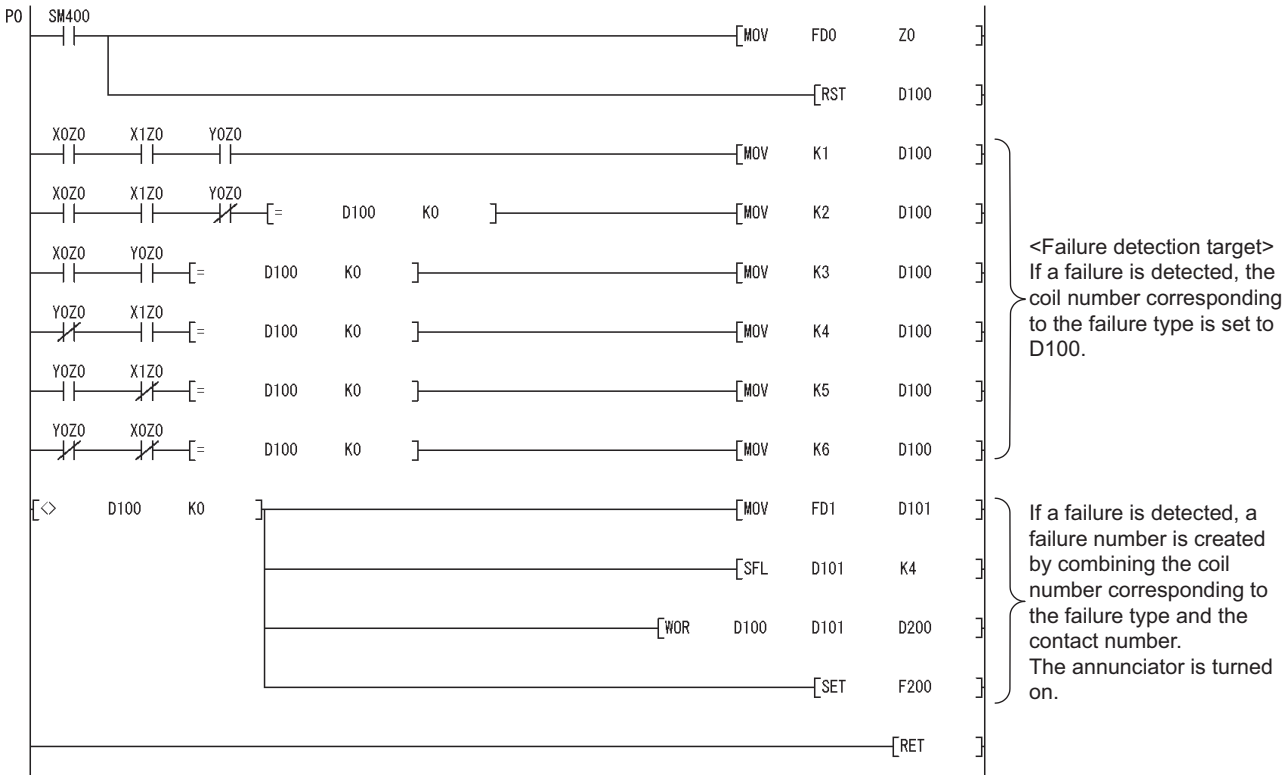
One CALL instruction is required for each device specified as check condition before the CHK instruction. (In the program before replacement, four CALL instructions need to be added since there are four check conditions before the CHK instruction.)

Device number and contact number of X devices (check condition) are specified in each CALL instruction. Contact number is used to display failure number when a failure is detected.

• Subroutine program

In the subroutine program, a failure status is checked using a failure detection ladder pattern. If a failure is detected, a failure number is stored in D200 and the annunciator F200 is turned on. Specify the following arguments for the subroutine program.

First argument	Device number of X device targeted for failure check	(Input)
Second argument	Contact number of X device targeted for failure check	(Input)



■Replacement method when failure detection ladder patterns are changed by the CHKCIR and CHKEND instructions

Failure detection ladder patterns can be changed in the subroutine program of the program after replacement.

Replacement example of the KEY instruction

■Example of device assignment

(Before replacement)

Application	Device
Numeric input execution instruction	M0
Input complete flag	M1
Input data area	D200 to D203
ASCII code input signal	X100 to X107
Strobe signal	X108

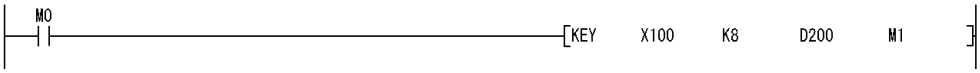
⇒

(After replacement)

Application	Device
Numeric input execution instruction	M0
Input complete flag	M1
Input data area	D200 to D203
ASCII code input signal	X100 to X107
Strobe signal	X108
Input data area address (BIN32)	D210 to D211
(Input data area + 0) address (BIN32)	D212 to D213
(Input data area + 1) address (BIN32)	D214 to D215
(Input data area + 2) address (BIN32)	D216 to D217
For shifting input data	D218
For converting input data	D219 to D220

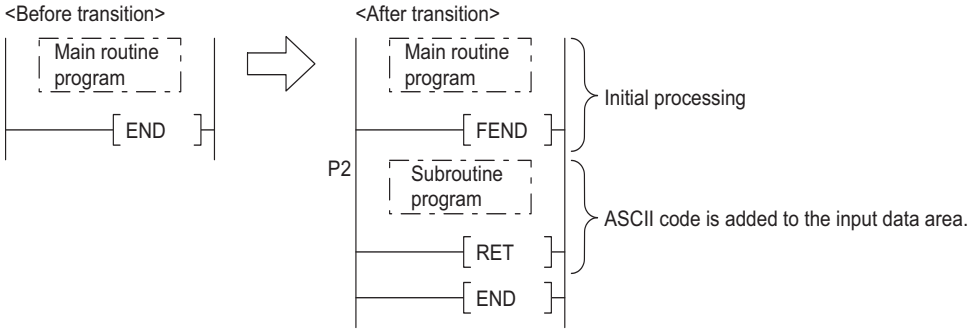
If the device numbers in the example above are used for other applications, assign unused device numbers instead.

■Program before replacement



■Program after replacement

In the sequence program after replacement, two programs are required as shown below.



• Main routine program

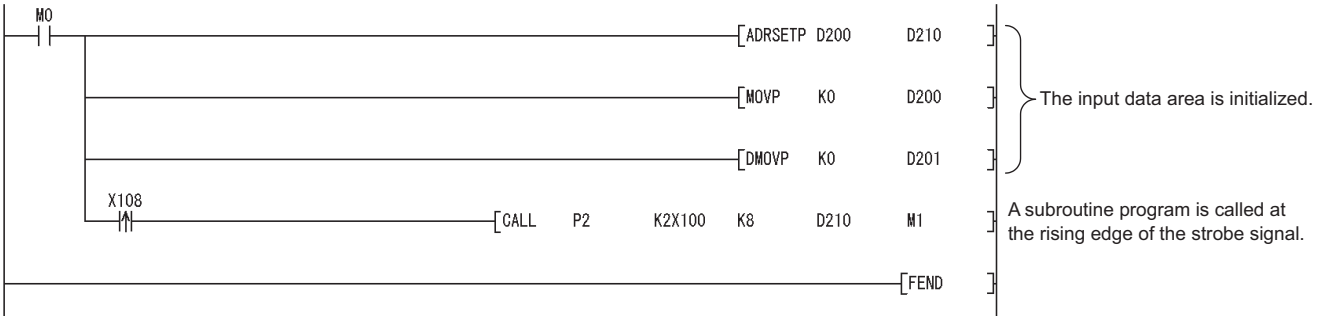
Set '0' in the input data area on the rising edge of the execution instruction ('M0' in the program below) and initialize the program.

Execute the CALL instruction on every rising edge of the strobe signal ('X108' in the program below) so that a subroutine program is called.

In the subroutine program, input codes are added to the input data area and the completion status is checked.

Pass the following data to the subroutine program at the execution of the CALL instruction.

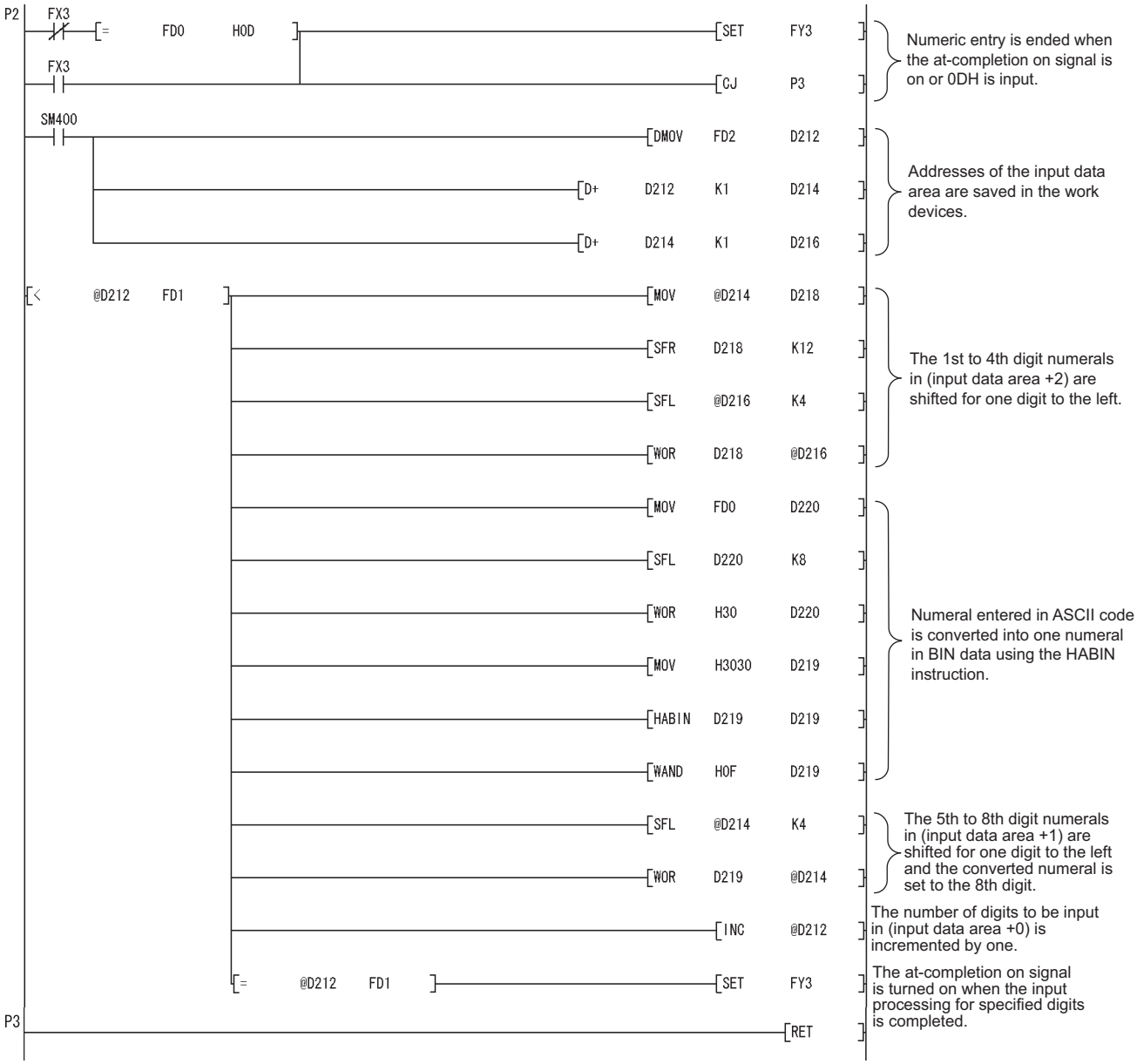
- ASCII code input value from the input module (Xn0 to Xn7)
- Number of digits to be input
- Indirect address of the input data area (Use the ADRSET instruction to acquire the indirect address for the input data area)
- Bit devices to be turned on when input is completed



• Subroutine program

In the subroutine program, ASCII codes specified by an argument are added to the input data area and the completion status is checked. Specify the following arguments for the subroutine program.

First argument	ASCII code input from the input module (K2Xn)	(Input)
Second argument	Number of digits to be input	(Input)
Third argument	Indirect address of the input data area	(Input)
Fourth argument	Bit device turned on when input is completed	(Output)



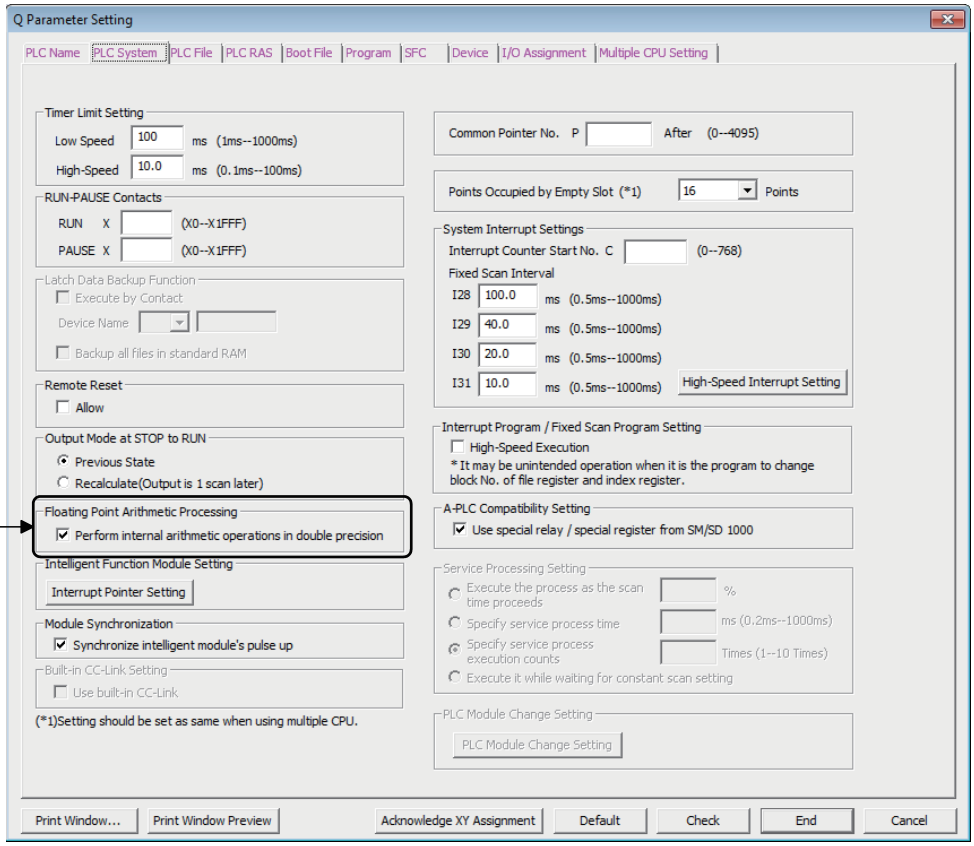
5 FUNCTIONS

5.1 Floating-point Operation Instructions

Differences between the High Performance model QCPU and Universal model QCPU

■High Performance model QCPU

The High Performance model QCPU can perform only the single-precision floating-point operation instructions.
Note, however, that internal operation processing can be performed in double-precision by selecting the item shown below (default: selected).



Selected by default.

■Universal model QCPU

The Universal model QCPU supports the double-precision floating-point operation instructions.
The operation can be performed either in single-precision or double-precision depending on the data.
Therefore, "Perform internal arithmetic operations in double-precision" item in the PLC system tab of the PLC parameter dialog box cannot be selected.
Because of this new function, operation results (both in single-precision and double-precision) slightly differ between the High Performance model QCPU and the Universal model QCPU if "Perform internal arithmetic operations in double-precision" is selected in the High Performance model QCPU.
If higher accuracy is required in floating-point operations, replace the floating-point operation instructions as described below.
☞ Page 33 Replacing all single-precision floating-point operation instructions with double-precision floating-point operation instructions
However, if six or less digits are used as significant digits for the floating-point operation instructions, replacement is not necessary. The single-precision floating-point operation results in the Universal model QCPU can be used as they are in the system.
When not replacing instructions, make sure that it does not cause any problems in the system.

Floating-point operation instructions for the Universal model QCPU

The following table lists floating-point operation instructions for the Universal model QCPU.
Specifications of the single-precision floating-point operation instructions are compatible with those for the High Performance model QCPU.

■List of floating-point operation instructions supported in the Universal model QCPU

Instruction name		Instruction symbol		Remarks
		Single-precision floating-point data	Double-precision floating-point data	
Comparison	Floating-point data comparison	LDE☐	LDED☐	☐ indicates one of the following; <>, =, <, >, <=, >=
		ANDE☐	ANDED☐	
		ORE☐	ORED☐	
Data transfer	Floating-point data transfer	EMOV(P)	EDMOV(P)	—
Four arithmetic operation	Floating-point data addition	E+(P)	ED+(P)	—
	Floating-point data subtraction	E-(P)	ED-(P)	
	Floating-point data multiplication	E*(P)	ED*(P)	
	Floating-point data division	E/(P)	ED/(P)	
Data conversion	Conversion from BIN 16-bit data to floating-point data	FLT(P)	FLTD(P)	—
	Conversion from BIN 32-bit data to floating-point data	DFLT(P)	DFLTD(P)	
	Conversion from floating-point data to BIN 16-bit data	INT(P)	INTD(P)	
	Conversion from floating-point data to BIN 32-bit data	DINT(P)	DINTD(P)	
	Floating-point sign inversion	ENEG(P)	EDNEG(P)	
Special function	SIN operation	SIN(P)	SIND(P)	—
	COS operation	COS(P)	COSD(P)	
	TAN operation	TAN(P)	TAND(P)	
	SIN^{-1}operation	ASIN(P)	ASIND(P)	
	COS^{-1}operation	ACOS(P)	ACOSD(P)	
	TAN^{-1}operation	ATAN(P)	ATAND(P)	
	Conversion from angle to radian	RAD(P)	RADD(P)	
	Conversion from radian to angle	DEG(P)	DEGD(P)	
	Square root	SQR(P)	SQRD(P)	
	Exponential operation	EXP(P)	EXPD(P)	
	Natural logarithm operation	LOG(P)	LOGD(P)	

Floating-point data can be converted mutually between single-precision and double-precision using instructions listed below.

Instruction name	Instruction symbol
Single-precision to double-precision conversion	ECON(P)
Double-precision to single-precision conversion	EDCON(P)

Advantages and disadvantages when using the double-precision floating-point data

The following table shows the advantages and disadvantages when executing the double-precision floating-point operation instructions in the Universal model QCPU.
If higher accuracy is required in floating-point operations, it is recommended to replace the instructions with the double-precision floating-point operation instructions.

■Advantages and disadvantages when using the double-precision floating-point operation instructions

Advantage	Disadvantage
The results are more accurate than those of the single-precision floating-point operation instructions.	The instruction processing speed is slower than that of the single-precision floating-point operation instructions.*1 Double-precision floating-operation data use twice as many word device points as single-precision floating-operation data.

*1 The processing speed of the double-precision floating-point operation instructions in the Universal model QCPU is higher than that of floating-point operation instructions using internal double-precision operations in the High Performance model QCPU.

The following table shows the comparison between single-precision and double-precision floating-point data.

Item		Single-precision floating-point data	Double-precision floating-point data
Word point required for data retention		2 words	4 words
Setting range		$-2^{128}<N\leq-2^{-126}$, 0, $2^{-126}\leq N<2^{128}$	$-2^{1024}<N\leq-2^{-1022}$, 0, $2^{-1022}\leq N<2^{1024}$
Precision (number of bits)	Mantissa part	23 bits	52 bits
	Exponent part	8 bits	11 bits
	Sign part	1 bit	1 bit
Instruction processing speed (Q04UDHCPU/ Q06UDHCPU) (minimum)	Data comparison (Conductive status) (LDE>= / LDED>=)	0.0285μs	3.6μs
	Data transfer (EMOV/EDMOV)	0.019μs	1.7μs
	Addition (3 devices) (E+ / ED+)	0.0665μs	4.8μs
	SIN operation (SIN/SIND)	4.1μs	8.5μs
Instruction processing speed (High-speed Universal model QCPU) (minimum)	Data comparison (Conductive status) (LDE>= / LDED>=)	0.0098μs	1.8μs
	Data transfer (EMOV/EDMOV)	0.0039μs	0.0078μs
	Addition (3 devices) (E+ / ED+)	0.015μs	1.9μs
	SIN operation (SIN/SIND)	1.6μs	2.6μs

Method of replacing High Performance model QCPU with Universal model QCPU

■Replacing all single-precision floating-point operation instructions with double-precision floating-point operation instructions

Single-precision floating-point data occupy two points of word device per data. On the other hand, four points are required per double-precision floating-point data. Therefore, all device numbers for storing floating-point data need to be reassigned.

Ex.

Replacing the floating-point operation 'A×B+C' (Changing all floating-point data into double-precision.)

• Device assignment

(Before replacement)

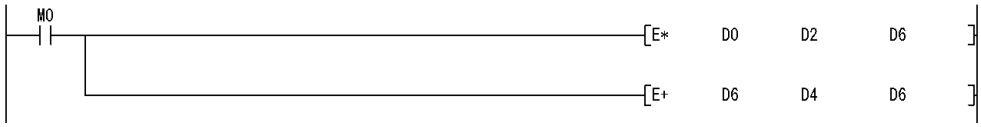
Application	Device	Data type
Data A	D0 to D1	Floating-point data (single precision)
Data B	D2 to D3	
Data C	D4 to D5	
Result	D6 to D7	

⇒

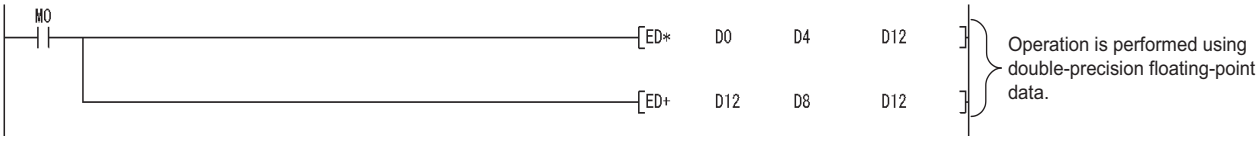
(After replacement)

Application	Device	Data type
Data A (D)	D0 to D3	Floating-point data (double precision)
Data B (D)	D4 to D7	
Data C (D)	D8 to D11	
Result (D)	D12 to D15	

• Program before replacement



• Program after replacement



■Replacing a part of floating-point operation instructions with double-precision floating-point operation instructions

Only operations that require high accuracy are replaced with double-precision floating-point operation instructions. Using the ECON and EDCON instructions, convert floating-point data mutually between single-precision and double-precision. The flow of a replacement program is as follows:

- Data required for operations are converted from single-precision to double-precision using the ECON instruction.
- Operations are performed in double-precision using the double-precision floating-point operation instructions.
- Operation results are converted from double-precision to single-precision using the EDCON instruction.

A program example that floating-point data are converted mutually between single-precision and double-precision before and after operations is shown below.

Ex.

Replacing the floating-point operation 'A×B+C' (Using the ECON and EDCON instructions)

- Device assignment

(Before replacement)

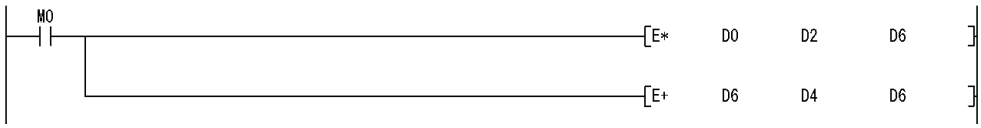
Application	Device	Data type
Data A	D0 to D1	Floating-point data (single precision)
Data B	D2 to D3	
Data C	D4 to D5	
Result	D6 to D7	

⇒

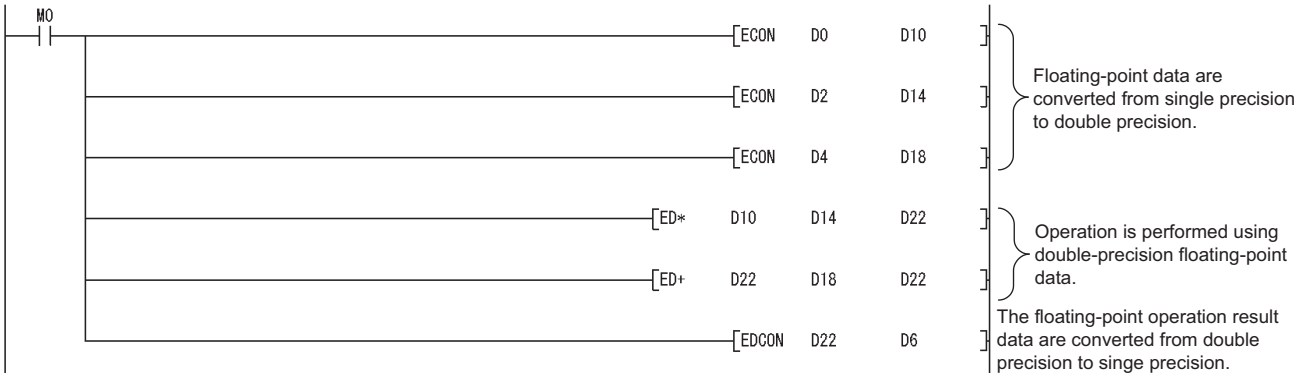
(After replacement)

Application	Device	Data type
Data A	D0 to D1	Floating-point data (single precision)
Data B	D2 to D3	
Data C	D4 to D5	
Result	D6 to D7	
Data A (D)	D10 to D13	Floating-point data (double precision)
Data B (D)	D14 to D17	
Data C (D)	D18 to D21	
Result (D)	D22 to D25	

- Program before replacement



- Program after replacement



■Replacing a part of floating-point operation instructions with double-precision floating-point operation instructions using subroutine programs

The flow of a replacement program described in "Replacing a part of floating-point operation instructions with double-precision floating-point operation instructions" above can be regarded as one subroutine program.

First, create subroutine programs for each floating-point operation instruction.

Then replace the original floating-point operation instructions with the CALL(P) instructions so that the corresponding subroutine program is called. With this method, changes in the program are minimized. But the processing for calling subroutine programs increases the scan time.

With this method, since conversions from double-precision to single-precision are performed for each instruction, rounding-off errors generated during operations are larger than those in the replacement program described below.

☞ Page 34 Replacing a part of floating-point operation instructions with double-precision floating-point operation instructions

Ex.

Replacing the floating-point operation 'A×B+C' (Using a subroutine program)

• Device assignment

(Before replacement)

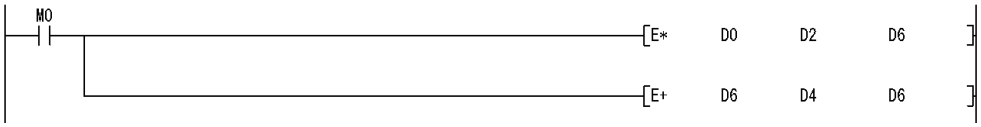
Application	Device	Data type
Data A	D0 to D1	Floating-point data (single precision)
Data B	D2 to D3	
Data C	D4 to D5	
Result	D6 to D7	

⇒

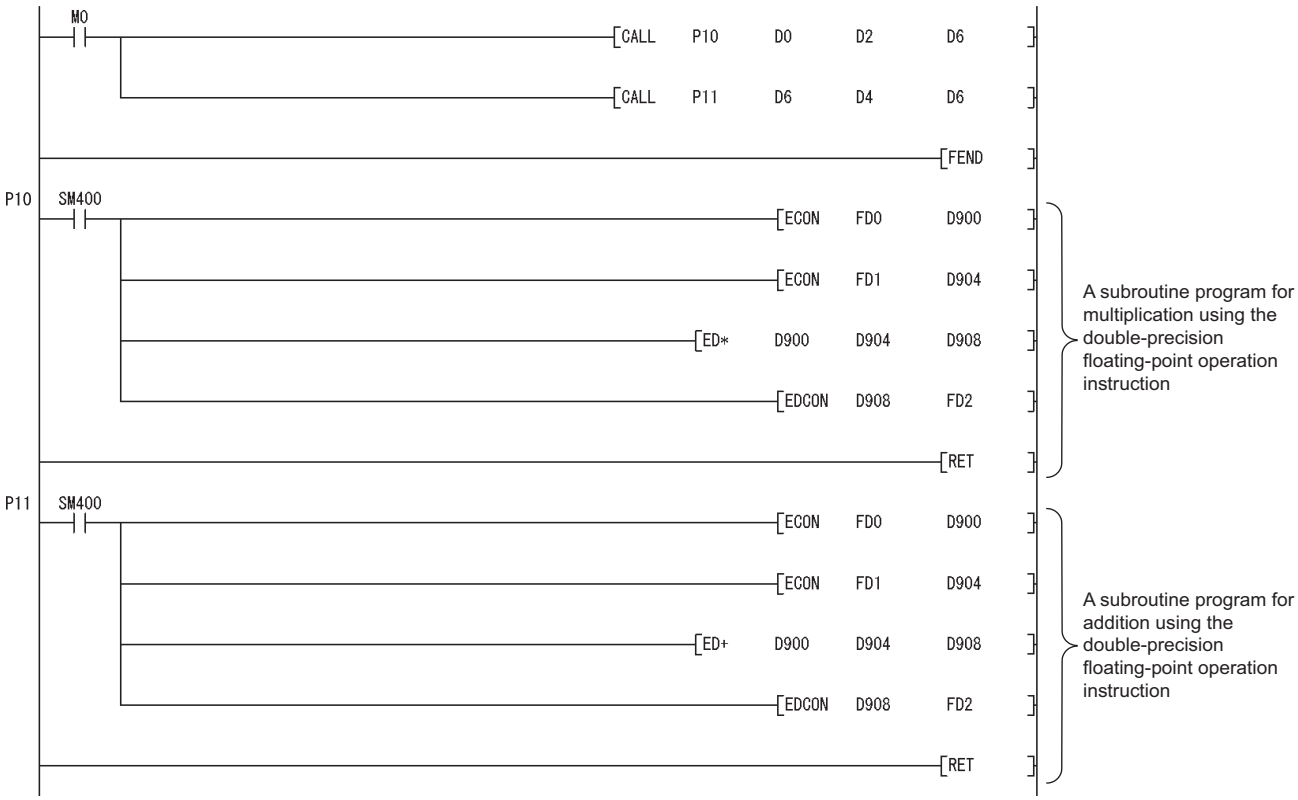
(After replacement)

Application	Device	Data type
Data A	D0 to D1	Floating-point data (single precision)
Data B	D2 to D3	
Data C	D4 to D5	
Result	D6 to D7	
Subroutine input data 1	D900 to D903	Floating-point data (double precision)
Subroutine input data 2	D904 to D907	
Subroutine operation result	D908 to D911	

• Program before replacement



• Program after replacement



5.2 Error Check Processing for Floating-point Data Comparison Instructions (excluding High-speed Universal model QCPU)

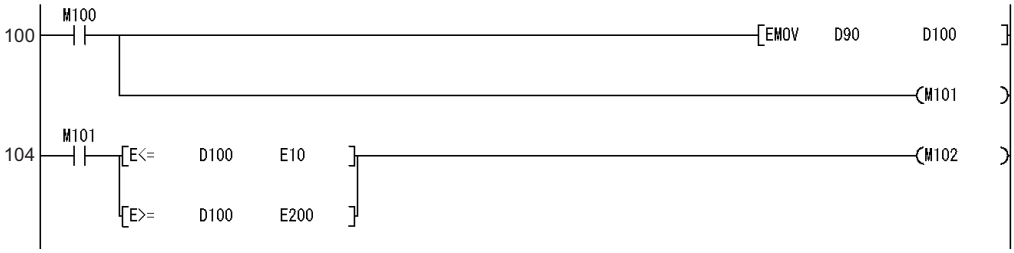
Input data check

Error check processing for floating-point data comparison instructions has been enhanced for the Universal model QCPU.
Input of a "special value" (-0, nonnumeric, unnormalized number, or ±∞) is checked, and if any special value are input, the CPU module detects "OPERATION ERROR" (error code: 4140).
When the LDE☐, ANDE☐, ORE☐, LDED☐, ANDED☐, and/or ORED☐ instructions (☐ indicates one of the following: =, <>, <, >, <=, >=) are used in the program, "OPERATION ERROR" (error code: 4140) can be detected if invalid floating-point data exist. This occurs even when interlocks are provided using the valid data flag (the signal which shows the floating-point validity).
Invalid floating-point data are not stored as the result of operations performed in the Universal model QCPU.
Reasons for those invalid data are considered as follows:

• The same device is used for storing floating-point data and other data, such as binary values, BCD values, and strings.
→ Use different devices for storing floating-point data and data other than floating-point data.
• Floating-point data externally written are invalid.
→ Take measures on the external-source side so that valid data are written.

If an error occurs in the floating-point data comparison instructions, take the above measures.
Example 1) Detecting "OPERATION ERROR" (error code: 4140) with the LDE☐ instruction

[Ladder mode]



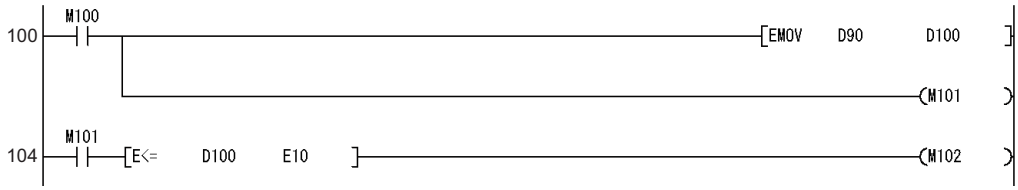
[List mode]

```
100  LD     M100
101  EMOV   D90    D100
103  OUT    M101
104  LD     M101
105  LDE<=  D100   E10
109  ORE>=  D100   E200
113  ANB
114  OUT    M102
```

In the ladder block starting from step 104, the floating-point data comparison instructions of steps 105 and 109 are not executed when M101 (valid data flag) is off.
However, the LDE<= instruction of step 105 and the ORE>= instruction of step 109 are executed regardless of the execution result of the LD instruction of step 104 in the program above. Therefore, even when M101 is off, "OPERATION ERROR" (error code: 4140) will be detected in the LDE<= instruction of step 105 if a 'special value' is stored in D100.
For the method of avoiding "OPERATION ERROR", refer to the following.
☞ Page 39 Method of avoiding "OPERATION ERROR" (error code: 4140)

Example 2) Not detecting "OPERATION ERROR" (error code: 4140) with the ANDE□ instruction

[Ladder mode]



[List mode]

```
100  LD      M100
101  EMOV    D90     D100
103  OUT     M101
104  LD      M101
105  ANDE<=  D100    E10
109  OUT     M102
```

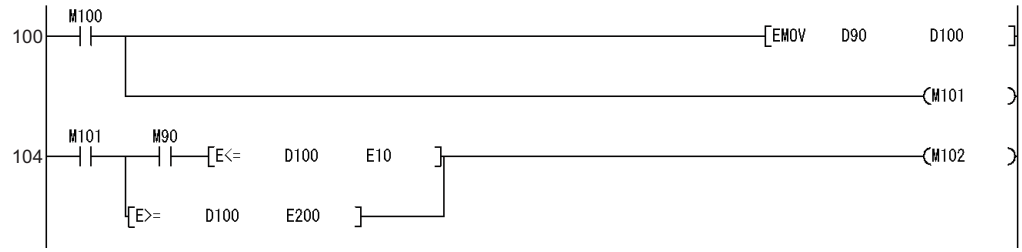
In the ladder block starting from step 104, the ANDE<= instruction of step 105 is not executed when M101 (valid data flag) is off.

The ANDE<= instruction of step 105 is not executed when M101 is off in the LD instruction of step 104 in the program above.

Therefore, when M101 is off, "OPERATION ERROR" (error code: 4140) will not be detected even if a 'special value' is stored in D100.

Example 3) Detecting "OPERATION ERROR" (error code: 4140) in the ANDE□ instruction

[Ladder mode]



[List mode]

```
100  LD      M100
101  EMOV    D90     D100
103  OUT     M101
104  LD      M101
105  LD      M90
106  ANDE<=  D100    E10
110  ORE>=   D100    E200
114  ANB
115  OUT     M102
```

In the ladder block starting from step 104, the ANDE<= instruction of step 106 and the ORE>= instruction of step 110 are not executed when M101 (valid data flag) is off.

However, if M90 is on in the LD instruction of step 105, the ANDE<= instruction of step 106 is executed.

Therefore, even when M101 is off, "OPERATION ERROR" (error code: 4140) will be detected in the ANDE<= instruction of step 106 if M90 is on and a 'special value' is stored in D100.

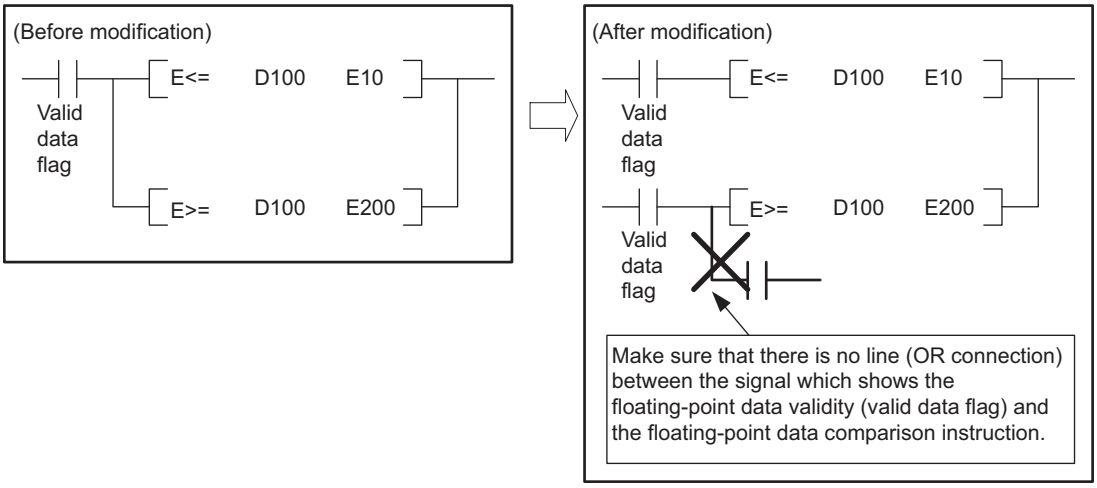
For the method of avoiding "OPERATION ERROR", refer to the following.

☞ Page 39 Method of avoiding "OPERATION ERROR" (error code: 4140)

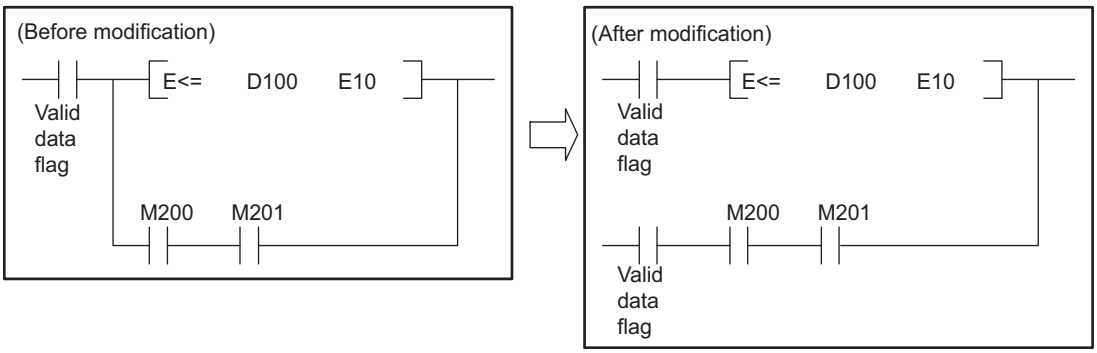
Method of avoiding "OPERATION ERROR" (error code: 4140)

As shown in the modification examples below, connect a valid data flag contact to a floating-point data comparison instruction is series. (Use the AND connection for connecting the contact of the valid data flag and floating-point data comparison instruction.)
Make sure that there is no vertical line (the OR connection) between the valid data flag and floating-point data comparison instruction.

<Modification example 1>



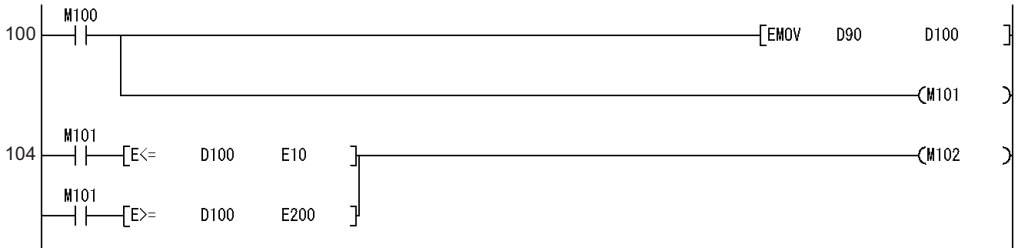
<Modification example 2>



Program examples corresponding to Examples 1) and 3) above are shown in Examples 4) and 5).

Example 4) Modified program (Example 1) ("OPERATION ERROR" (error code: 4140) is no longer detected.)

[Ladder mode]

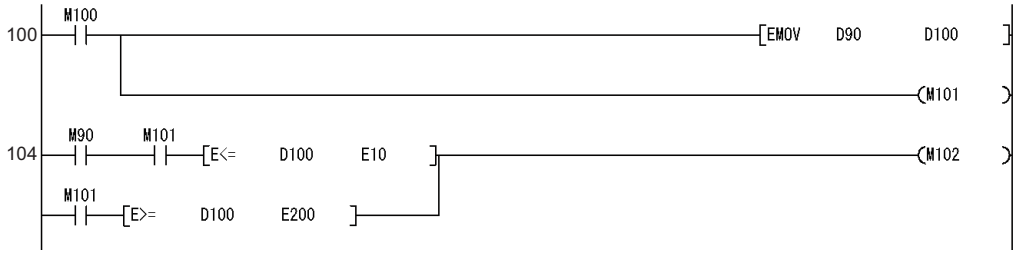


[List mode]

```
100  LD      M100
101  EMOV    D90     D100
103  OUT     M101
104  LD      M101
105  ANDE<=  D100    E10
109  LD      M101
110  ANDE>=  D100    E200
114  ORB
115  OUT     M102
```

Example 5) Modified program (Example 3) ("OPERATION ERROR" (error code: 4140) is no longer detected.)

[Ladder mode]



[List mode]

```
100  LD      M100
101  EMOV    D90     D100
103  OUT     M101
104  LD      M90
105  AND     M101
106  ANDE<=  D100    E10
110  LD      M101
111  ANDE>=  D100    E200
115  ORB
116  OUT     M102
```

5.3 Range Check Processing for Index-modified Devices

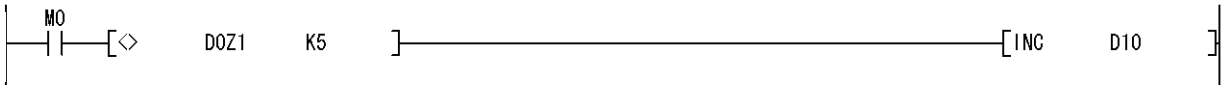
Device range check

Error check processing at index modification of devices has been enhanced for the Universal model QCPU.
Each index-modified device range is checked, and if the check target device is outside the device range before index modification, the CPU module detects "OPERATION ERROR" (error code: 4101).

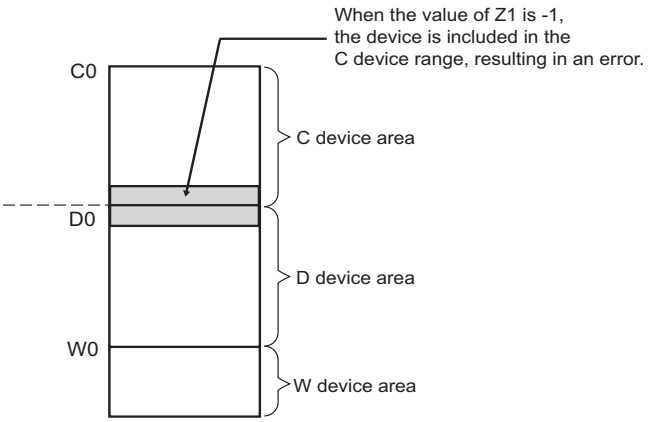
Point

For details on the index-modified device range check, refer to the following.
MELSEC-Q/L Programming Manual (Common Instruction)

Example 1) Detecting "OPERATION ERROR" (error code: 4101) by error check processing at index modification of devices



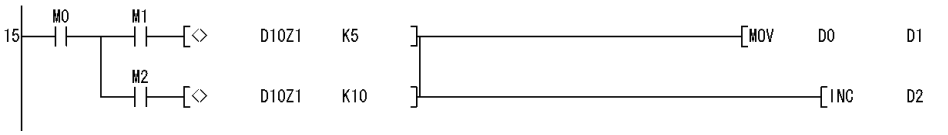
In Example 1), when the contact (M0) is on and the value, -1 or less, is specified in Z1, the device D0Z1 is included in the C device range, exceeding the D device range, as shown in the following figure. As a result, "OPERATION ERROR" (error code: 4101) will be detected.



When an error is detected, check the index modification value (value of Z1 in the above example) and remove the error cause.
Examples of the cases where an error is detected and not detected are shown below.

Example 2) Detecting "OPERATION ERROR" (error code: 4101)

[Ladder mode]

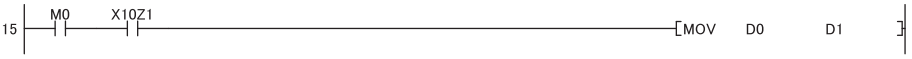


[List mode]

```
15  LD      M0
16  LD      M1
17  AND<>   D10Z1   K5
20  LD      M2
21  AND<>   D10Z1   K10
24  ORB
25  ANB
26  MOV     D0      D1
28  INC     D2
```

In Example 2, in the ladder block starting from the step 15, the AND < > instruction of the step 17 or 21 is supposed to be not executed when M0 (valid data flag) is off.
However, since the LD instruction which is always executed is used in the step 16 and 20, the AND < > instruction of the step 17 or 21 is executed regardless of the execution status of the LD instruction in the step 15 when M1 or M2 is on.
For this reason, even when M0 is off, if the D10Z1 value is outside the D device range, "OPERATION ERROR" (error code: 4101) will be detected in the AND < > instruction of the step 17.
Note that the step 26 (MOV D0 D1) and the step 28 (INC D2) are not executed.
For the method of avoiding "OPERATION ERROR" (error code: 4101), refer to the following.
☞ Page 43 Actions taken to avoid "OPERATION ERROR" (error code: 4101)

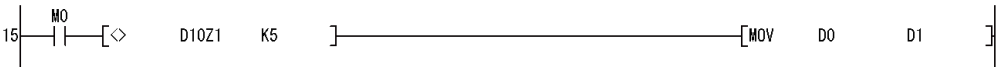
Example 3) Detecting "OPERATION ERROR" (error code: 4101)



In Example 3, even when M0 (valid data flag) in the step 15 is off, the AND instruction in the next step (step 16) will be executed. For this reason, if the X10Z1 value is outside the X device range,
"OPERATION ERROR" (error code: 4101) will be detected in the AND instruction of the step 16.
For the actions to be taken to avoid "OPERATION ERROR" (error code: 4101), refer to the following.
☞ Page 43 Actions taken to avoid "OPERATION ERROR" (error code: 4101)

Example 4) Not detecting "OPERATION ERROR" (error code: 4101)

[Ladder mode]



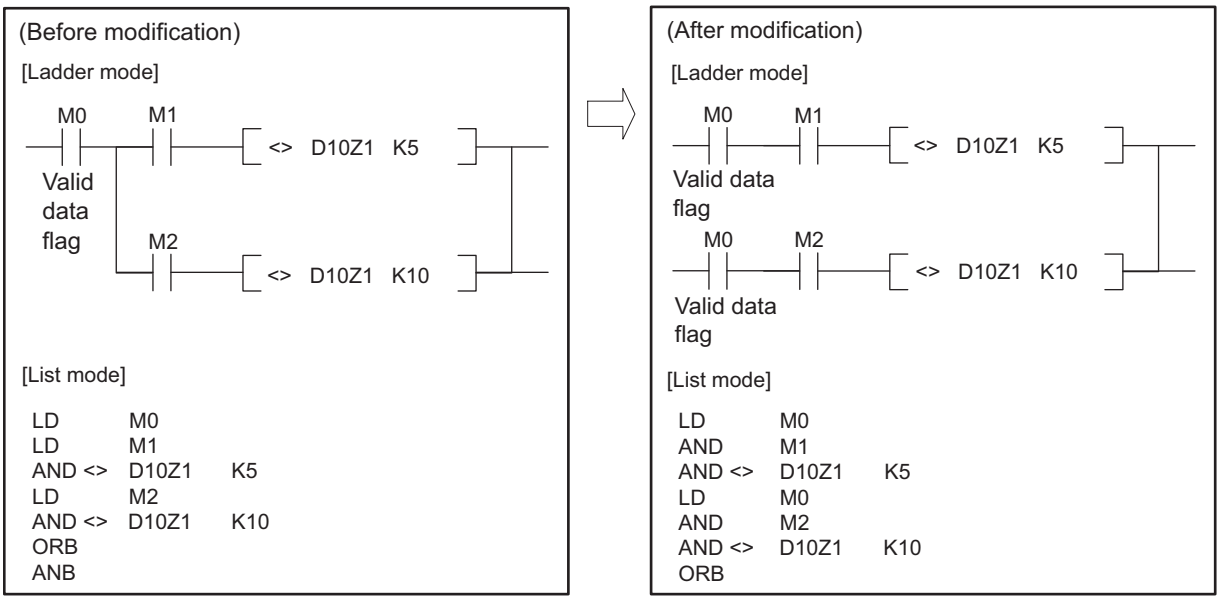
[List mode]

```
15  LD      M0
16  AND<>   D10Z1   K5
19  MOV     D0      D1
```

In Example 4, the AND < > instruction of the step 16 is not executed when M0 (valid data flag) of the step 15 is off.
For this reason, "OPERATION ERROR" (error code: 4101) will not be detected no matter what the D10Z1 value is.

Actions taken to avoid "OPERATION ERROR" (error code: 4101)

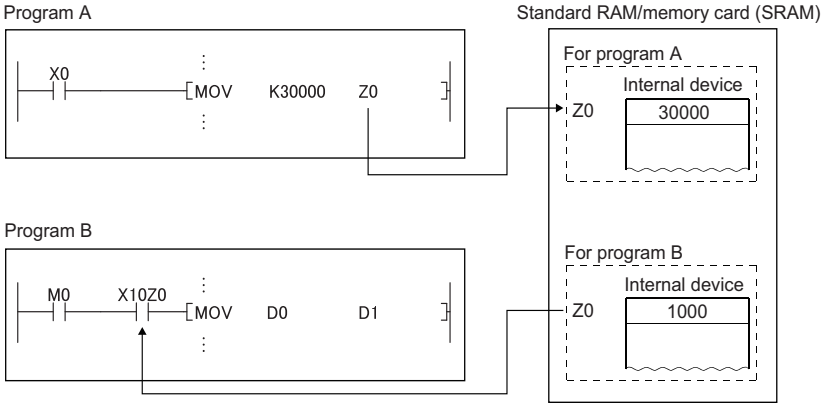
If the index-modified device range does not need to be checked, set the parameter as described in ❶ below.
If the index-modified device range needs to be checked, but the detection of errors shown in Examples 2) and 3) above should be avoided, take actions described in ❷ to ❹.
☞ Page 41 Device range check
❶ Deselect the "Check device range at indexing." item in the PLC RAS tab of the PLC parameter dialog box so that the index-modified device range will not be checked.
❷ As shown in the modification examples below, connect the contacts of valid data flag in series for each instruction that checks the index-modified device range. (This will not apply to the High-speed Universal model QCPU.) (This will not apply to the High-speed Universal model QCPU.)
<Modification example>



In the program before modification (on the left), the instruction immediately before the AND < > instruction is regarded as the LD instruction. However, in the program after modification (on the right), the same instruction will be regarded as the AND instruction.
In the program after modification, only when both contacts of M0 and M1 (or M2) turn on, the AND < > instruction is executed. As a result, no error will be detected during index-modified device range check processing.
❸ Use the index register as a local device.
With a project where multiple programs are executed, if the program where "OPERATION ERROR" (error code: 4101) is detected is executed alone and no error occurs, use the index register as a local device.
This enables the index register to be used independently by each program. Even if another program overwrites the index register with a "value that causes the index-modified device to be outside the device range", it will not affect the value of the index register used in the program where the error occurs. As a result, "OPERATION ERROR" (error code: 4101) will not be detected.
Note that the scan time increases because the time for saving and restoring the local device file increases. For the local device settings, refer to the following.
📖 QnUCPU User's Manual (Function Explanation, Program Fundamentals)

Ex.

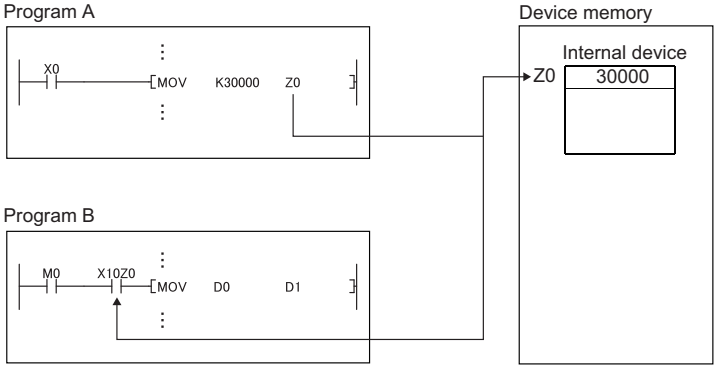
When the index register is used as a local device



Even when program A overwrites the index register Z0 with a value of 30000, no change is made to the index register Z0 used by program B. No error occurs as long as X10Z0 does not exceed the X device range.

Ex.

When the index register is not used as a local device

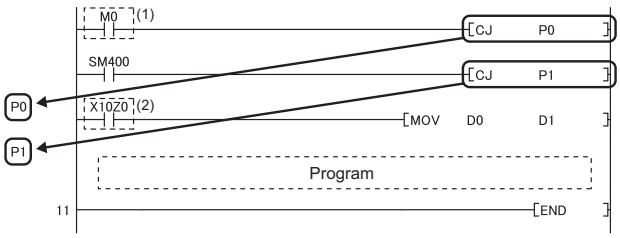


When program A overwrites the index register Z0 with a value of 30000, the value of the index register Z0 used by program B is also changed. An error occurs when X10Z0 exceeds the X device range.

❹ Use the CJ instruction.

When the CJ instruction is used as shown below and the previous condition ("LD M0" in the figure (1) below) is off, avoid the execution of a contact instruction that uses the index register ("LD X10Z0" in the figure (2) below). When the condition in the figure (1) below is off, the instruction in the figure (2) below is not executed and the value of the device used as a contact is not read. Thus, the device range check processing does not detect "OPERATION ERROR" (error code: 4101).

Note that the use of the CJ instruction increases the scan time.



5.4 Device Latch Function

Overview

The device latch function[*1] for the Universal model QCPU is more enhanced compared to that for the High performance model QCPU.

This section describes the enhanced device latch function in the Universal model QCPU.

*1 The latch function is used to hold device data even when the CPU module is powered off or reset.

Device data latch methods

Device data of the Universal model QCPU can be latched by:

- using the large-capacity file register (R, ZR),[*1]
- writing/reading device data to/from the standard ROM (with the SP.DEVST and S(P).DEVLD instructions),
- specifying a latch range of internal user devices, or
- setting intervals ("Time Setting") in the latch interval setting parameter.[*2]

*1 The extended data register (D) and extended link register (W) are included.

*2 Only the High-speed Universal model QCPU supports the setting.

Details of each latch method

■Large-capacity file register (R, ZR)

The file register is the device that can be latched by using the battery.

File register size is larger and processing speed is higher in the Universal model QCPU, compared to the High Performance model QCPU.

To latch a lot of data (many device points), use of a file register is effective.

The following table lists the allowable file register size in each CPU module.

- File register size

CPU module		File register (R, ZR) size in the standard RAM
Q02UCPU		64K points
Q03UD(E)CPU		96K points
Q03UDVCPU	Without an extended SRAM cassette	96K points
	With an extended SRAM cassette (1M)	608K points
	With an extended SRAM cassette (2M)	1120K points
	With an extended SRAM cassette (4M)	2144K points
	With an extended SRAM cassette (8M)	4192K points
Q04UD(E)CPU		128K points
Q04UDVCPU	Without an extended SRAM cassette	128K points
	With an extended SRAM cassette (1M)	640K points
	With an extended SRAM cassette (2M)	1152K points
	With an extended SRAM cassette (4M)	2176K points
	With an extended SRAM cassette (8M)	4224K points
Q06UD(E)CPU		384K points
Q06UDVCPU	Without an extended SRAM cassette	384K points
	With an extended SRAM cassette (1M)	896K points
	With an extended SRAM cassette (2M)	1408K points
	With an extended SRAM cassette (4M)	2432K points
	With an extended SRAM cassette (8M)	4480K points
Q10UD(E)HCPU, Q13UD(E)HCPU		512K points
Q13UDVCPU	Without an extended SRAM cassette	512K points
	With an extended SRAM cassette (1M)	1024K points
	With an extended SRAM cassette (2M)	1536K points
	With an extended SRAM cassette (4M)	2560K points
	With an extended SRAM cassette (8M)	4608K points

CPU module		File register (R, ZR) size in the standard RAM
Q20UD(E)HCPU, Q26UD(E)HCPU		640K points
Q26UDVCPU	Without an extended SRAM cassette	640K points
	With an extended SRAM cassette (1M)	1152K points
	With an extended SRAM cassette (2M)	1664K points
	With an extended SRAM cassette (4M)	2688K points
	With an extended SRAM cassette (8M)	4736K points
Q50UDEHCPU		768K points
Q100UDEHCPU		896K points

■Writing/reading device data to/from the standard ROM (SP.DEVST/S(P).DEVLD instructions)

Device data of the Universal model QCPU can be latched using the SP.DEVST and S (P).DEVLD instructions (instructions for writing/reading data to/from the standard ROM).

Utilizing the standard ROM allows data backup without batteries. This method is effective for latching data that will be updated less frequently.

■Specifying the latch range of internal user devices

Device data of the Universal model QCPU can be latched by specifying a latch range of internal user devices in the same way as for the High Performance model QCPU.

The ranges can be set in the Device tab of the PLC parameter dialog box.

Internal user devices that can be latched are as follows:

- Latch relay (L)
- Link relay (B)
- Annunciator (F)
- Edge relay (V)
- Timer (T)
- Retentive timer (ST)
- Counter (C)
- Data register (D)
- Link register (W)

Point

- If latch ranges of internal user devices are specified in the Universal model QCPU, the processing time will increase in proportion to the points of the device to be latched. (For example, if 8K points are latched for the latch relay (L) with the QnUD(E)(H)CPU, the processing time is 28.6μs.) To shorten the scan time, remove unnecessary latch device points to minimize the latch range.
- The scan time will not increase when a latch range of the file register (R, ZR) is specified.

How to shorten the scan time

When data to be latched are stored in a file register (R or ZR), the processing time is shorter than that for latching internal user device.

Ex.

Reducing the latch points of the data register (D) from 8K points to 2K points, and using the file register (ZR) instead (when the Q06UDVCPU is used).

■Differences between before and after moving latch points of the data register (D) to the file register (ZR)

Item		Before	After
Latch points for data register (D)		8192 (8K) points	2048 (2K) points (6K points are moved to file register.)
Number of devices in the program	Data register (D) (Latch range)	400	100
	File register (ZR) (Standard RAM)	0	300
Additional scan time		0.37ms	0.11ms*1
Number of steps increased		—	300 steps

*1 Indicates the time required additionally when file register data are stored in the standard RAM.

Point

The High-speed Universal model QCPU can choose a latch interval setting between "Each Scan" and "Time Setting" in parameter. When "Time Setting" is selected, latch data processing starts during the first END processing after a preset time has elapsed. Since the latch data processing is performed asynchronous to the sequence program, an increase in scan time is reduced.

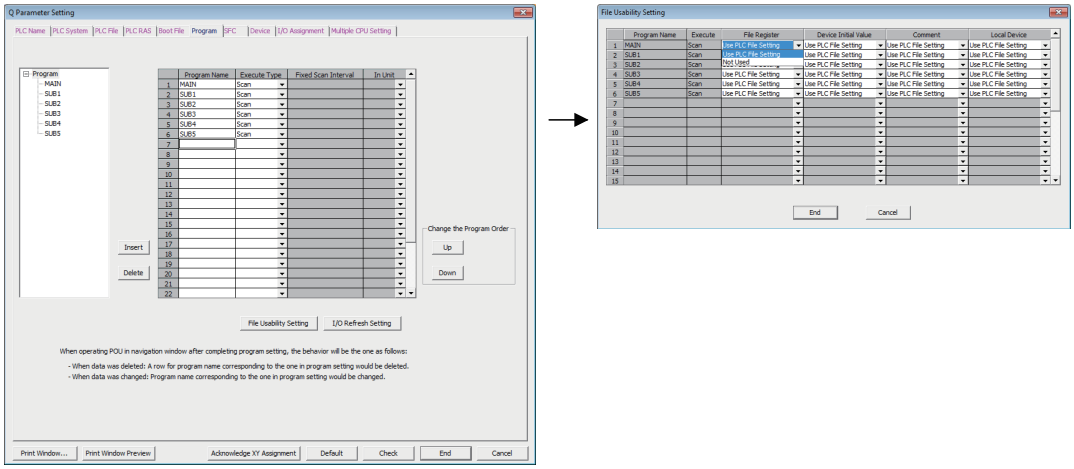
5.5 File Usability Setting

Differences between the High Performance model QCPU and Universal model QCPU

■High Performance model QCPU

In the High Performance model QCPU, file usability ("Use PLC file setting" or "Not used") of the following files can be set for each program on the window opened by clicking the "File usability setting" button on the Program tab of the PLC parameter dialog box.

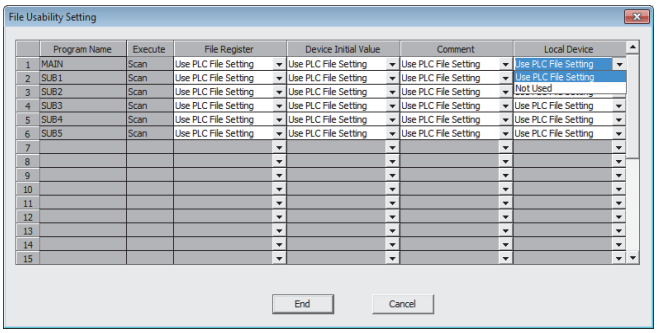
- File register
- Device initial value
- Comment
- Local device



■Universal model QCPU

In the Universal model QCPU, file usability of the following files[*1] cannot be set for each program on the window opened by clicking the "File usability setting" button on the Program tab of the PLC parameter dialog box.

- File register
- Initial device value
- Comment



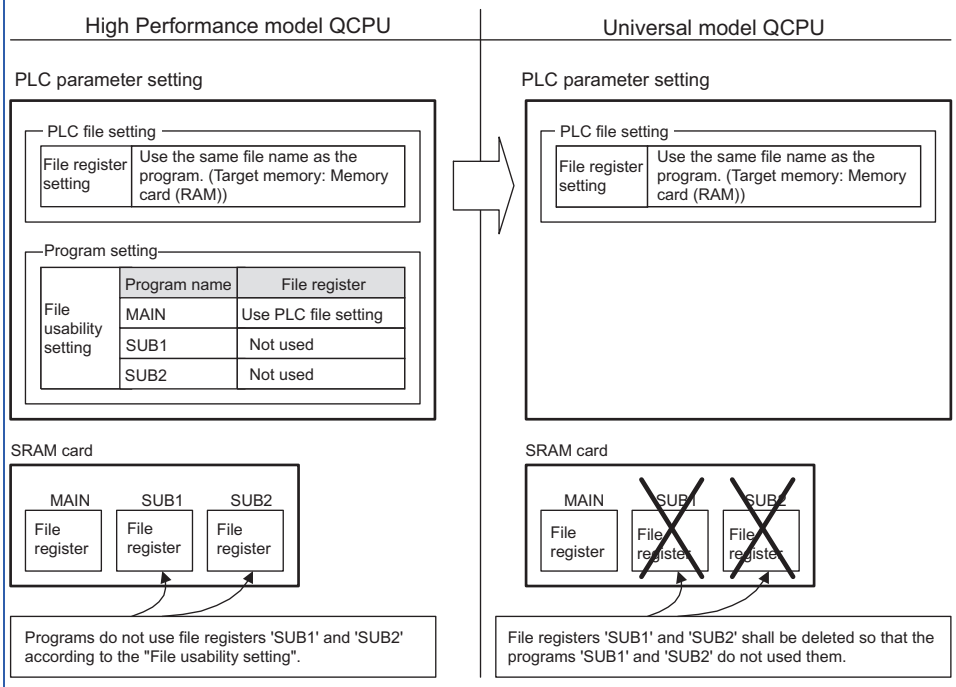
*1 The local device file usability setting is also not available for the Universal model QCPU if the serial number (first five digits) is "10011" or earlier.
If the local device is set to be used in the PLC file tab of the PLC parameter dialog box in the High Performance model QCPU, all the programs use the local device in the Universal model QCPU after replacement.

When the file usability setting is set in the High Performance model QCPU, change the setting in the following pages.

Method of replacing High Performance model QCPU with Universal model QCPU

Replacement method varies depending on the settings in the PLC file tab of the PLC parameter dialog box.

■Replacement method

Setting in the PLC file tab	Setting in Universal model QCPU
"Not used." is selected.	No change in parameter setting is required. Operation of the Universal model QCPU is the same regardless of the file usability setting in the High Performance model QCPU.
"Use the same file name as the program." is selected.	When file usability is set to "Not used." in the High Performance model QCPU, delete the corresponding program file (file register, device initial value or comment), which uses the same name as the program, from the target memory. The Universal model QCPU executes a program without using a program file if no program file that uses the same name as the program exists in the target memory. 
"Use the following file." is selected.	No change in parameter setting is required. Operation of the Universal model QCPU is the same regardless of the file usability setting in the High Performance model QCPU.

5.6 Parameter-valid Drive and Boot File Setting

Differences between the High Performance model QCPU and Universal model QCPU

■High Performance model QCPU

The parameter-valid drive is specified at the switches on the front panel of the High Performance model QCPU.

■Universal model QCPU

The Universal model QCPU automatically determines the parameter-valid drive, depending on the existence of parameters in the drive (program memory, memory card, SD memory card, or standard ROM). Therefore, when replacing the High Performance model QCPU with the Universal model QCPU, changing the boot file setting for parameter and/or moving files to another drive may be required.

When replacing the module, change the setting as follows.

Replacing High Performance model QCPU with Universal model QCPU

■When the parameter-valid drive is set to the standard ROM in the High Performance model QCPU

• When the parameter-valid drive is set to the standard ROM

Setting in High Performance model QCPU Setting in the Boot file tab of the PLC parameter dialog box	Setting in Universal model QCPU
No boot file setting	Change the setting so that the Universal model QCPU can refer to the parameters in the standard ROM • Changes in parameter setting are not required. • Delete parameters that exist in the program memory, memory card, and/or SD memory card.*1
Settings in the Boot file tab (No boot file setting for parameters) • Type: Program • Transfer from: Standard ROM • Transfer to: Program memory	Change the setting so that programs are stored in the program memory in the first place, instead of booting from the standard ROM. • Delete all settings for parameter in the Boot file tab of the PLC parameter dialog box. • Delete parameters that exist in the program memory, memory card, and/or SD memory card.*2 • Move the programs with boot setting into the program memory from the standard ROM.*1
Settings in the Boot file tab • Type: Program • Transfer from: Standard ROM • Transfer to: Program memory Or • Type: Parameter • Transfer from: Standard ROM • Transfer to: Program memory	Change the setting so that programs and parameters are stored in the program memory in the first place, instead of booting from the standard ROM. • Move the programs and parameters with boot setting into the program memory from the standard ROM.*1 • Delete all settings for parameter in the Boot file tab of the PLC parameter dialog box.
Settings in the Boot file tab (No boot file setting for parameters) • Type: Program • Transfer from: Memory card • Transfer to: Program memory	Change the setting so that the Universal model QCPU can refer to the parameters in the memory card or SD memory card, and programs are booted from the card to the program memory. • Move the parameters in the standard ROM into the memory card or SD memory card. • Make setting so that programs are booted from the memory card or SD memory card to the program memory in the Boot file tab of the PLC parameter dialog box.*3
Settings in the Boot file tab • Type: Program • Transfer from: Standard ROM • Transfer to: Program memory Or • Type: Parameter • Transfer from: Standard ROM • Transfer to: Program memory	Change the setting so that the Universal model QCPU can refer to the parameters in the memory card or SD memory card, and programs and parameters are booted from the card to the program memory. • Move the parameters in the standard ROM into the memory card or SD memory card. • Make setting so that programs and parameters are booted from the memory card or SD memory card to the program memory in the Boot file tab of the PLC parameter dialog box.*3

Setting in High Performance model QCPU Setting in the Boot file tab of the PLC parameter dialog box	Setting in Universal model QCPU
Settings in the Boot file tab • Type: Data other than program and parameter • Transfer from: Memory card • Transfer to: Program memory Or • Type: Data other than program and parameter • Transfer from: Standard ROM • Transfer to: Program memory (Data other than program and parameter indicate initial device value, device comment, and label program.)	Delete all settings for data other than program and parameter in the boot file setting. Since these data can be used even not stored in the program memory, it is not necessary to transfer them to the program memory. Or, change the setting so that they are stored in the program memory in the first place. • Delete all settings for data other than program and parameter in the Boot file tab of the PLC parameter dialog box. • Move the data other than programs and parameters into the program memory as needed.

*1 Since the Universal model QCPU holds the data in the program memory even when the battery voltage drops, the boot file setting is not necessary.

*2 The Universal model QCPU searches for parameters in order of in the program memory, in the memory card or SD memory card, and in the standard ROM. Then, the module uses the parameters found first. If parameters exist in the program memory, memory card, or SD memory card, the Universal model QCPU does not use the parameters in the standard ROM.

*3 The Universal model QCPU ignores the boot file setting for parameters in the standard ROM.

■When the parameter-valid drive is set to the memory card (RAM) or memory card (ROM) in the High Performance model QCPU

• When the parameter-valid drive is set to the memory card (RAM) or memory card (ROM)

Setting in High Performance model QCPU Setting in the Boot file tab of the PLC parameter dialog box	Setting in Universal model QCPU
No boot file setting	Change the setting so that the Universal model QCPU can refer to the parameters in the memory card or SD memory card. • Changes in parameter setting are not required. • Delete parameters that exist in the program memory.*2
Settings in the Boot file tab (No boot file setting for parameters) • Type: Program • Transfer from: Memory card • Transfer to: Program memory	Change the setting so that the Universal model QCPU can refer to the parameters in the memory card or SD memory card. • Changes in parameter setting are not required. • Delete parameters that exist in the program memory.*2
Settings in the Boot file tab • Type: Program • Transfer from: Standard ROM • Transfer to: Memory card Or • Type: Parameter • Transfer from: Memory card • Transfer to: Program memory	No changes are required.
Settings in the Boot file tab (No boot file setting for parameters) • Type: Program • Transfer from: Standard ROM • Transfer to: Program memory	Change the setting so that programs are stored in the program memory in the first place, instead of booting from the standard ROM. • Move the programs targeted for booting from the standard ROM into the program memory.*1 • Delete all settings for program in the Boot file tab of the PLC parameter dialog box. • Delete parameters that exist in the program memory.*2
Settings in the Boot file tab • Type: Program • Transfer from: Standard ROM • Transfer to: Program memory Or • Type: Parameter • Transfer from: Memory card • Transfer to: Program memory	Change the setting so that programs are stored in the program memory in the first place, instead of booting from the standard ROM. • Move the programs targeted for booting from the standard ROM into the program memory.*1 • Delete all settings for program in the Boot file tab of the PLC parameter dialog box.

Setting in High Performance model QCPU Setting in the Boot file tab of the PLC parameter dialog box	Setting in Universal model QCPU
Settings in the Boot file tab • Type: Data other than program and parameter • Transfer from: Memory card • Transfer to: Program memory Or • Type: Data other than program and parameter • Transfer from: Standard ROM • Transfer to: Program memory (Data other than program and parameter indicate initial device value, device comment, and label program.)	Delete all settings for data other than program and parameter in the boot file setting. Since these data can be used even not stored in the program memory, it is not necessary to transfer them to the program memory. Or, change the setting so that they are stored in the program memory in the first place. • Delete all settings for data other than program and parameter in the Boot file tab of the PLC parameter dialog box. • Move the data other than program and parameter into the program memory as needed.

*1 Since the Universal model QCPU holds the data in the program memory even when the battery voltage drops, the boot file setting is not necessary.

*2 The Universal model QCPU searches for parameters in order of in the program memory, in the memory card or SD memory card, and in the standard ROM. Then, the module uses the parameters found first. If parameters exist in the program memory, memory card, or SD memory card, the Universal model QCPU does not use the parameters in the standard ROM.

5.7 External Input/Output Forced On/Off Function

Differences between the High Performance model QCPU and Universal model QCPU

■High Performance model QCPU

External input/output can be forcibly turned on/off on the window opened by selecting [Online] ⇨ [Debug] ⇨ [Forced input output registration/cancellation] in the programming tool.

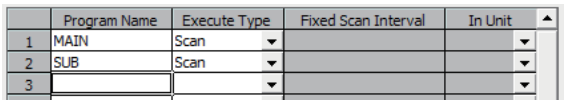
■Universal model QCPU

If the serial number (first five digits) is "10041" or earlier, the external input/output forced on/off function cannot be used.
External input/output can be forcibly turned on/off by using the replacement program described below.

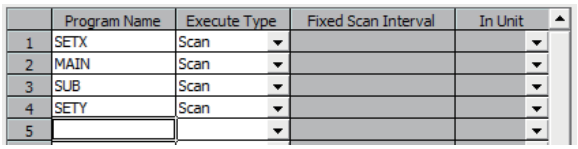
Method of replacing High Performance model QCPU with Universal model QCPU

As shown in the following figure, add program names, "SETX" and "SETY", in the Program tab of the PLC parameter dialog box.

<Before replacement>



<After replacement>



The following table shows the program setting of the "SETX" and "SETY".

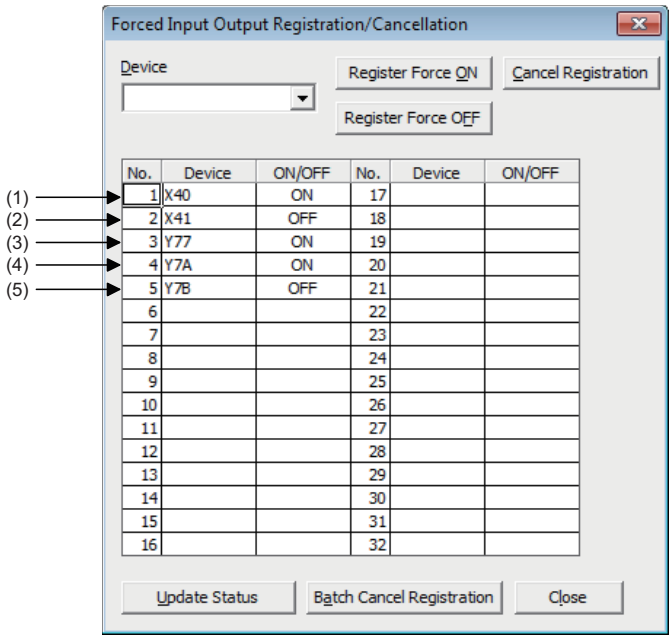
Program name	Execution type	Position where program is added
SETX	Scan	Start of Program setting (No.1)
SETY	Scan	End of Program setting

Ex.

Forcibly turning X40, X77, and X7A on, and X41 and Y7B off

The programs, "SETX" and "SETY", turns on or off the X and Y devices, which have been registered for forced on/off using the external input/output forced on/off function, at each scan using the SET and RST instructions.

High Performance model QCPU



Universal model QCPU

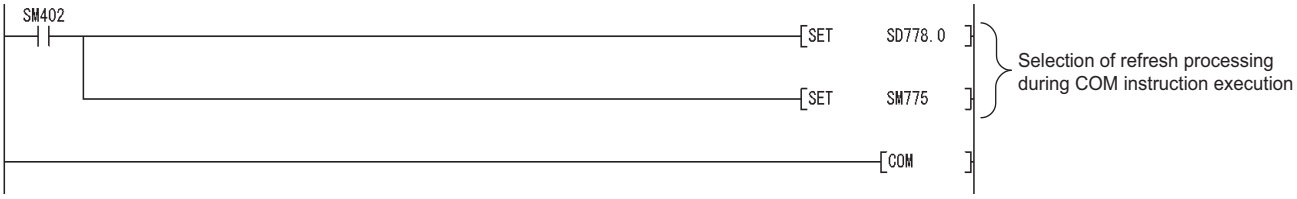
• Program example of "SETX"

• Program example of "SETY"

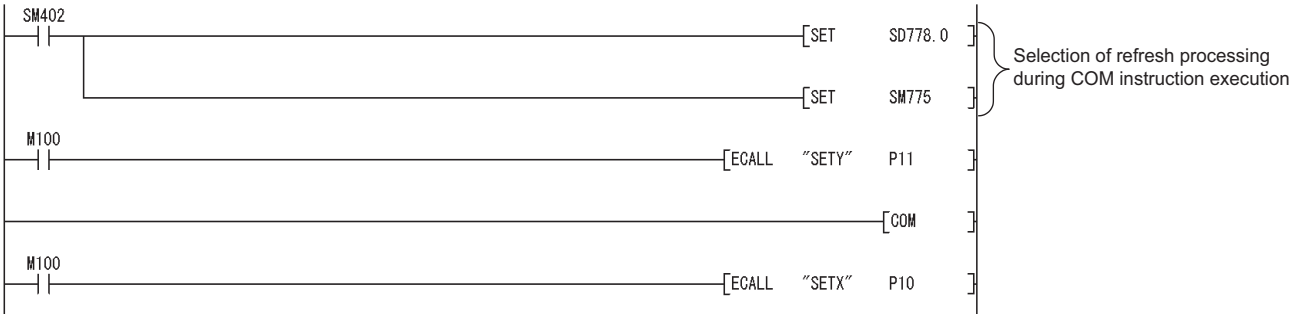
Replacing the COM instruction

If the COM instruction is used, add subroutine calls for P10 and P11 before and after the COM instruction. P10 and P11 are pointers shown in the previous program examples. When SM775 is on (Executes refresh set by SD778) and also the 0 bit of SD778 is off (Do not execute I/O refresh), replacement of the instruction is not necessary.

■Program before replacement



■Program after replacement



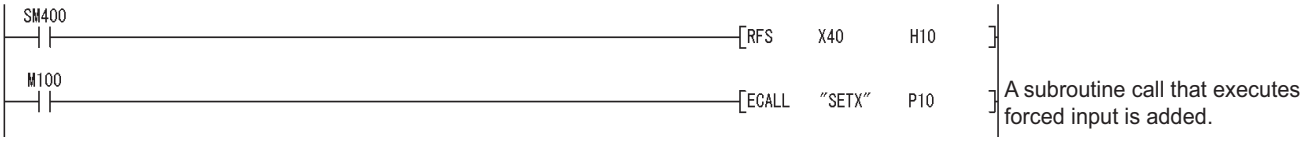
Replacing the RFS instruction

If any I/O numbers targeted for forced on/off are included in the partial refresh range specified by the RFS instruction, add subroutine calls for P10 and P11 before and after the RFS instruction. P10 and P11 are pointers shown in the previous program examples.

If no I/O number targeted for forced on/off is included, addition of subroutine calls for P10 and P11 is not necessary.

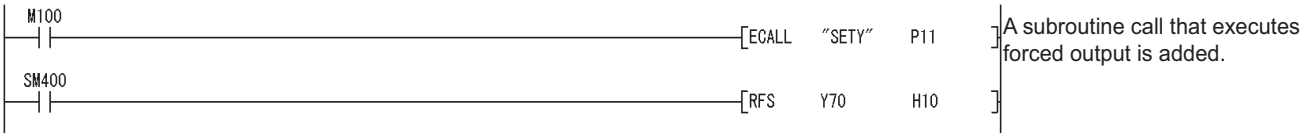
■When partial refresh for input (X) is executed by the RFS instruction

Add a subroutine call that executes forced input after the RFS instruction.



■When partial refresh for output (Y) is executed by the RFS instruction

Add a subroutine call that executes forced output before the RFS instruction.



Restrictions

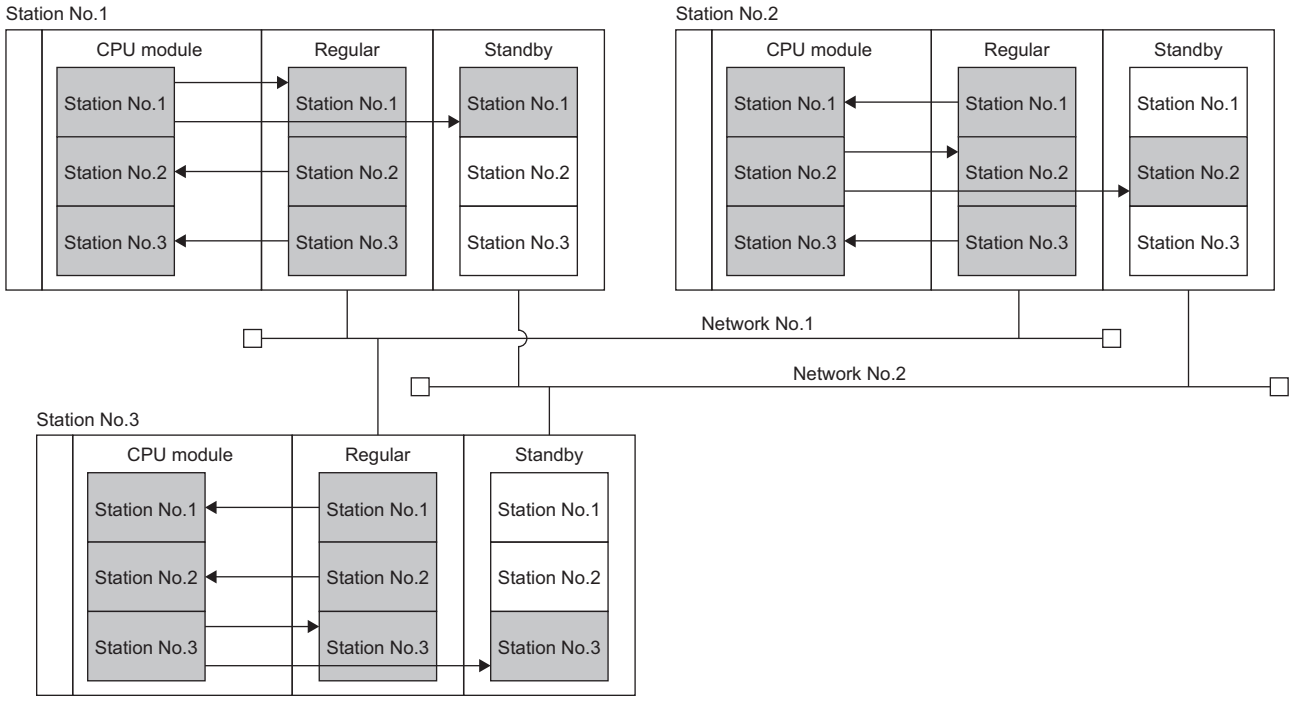
Replacement methods described in this section do not apply in the following cases:

- Input and output targeted for forced on/off are referred to or changed using the direct input device (DX)/direct output device (DY).
- Input and output targeted for forced on/off are referred to or changed within an interrupt program.

5.8 Alternative Methods for the Simple Dual-structured Network of MELSECNET/H

An example of alternative to the simple dual-structured network for MELSECNET/H is as shown below.

System configuration



Network parameter

Set the network range assignment as below.

Station No.	Send Range for each Station			Send Range for each Station		
	LB			LW		
	Points	Start	End	Points	Start	End
1	16	0000	000F	16	0000	000F
2	16	0010	001F	16	0010	001F
3	16	0020	002F	16	0020	002F

Program for the simple dual-structured system

(1) Check the cyclic transmission status when the regular network is normal.
(2) Check the cyclic transmission status when the standby network is normal.
(3) Switch the network to the regular network when the standby network is faulty. (The regular network is forcibly used in the first scan after RUN.)
(4) Switch the network to the standby network when the regular network is faulty.

The following table lists the refresh setting device (SM) for each network.

Item	Module 1	Module 2	Module 3	Module 4
Distinction between regular/standby network (Off: Regular network, On: Standby network)	SM255	SM260	SM265	SM270
Refresh from the network modules to the CPU (Off: Refreshes, On: Does not refresh)	SM256	SM261	SM266	SM271
Refresh from the CPU to the network modules (Off: Refreshes, On: Does not refresh)	SM257	SM262	SM267	SM272

For details, refer to Section 7.7 in the following.

📖 Q Corresponding MELSECNET/H Network System Reference Manual (PLC to PLC network)

Program for the alternative system

As an alternative to the special relay for the simple dual-structured network of MELSECNET/H, write programs to detect each network error and perform refresh by using instructions that use link direct devices (J□\B□, J□\W□) instead of setting refresh parameters of the network.

The station-based block data assurance for cyclic data does not support the refresh performed by instructions. If this assurance is used in the simple dual-structured system, interlock programs for each link data are required. For details on the interlock programs, refer to Section 6.2.3 "Interlock program example" in the following.

📖 Q Corresponding MELSECNET/H Network System Reference Manual (PLC to PLC network)

■Refresh parameter

Delete refresh parameters except those related to the link special relay (SB)/link special register (SW).

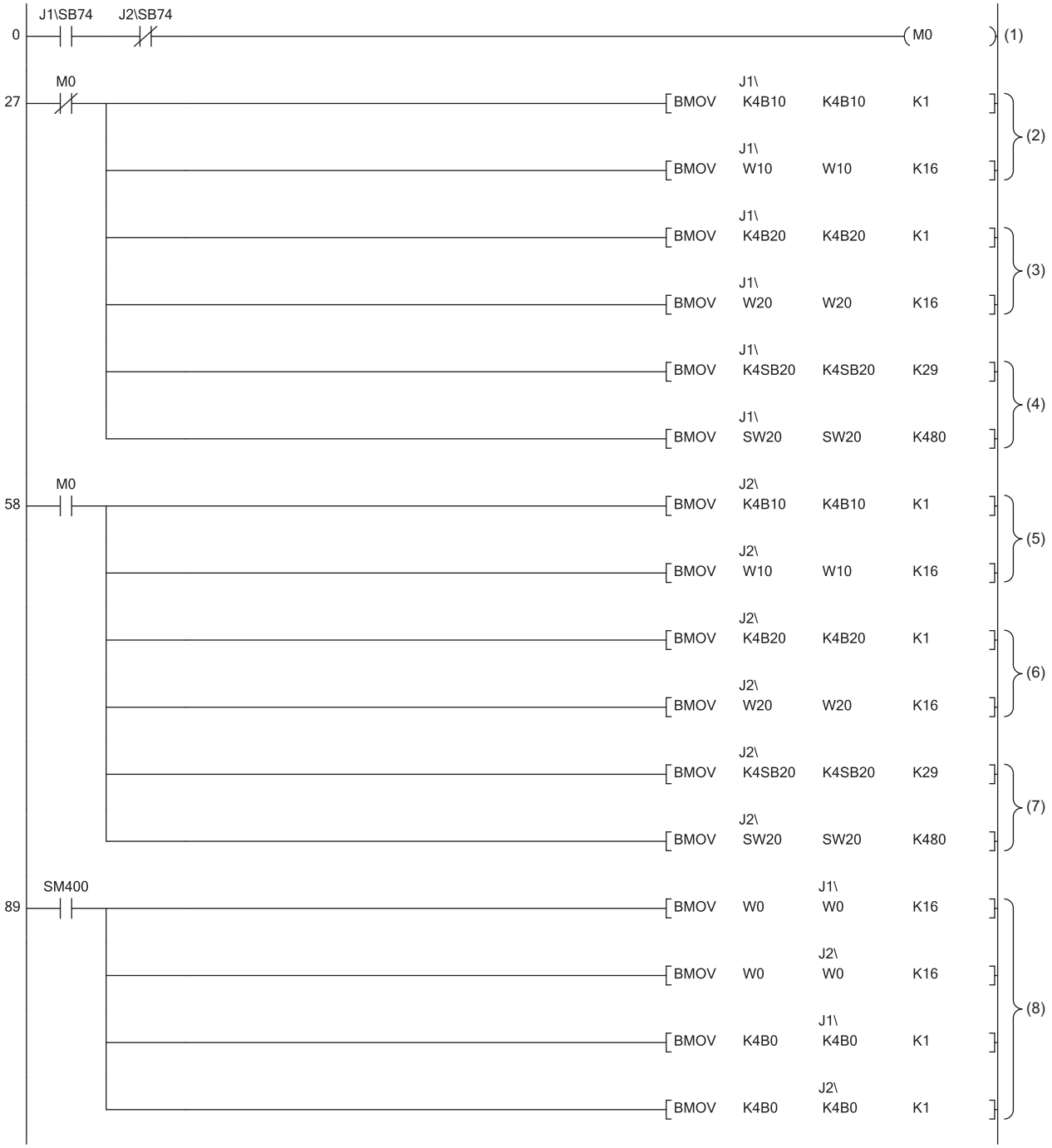
	Link Side					PLC Side			
	Dev. Name	Points	Start	End		Dev. Name	Points	Start	End
Transfer SB	SB	32	0000	001F	⇔	SB	32	0000	001F
Transfer SW	SW	32	0000	001F	⇔	SW	32	0000	001F
Random Cyclic	LB				⇔				
Random Cyclic	LW				⇔				
Transfer 1					⇔				
Transfer 2					⇔				

- Set the refresh parameters only for SB0 to SB1F and SW0 to SW1F (range of the CPU to the networks).
- Delete the refresh parameters except for SB/SW.

■Program

The following figure shows programs for each station number.

• Program for station number 1



(1) Switch the target of refresh from the network to the CPU to the standby network (M0: On) when the regular network is faulty and the standby network is normal. In other cases, switch the target to the regular network (M0: Off).

(2) Load the device data from the send area of the station number 2 on the regular network to the device memory in the CPU module of the station number 1.

(3) Load the device data from the send area of the station number 3 on the regular network to the device memory in the CPU module of the station number 1.

(4) Load the device data from the area SB20 to SB1FF and SW20 to SW1FF on the regular network to the area SB20 to SB1FF and SW20 to SW1FF of the station number 1.

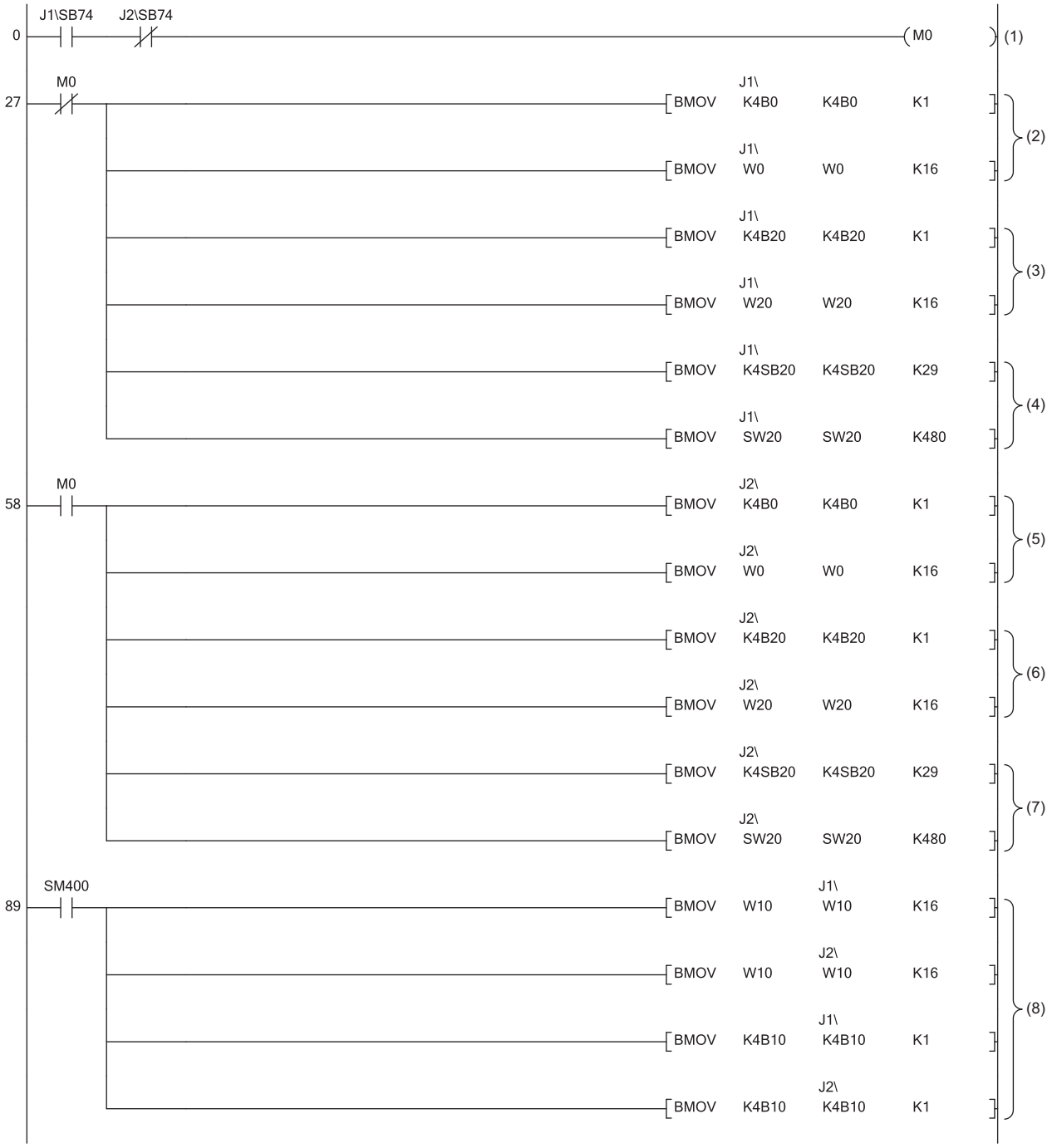
(5) Load the device data from the send area of the station number 2 on the standby network to the device memory in the CPU module of the station number 1.

(6) Load the device data from the send area of the station number 3 on the standby network to the device memory in the CPU module of the station number 1.

(7) Load the device data from the area SB20 to SB1FF and SW20 to SW1FF on the standby network to the area SB20 to SB1FF and SW20 to SW1FF of the station number 1.

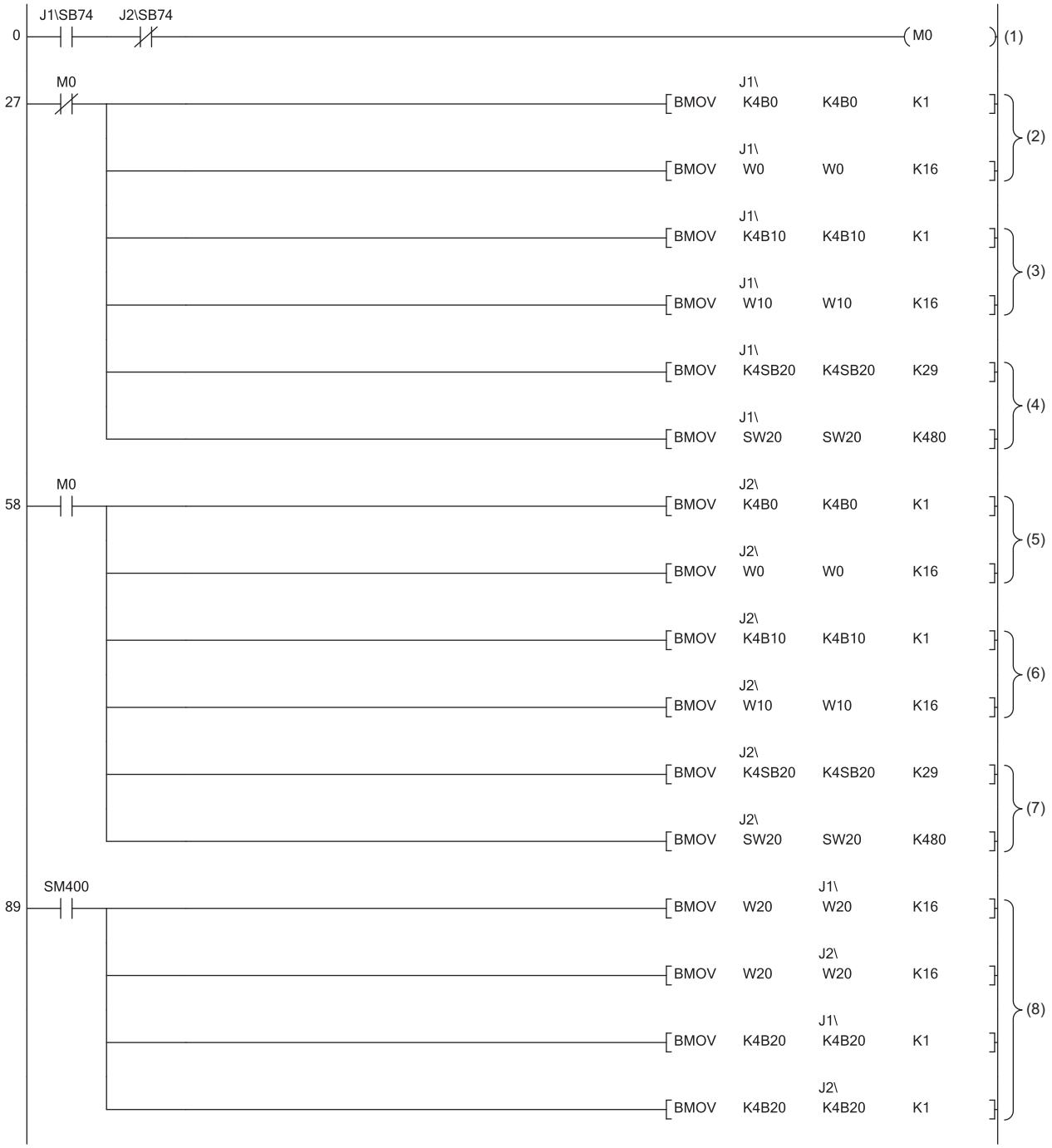
(8) Write the send data to the send area of the station number 1 on the control network/standby network.

• Program for station number 2



(1) Switch the target of refresh from the network to the CPU to the standby network (M0: On) when the regular network is faulty and the standby network is normal. In other cases, switch the target to the regular network (M0: Off).
(2) Load the device data from the send area of the station number 1 on the regular network to the device memory in the CPU module of the station number 2.
(3) Load the device data from the send area of the station number 3 on the regular network to the device memory in the CPU module of the station number 2.
(4) Load the device data from the area SB20 to SB1FF and SW20 to SW1FF on the regular network to the area SB20 to SB1FF and SW20 to SW1FF of the station number 2.
(5) Load the device data from the send area of the station number 1 on the standby network to the device memory in the CPU module of the station number 2.
(6) Load the device data from the send area of the station number 3 on the standby network to the device memory in the CPU module of the station number 2.
(7) Load the device data from the area SB20 to SB1FF and SW20 to SW1FF on the standby network to the area SB20 to SB1FF and SW20 to SW1FF of the station number 2.
(8) Write the send data to the send area of the station number 2 on the control network/standby network.

• Program for station number 3



(1) Switch the target of refresh from the network to the CPU to the standby network (M0: On) when the regular network is faulty and the standby network is normal. In other cases, switch the target to the regular network (M0: Off).
(2) Load the device data from the send area of the station number 1 on the regular network to the device memory in the CPU module of the station number 3.
(3) Load the device data from the send area of the station number 2 on the regular network to the device memory in the CPU module of the station number 3.
(4) Load the device data from the area SB20 to SB1FF and SW20 to SW1FF on the regular network to the area SB20 to SB1FF and SW20 to SW1FF of the station number 3.
(5) Load the device data from the send area of the station number 1 on the standby network to the device memory in the CPU module of the station number 3.
(6) Load the device data from the send area of the station number 2 on the standby network to the device memory in the CPU module of the station number 3.
(7) Load the device data from the area SB20 to SB1FF and SW20 to SW1FF on the standby network to the area SB20 to SB1FF and SW20 to SW1FF of the station number 3.
(8) Write the send data to the send area of the station number 3 on the control network/standby network.

6 SPECIAL RELAY AND SPECIAL REGISTER

The Universal model QCPU does not support the special relay and special register described below.
☞ Page 62 Special Relay List, ☞ Page 64 Special Register List
Replace them using the method described in the following table or delete the corresponding sections.

6.1 Special Relay List

The following table lists the special relay not supported in the Universal model QCPU and measures.

Special relay not supported in the Universal model QCPU and measures

Number	Name/Description		Measures
SM80	CHK detection		The Universal model QCPU does not support the CHK instruction. For the replacing method of the CHK instruction, refer to the following. ☞ Page 17 Program Replacement Examples
SM91	Step transition monitoring timer start		The Universal model QCPU does not support the step transition monitoring timer function. For the replacement methods for the step transition monitoring timer, refer to the “Appendix 3 Restrictions on Basic Model QCPU, Universal Model QCPU, and LCPU and Alternative Methods” in the MELSEC-Q/L/QnA Programming Manual (SFC).
SM92			
SM93			
SM94			
SM95			
SM96			
SM97			
SM98			
SM99			
SM250	Largest mounted I/O number read		Operation of SD250 is not necessary. The Universal model QCPU always stores the largest mounted I/O number in SD250. Delete the corresponding parts.
SM255	MELSECNET/H module 1 information	Indicates regular network or standby network	These special relay areas are for the simple dual-structured network. The Universal model QCPU does not have the special relay areas for the simple dual-structured network. For alternative methods of these special relay areas, refer to the following. ☞ Page 56 Alternative Methods for the Simple Dual-structured Network of MELSECNET/H
SM256		At refresh from link module to CPU, selects whether to read data from the link module.	
SM257		At refresh from CPU to link module, selects whether to write data to the link module.	
SM260	MELSECNET/H module 2 information	Indicates regular network or standby network	
SM261		At refresh from link module to CPU, selects whether to read data from the link module.	
SM262		At refresh from CPU to link module, selects whether to write data to the link module.	
SM265	MELSECNET/H module 3 information	Indicates regular network or standby network	
SM266		At refresh from link module to CPU, selects whether to read data from the link module.	
SM267		At refresh from CPU to link module, selects whether to write data to the link module.	
SM270	MELSECNET/H module 4 information	Indicates regular network or standby network	
SM271		At refresh from link module to CPU, selects whether to read data from the link module.	
SM272		At refresh from CPU to link module, selects whether to write data to the link module.	
SM280	CC-Link error		Replace the relay with the I/O signals (Xn0, Xn1, and XnF) of the mounted CC-Link module.
SM330	Operation mode for low-speed execution type program		The Universal model QCPU does not support low-speed execution type programs. Delete the corresponding parts.

Number	Name/Description	Measures
SM331	Normal SFC program execution status	The Universal model QCPU supports only normal SFC programs. Delete SM331 and SM332 which are used as interlocks or replace them with SM321.
SM332	Program execution management SFC program execution status	
SM390	Access execution flag	Modify the program that the Module ready signal (Xn) is used as an interlock according to sample programs described in the manual for each module.
SM404	ON for only 1 scan after RUN of low-speed execution type programs	The Universal model QCPU does not support low-speed execution type programs. Delete the corresponding parts or replace them with the special relays for scan execution type programs (SM402 and SM403).
SM405	OFF for only 1 scan after RUN of low-speed execution type programs	
SM430	User timing clock No.5 (for low-speed execution type programs)	The Universal model QCPU does not support low-speed execution type programs. Delete the corresponding parts or replace them with the special relays for scan execution type programs (SM420 to SM424).
SM431	User timing clock No.6 (for low-speed execution type programs)	
SM432	User timing clock No.7 (for low-speed execution type programs)	
SM433	User timing clock No.8 (for low-speed execution type programs)	
SM434	User timing clock No.9 (for low-speed execution type programs)	
SM510	Low-speed execution type program executing flag	The Universal model QCPU does not support low-speed execution type programs. Delete the corresponding sections.
SM551	Module service interval time read	The Universal model QCPU does not support the service interval measurement function. Delete the corresponding sections.
SM672	Memory card file register access range flag	When outside the range of the file register in the memory card is accessed, the Universal model QCPU detects "OPERATION ERROR" (error code: 4101). Programming for detecting errors using this special relay is not necessary. Delete the corresponding sections.
SM710	CHK instruction priority flag	The Universal model QCPU does not support the CHK instruction. For the replacing method of the CHK instruction, refer to the following. ☞ Page 17 Program Replacement Examples
SM734	XCALL instruction execution condition specification	The Universal model QCPU executes the XCALL instruction on the rising edge of execution condition as well. There is no application for this special relay. Delete the corresponding sections.
SM735*1	SFC comment readout instruction in-execution flag	The Universal model QCPU does not support the following instructions: • SFC step comment readout instruction (S(P).SFCSCOMR) • SFC transition condition comment readout instruction (S(P).SFCTCOMR) Delete the corresponding sections.
SM1780*2	Power supply off detection flag	The Universal model QCPU does not store redundant power supply system information in SM1780 to SM1783. Delete the corresponding sections. (SM1780 to SM1783 are always off.)
SM1781*2	Power supply failure detection flag	
SM1782*2	Momentary power failure detection flag for power supply 1	
SM1783*2	Momentary power failure detection flag for power supply 2	

*1 The special relay can be used if the serial number (first five digits) of the Universal model QCPU is "12052" or later.

*2 The special relay can be used if the serial number (first five digits) of the Universal model QCPU is "10042" or later.

6.2 Special Register List

The following table lists the special register not supported in the Universal model QCPU and measures.

Special registers not supported in the Universal model QCPU and measures

Number	Name/Description	Measures
SD80	CHK number	The Universal model QCPU does not support the CHK instruction. For the replacing method of the CHK instruction, refer to the following. ☞ Page 17 Program Replacement Examples
SD90	Step transition monitoring timer setting value	The Universal model QCPU does not support the step transition monitoring timer function. For the replacement methods for the step transition monitoring timer, refer to the “Appendix 3 Restrictions on Basic Model QCPU, Universal Model QCPU, and LCPU and Alternative Methods” in the MELSEC-Q/L/QnA Programming Manual (SFC).
SD91		
SD92		
SD93		
SD94		
SD95		
SD96		
SD97		
SD98		
SD99		
SD280	CC-Link error	Replace these registers with the I/O signals (Xn0, Xn1, and XnF) of the mounted CC-Link module.
SD281		
SD315	Time reserved for communication processing	Service processing setting is available for the Universal model QCPU on the PLC system setting tab of the PLC parameter dialog box. Select "Specify service process time." for the service processing setting parameter and set the service processing time. Other setting methods can be selected as well.
SD430	Low-speed scan counter	The Universal model QCPU does not support low-speed execution type programs. Delete the corresponding section or replace it with the special register for scan execution type programs (SD420).
SD510	Low-speed execution type program number	The Universal model QCPU does not support low-speed execution type programs. Delete the corresponding section or replace it with the special register for scan execution type programs (SD500).
SD528	Current scan time for low-speed execution type programs	The Universal model QCPU does not support low-speed execution type programs. Delete the corresponding sections or replace them with the special registers for scan execution type programs (SD520 and SD521).
SD529		
SD532	Minimum scan time for low-speed execution type programs	The Universal model QCPU does not support low-speed execution type programs. Delete the corresponding sections or replace them with the special registers for scan execution type programs (SD524 to SD527).
SD533		
SD534	Maximum scan time for low-speed execution type programs	
SD535		
SD544	Cumulative execution time for low-speed execution type programs	The Universal model QCPU does not support low-speed execution type programs. Delete the corresponding sections.
SD545		
SD546	Execution time for low-speed execution type programs	The Universal model QCPU does not support low-speed execution type programs. Delete the corresponding sections.
SD547		
SD550	Service interval measurement module	The Universal model QCPU does not support the service interval measurement function. Delete the corresponding sections.
SD551	Service interval time	
SD552		
SD720	Program No. specification for PLAODP instruction	The Universal model QCPU does not support the PLAODP instruction. Delete the corresponding section.
SD1780*1	Power supply off detection status	The Universal model QCPU does not store redundant power supply system information in SD1780 to SD1783. Delete the corresponding sections. (SD1780 to SD1783 are always off.)
SD1781*1	Power supply failure detection status	
SD1782*1	Momentary power failure detection counter for power supply 1	
SD1783*1	Momentary power failure detection counter for power supply 1	

*1 The special register can be used if the serial number (first five digits) of the Universal model QCPU is "10042" or later.

REVISIONS

Version	Date of Issue	Revision
-	January 2008	First edition
A	March 2008	• The following modules have been added. Q13UDHCPU, Q26UDHCPU. • The following chapters and sections have been modified in accordance with the version upgrade of the Universal model QCPU. Chapter 1 (2), Chapter 2, Section 4.4.
B	May 2008	• The following modules have been added. Q03UDECPU, Q04UDEHCPU, Q06UDEHCPU, Q13UDEHCPU, Q26UDEHCPU. • The following chapters and sections have been modified in accordance with the version upgrade of the Universal model QCPU. Chapter 1 (1), (6), Chapter 2, Section 4.3, 4.6, 5.1, 5.2. • Software listed in Chapter 2 (3) "Software needed to be upgraded for the compatibility with the Universal model QCPU" have been reviewed and modified. • "GX Converter" has been added to the list in Chapter 2 (4) "Software not supported in the Universal model QCPU".
C	December 2008	• The following modules have been added. Q10UDHCPU, Q20UDHCPU, Q10UDEHCPU, Q20UDEHCPU. • The following chapters and sections have been modified in accordance with the version upgrade of the Universal model QCPU. Chapter 1 , Chapter 2, Chapter 3.
D	January 2009	Section 4.3 has been added
E	September 2009	Chapter 1 (4); Two precaution items have been added to the following table. External communication.
F	July 2011	Reference manuals, chapters, and sections have been modified in accordance with manual structure changes.
G	January 2012	Descriptions on the new functions of the Universal model QCPU (serial number (first five digits) "13102" or later) have been added.
H	February 2013	Descriptions on the new CPU module, High-speed Universal model QCPU, have been added.
I	April 2015	The following chapters and section have been modified in accordance with the descriptions in the relevant manual. Chapter 1, 2, Section 4.3.
J	December 2015	Products needed to be replaced for the compatibility with the Universal model QCPU in Chapter 2 (1) are reviewed and modified.
K	July 2016	Descriptions have been modified throughout the bulletin in accordance with issue of the introduction (FA-A-0209) of this bulletin.
L	February 2017	Descriptions have been reviewed and modified throughout the bulletin.
M	August 2017	Chapter 2, 3, and Section 6.1 have been modified and Section 5.8 has been added.
N	March 2019	Available for e-Manual Viewer