Mitsubishi Programmable Controller  Training Manual
Q-series basic course (for GX Works2)

MITSUBISHI
Mitsubishi Programmable Controller
Training Manual

Q-series basic course (for GX Works2)

Specifications subject to change without notice.
SAFETY PRECAUTION

(Always read these instructions before using the products.)

When designing the system, always read the relevant manuals and give sufficient consideration to safety. During the exercise, pay full attention to the following points and handle the product correctly.

[EXERCISE PRECAUTIONS]

⚠️ WARNING

- Do not touch the terminals while the power is on to prevent electric shock.
- Before opening the safety cover, make sure to turn off the power or ensure the safety.
- Do not touch the movable portion.

⚠️ CAUTION

- Follow the instructor's direction during the exercise.
- Do not remove the module of the demonstration machine or change wirings without permission. Doing so may cause failures, malfunctions, personal injuries and/or a fire.
- Turn off the power before installing or removing the module. Failure to do so may result in malfunctions of the module or electric shock.
- When the demonstration machine (such as X/Y table) emits abnormal odor/sound, press "Power switch" or "Emergency switch" to turn off.
- When a problem occurs, notify the instructor as soon as possible.
**REVISIONS**

*The textbook number is written at the bottom left of the back cover.*

<table>
<thead>
<tr>
<th>Print date</th>
<th><em>Textbook number</em></th>
<th>Revision</th>
</tr>
</thead>
</table>

This textbook confers no industrial property rights or any rights of any other kind, nor does it confer any patent licenses. Mitsubishi Electric Corporation cannot be held responsible for any problems involving industrial property rights which may occur as a result of using the contents noted in this textbook.

© 2012 MITSUBISHI ELECTRIC CORPORATION
## CONTENTS

### CHAPTER 1  BASICS OF PROGRAMMABLE CONTROLLER  1- 1 to 1-14

1.1 Program........................................................................................................... 1- 1  
1.2 Program Processing Procedure........................................................................ 1- 4  
1.3 MELSEC-QnUD Module Configuration............................................................ 1- 5  
1.4 External I/O Signal and I/O Number................................................................. 1-11  
1.5 System Configuration and I/O Number of Demonstration Machine................. 1-14

### CHAPTER 2  OPERATING GX Works 2 2- 1 to 2-64

2.1 Features of GX Works2.................................................................................... 2- 3  
2.1.1 MELSOFT IQ Works.................................................................................... 2- 7  
2.2 Basic Knowledge Required for Operating GX Works2.................................... 2- 9  
2.2.1 Screen configuration in GX Works2............................................................ 2- 9  
2.2.2 Ladder editor.............................................................................................. 2-11  
2.2.3 Project........................................................................................................ 2-20  
2.3 Operation Before Creating Ladder Program.................................................. 2-22  
2.3.1 Starting up GX Works2............................................................................... 2-22  
2.3.2 Creating a new project................................................................................. 2-23  
2.4 Preparation for Starting Up CPU...................................................................... 2-25  
2.5 Creating Ladder Program................................................................................ 2-32  
2.5.1 Creating a ladder program using the function keys.................................... 2-32  
2.5.2 Creating a ladder program using the tool buttons.................................... 2-34  
2.6 Converting Program (Ladder Conversion)....................................................... 2-36  
2.7 Writing/Reading Data to/from Programmable Controller CPU....................... 2-37  
2.8 Monitoring Ladder Program Status................................................................. 2-40  
2.9 Diagnosing Programmable Controller CPU.................................................... 2-43  
2.10 Editing Ladder Program.................................................................................. 2-45  
2.10.1 Modifying a part of the ladder program.................................................... 2-45  
2.10.2 Drawing/deleting lines.............................................................................. 2-47  
2.10.3 Inserting/deleting rows............................................................................. 2-50  
2.10.4 Cutting/copying ladder program............................................................... 2-55  
2.11 Verifying Data............................................................................................... 2-58  
2.12 Saving Ladder Program.................................................................................. 2-59  
2.12.1 Saving newly-created or overwritten projects........................................ 2-59  
2.12.2 Saving a project with another name......................................................... 2-60  
2.13 Reading the saved project.............................................................................. 2-61  
2.14 Opening Projects in Different Format............................................................ 2-62  
2.15 Saving Projects in Different Format............................................................... 2-63

### CHAPTER 3  DEVICE AND PARAMETER OF PROGRAMMABLE CONTROLLER  3- 1 to 3- 6

3.1 Device............................................................................................................. 3- 1  
3.2 Parameter........................................................................................................ 3- 3
CHAPTER 4  SEQUENCE AND BASIC INSTRUCTIONS -PART 1-  4- 1 to 4-42

4.1 List of Instruction Explained in this Chapter ................................................. 4- 1
4.2 Differences between OUT and SET / RST .................................................. 4- 4
4.3 Measuring Timer ......................................................................................... 4- 5
4.4 Counting by Counter ................................................................................. 4- 6
4.5 PLS / PLF .................................................................................................... 4-14
4.6 MC / MCR .................................................................................................. 4-20
4.7 FEND / CJ / SCJ / CALL / RET ................................................................. 4-24
4.8 Exercise ...................................................................................................... 4-35
4.8.1 Exercise 1  LD to NOP ........................................................................ 4-35
4.8.2 Exercise 2  SET, RST ........................................................................ 4-36
4.8.3 Exercise 3  PLS, PLF ........................................................................ 4-38
4.8.4 Exercise 4  CJ, CALL, RET, FEND .................................................... 4-39

CHAPTER 5  BASIC INSTRUCTION -PART 2-  5- 1 to 5-58

5.1 Notation of Values (Data) ................................................................. 5- 1
5.2 Transfer Instruction ............................................................................. 5- 9
5.2.1 MOV (P) ............................................................................................ 5- 9
5.2.2 BIN (P) ............................................................................................ 5-16
5.2.3 BCD (P) ............................................................................................ 5-18
5.2.4 Example of specifying digit for bit devices and transferring data .... 5-21
5.2.5 FMOV (P) / BMOV (P) ................................................................. 5-22
5.3 Comparison Operation Instruction ...................................................... 5-27
5.4 Arithmetic Operation Instruction ......................................................... 5-32
5.4.1 +(P) / -(P) ...................................................................................... 5-32
5.4.2 *(P) / /(P) ....................................................................................... 5-36
5.4.3 32-bit data instructions and their necessity ....................................... 5-41
5.4.4 Calculation examples for multiplication and division including decimal points 5-43
5.5 Index Register and File Register ............................................................ 5-44
5.5.1 How to use index register Z ........................................................... 5-44
5.5.2 How to use file register R ............................................................... 5-46
5.6 External Setting of Timer/Counter Set Value and External Display of Current Value 5-49
5.7 Exercise .................................................................................................. 5-51
5.7.1 Exercise 1  MOV ................................................................. Project name QTEST5  5-51
5.7.2 Exercise 2  BIN and BCD conversion ................................................ Project name QTEST6  5-52
5.7.3 Exercise 3  FMOV ................................................................. Project name QTEST7  5-53
5.7.4 Exercise 4  Comparison instruction ................................................... Project name QTEST8  5-54
5.7.5 Exercise 5  Addition and subtraction instructions .............................. Project name QTEST9  5-55
5.7.6 Exercise 6  Multiplication and division instructions ............................ Project name QTEST10  5-56
5.7.7 Exercise 7  D-multiplication and D-division ........................................ Project name QTEST11  5-57
CHAPTER 6  HOW TO USE OTHER FUNCTIONS  6- 1 to 6-36

6.1  Test Function at Online ................................................................. 6- 1
6.1.1  Turning on and off the device "Y" forcibly ........................................... 6- 2
6.1.2  Setting and resetting the device "M"...................................................... 6- 4
6.1.3  Changing the current value of the device "T"......................................... 6- 5
6.1.4  Reading error steps ............................................................................ 6- 6
6.1.5  Remote STOP and RUN ..................................................................... 6- 7
2 6.2  Forced I/O Assignment by Parameter Settings ...................................... 6- 8
6.3  How to Use Retentive Timers .................................................................. 6-10
6.4  Device Batch Replacement ...................................................................... 6-12
6.4.1  Batch replacement of device numbers .................................................. 6-12
6.4.2  Batch change of specified devices between normally open contacts and normally closed contacts ................................................................. 6-13
6.5  Online Program Change ...................................................................... 6-14
6.6  Registering Devices .............................................................................. 6-15
6.7  How to Create Comments ..................................................................... 6-16
6.8  Setting Security for Projects .................................................................. 6-23
6.8.1  Setting and resetting security for projects ............................................. 6-24
6.8.2  Managing (adding, deleting, and changing) users .................................. 6-25
6.8.3  Logging in projects .............................................................................. 6-29
6.8.4  Changing access authority for each access level ................................... 6-30
6.9  Sampling Trace Function ...................................................................... 6-31

CHAPTER 7  PROGRAMMING INTELLIGENT FUNCTION MODULE  7- 1 to 7-26

7.1  Intelligent Function Module .................................................................. 7- 1
7.2  Data Communication between Intelligent Function Modules and CPUs ......................................................... 7- 2
7.2.1  I/O signals to CPUs ............................................................................ 7- 3
7.2.2  Data communication with intelligent function modules .......................... 7- 4
7.3  Communication with Intelligent Function Module ..................................... 7- 5
7.3.1  Communication methods with intelligent function modules .................. 7- 5
7.4  Intelligent Function Module System in Demonstration Machine .......... 7- 6
7.5  Q64AD Analog/Digital Converter Module ............................................ 7- 7
7.5.1  Names of parts .................................................................................... 7- 7
7.5.2  A/D conversion characteristics .............................................................. 7- 8
7.5.3  List of I/O signals and buffer memory assignment .................................. 7- 9
7.5.4  Adding or setting intelligent function module data ............................... 7-12
7.5.5  Exercise with the demonstration machine ............................................ 7-16
7.6  Q62DAN Digital/Analog Converter Module ............................................ 7-17
7.6.1  Names of parts .................................................................................... 7-17
7.6.2  D/A conversion characteristics .............................................................. 7-18
7.6.3  List of I/O signals and buffer memory assignment .................................. 7-19
7.6.4  Adding or setting intelligent function module data ............................... 7-21
7.6.5  Exercise with the demonstration machine ............................................ 7-25
CHAPTER 8  SIMULATION FUNCTION  
8.1 Simulation Function ................................................................. 8-1
8.2 Starting/Stopping Simulation ..................................................... 8-1
8.3 Debugging with Example Program ............................................. 8-2
  8.3.1 Monitoring and testing device status .................................... 8-3

CHAPTER 9  MAINTENANCE  
9.1 Typical Trouble ................................................................. 9-1
9.2 Maintenance ......................................................................... 9-2
9.3 Consumable Product ............................................................ 9-3
9.4 Service Life of Output Relay .................................................... 9-4
9.5 Spare Product ........................................................................ 9-5
9.6 Using Support Equipment ....................................................... 9-7

APPENDIX  
Appendix 1  I/O Control Mode ..................................................... App.-1
  1.1 Direct mode ......................................................................... App.-1
  1.2 Refresh mode ....................................................................... App.-2
  1.3 Comparisons between the direct mode and refresh mode ........ App.-3

Appendix 2  Special Relay ............................................................. App.-4
Appendix 3  Special Register ........................................................ App.-5
Appendix 4  Application Program Example ..................................... App.-6
  4.1 Flip-flop ladder .................................................................. App.-6
  4.2 One shot ladder .................................................................. App.-8
  4.3 Long-time timer .................................................................. App.-9
  4.4 Off delay timer .................................................................... App.-10
  4.5 On delay timer (momentary input) ....................................... App.-11
  4.6 ON-OFF repeat ladder ......................................................... App.-12
  4.7 Preventing chattering input ................................................ App.-12
  4.8 Ladders with a common line ............................................... App.-13
  4.9 Time control program ....................................................... App.-14
  4.10 Clock ladder ....................................................................... App.-15
    4.10.1 Clock function (supplement) .......................................... App.-16
  4.11 Starting \( \bigcirc \) - \( \triangle \) operation of electrical machinery ...... App.-18
  4.12 Displaying elapsed time and outputting before time limit ...... App.-19
  4.13 Retentive timer .................................................................. App.-20
  4.14 Switching timer set value externally .................................... App.-21
  4.15 Setting counters externally ................................................ App.-22
  4.16 Measuring operation time ................................................... App.-24
  4.17 Measuring cycle time .......................................................... App.-24
  4.18 Application example of (D) CML (P) .................................... App.-25
  4.19 Program showing divided value of 4-digit BIN value to 4 places of decimals ... App.-26
  4.20 Carriage line control .......................................................... App.-29
  4.21 Compressor sequential operation using ring counters .......... App.-31
  4.22 Application example of positioning control ......................... App.-35
  4.23 Application example using index Z .................................... App.-36
  4.24 Application example of FIFO instruction ............................. App.-38

(4)
INTRODUCTION

This textbook explains the programmable controller, the program editing methods with GX Works2, the sequence instructions and the application instructions for understanding the MELSEC-Q series programming.

The multiple CPU system is available for the MELSEC-Q series with multiple CPU modules, but this textbook explains the case in which one CPU module is used.

The related manuals are shown below.

1. QCPU User’s Manual (Hardware Design, Maintenance and Inspection)
   SH-(NA)080483ENG
   Explains the hardware.

2. QnUCPU User’s Manual (Function Explanation, Program Fundamentals)
   SH(NA)-080807ENG
   Explains the functions and programming method.

3. MELSEC-Q/L Programming Manual (Common Instruction)
   SH(NA)-080809ENG
   Explains details of each instruction.

4. GX Works2 Beginner's Manual (Simple Project)
   SH(NA)-080787ENG

5. GX Works2 Version 1 Operating Manual (Common)
   SH(NA)-080779ENG

6. GX Works2 Version 1 Operating Manual (Simple Project)
   SH(NA)-080780ENG

7. Before Using the Product
   BCN-P5782

   SH(NA)-080055

   SH(NA)-080054

    SH(NA)-080042

11. MELSOFT GX Works2 FB Quick Start Guide
    L-08182ENG
CHAPTER 1  BASICS OF PROGRAMMABLE CONTROLLER

1.1 Program

If a programmable controller is assumed as a control ladder, it can be described by an input ladder, output ladder, and internal sequential operation.

Figure 1.1 Programmable controller configuration

A programmable controller is an electronic device centered around microcomputers. Actually, a programmable controller is assemblies of relays, timers, and counters. As shown in figure 1.1, the internal sequential operation is executed by turning on or off the coil. The on/off condition of the coil depends on the connection condition (in series or in parallel) and results of the normally open or normally closed contacts.

"Relay", which is also called an electromagnetic relay, is a switch to relay signals. The relay is a key component to make up a logic ladder.

1) Energizing the coil — Magnetization
   - The normally open contact closes. (Conducted)
   - The normally closed contact opens. (Not conducted)

2) De-energizing the coil — Demagnetization
   - The normally open contact opens. (Not conducted)
   - The normally closed contact closes. (Conducted)
**Internal Sequential Operation**

The following shows the signal flow of the internal sequential operation of figure 1.1.

1) When the sensor turns on, the coil of the input relay X6 is magnetized.
2) Magnetizing the coil of the input relay X6 conducts the normally open contact X6 and magnetizes the coil of the output relay Y74.
   (As the timer is not magnetized at this time, the normally closed contact remains conducted.)
3) Once the coil of the output relay Y74 is magnetized, the external output contact Y74 is conducted and the magnetic contactor (MC) is turned on.
4) Turning off the sensor demagnetizes the coil of the input relay X6 and the normally open contact X6 becomes non-conductive.
   As the self-maintaining normally open contact Y74 is conducted, the coil remains magnetized. (Self-maintaining operation)
5) When the coil of the output relay Y74 is magnetized (with the normally open contact Y74 conducted), turning off the sensor (with normally closed contact X6 conducted) magnetizes the coil of the timer T1 and the timer starts measuring the time.
   After three sec. (K30 indicates 3.0sec.), the normally open contact of the timer becomes conducted and the normally closed contact becomes non-conductive.
6) As a result, the coil of the output relay Y74 demagnetizes and the load magnet contactor drops.
   Also, the output relay self-maintenance is released.

**Operation diagram**

The following time chart explains the input/output relays and timer operations.
The internal sequential operation can be regarded as the program of the programmable controller. The program is saved in the program memory as similar to the instruction list.

<table>
<thead>
<tr>
<th>Step number</th>
<th>Instruction word</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LD</td>
<td>X6</td>
</tr>
<tr>
<td>1</td>
<td>OR</td>
<td>Y74</td>
</tr>
<tr>
<td>2</td>
<td>ANI</td>
<td>T1</td>
</tr>
<tr>
<td>3</td>
<td>OUT</td>
<td>Y74</td>
</tr>
<tr>
<td>4</td>
<td>LD</td>
<td>Y74</td>
</tr>
<tr>
<td>5</td>
<td>ANI</td>
<td>X6</td>
</tr>
<tr>
<td>6</td>
<td>OUT</td>
<td>T1 K30</td>
</tr>
<tr>
<td>10</td>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1.2 Program

- A program consists of a large number of instruction words and devices.

- The instructions contain instruction words and devices. In addition, the instructions are numbered to represent the order of operations. The numbers are called step numbers. (Instruction words are also called instructions.)

- The number of steps varies depending on the types of instructions or the setting method for the values to be used for the I/O numbers and operations. (The more steps are needed for the operation with complicated operation.)

- The instructions repeat from the step number 0 to the END instruction. (This is called "repeat operation", "cyclic operation" or "scanning".) Amount of time necessary for one cycle is called operation cycle (scan time).

- The number of steps from the step number 0 to the END instruction is the length or size of the program.

- The program is stored in the program memory inside the CPU. The operation is executed in a ladder block unit. One ladder block ranges from the operation start instruction (LD, LDI) to the OUT instruction (including the data instruction).
1.2 Program Processing Procedure

The operation process is executed in series from the start step of the program memory left to right and top to bottom (in the order of 1), 2) ... 17)) in a ladder block unit as shown below.
1.3 MELSEC-QnUD Module Configuration

(1) Universal model

The Universal model QCPU is used for a training in this textbook, therefore, "QCPU" indicates "Universal model QCPU" unless otherwise noted.

(2) Basic configuration of a programmable controller system

The following figure shows an actual programmable controller configuration.

Figure 1.3 MELSEC-QnUD module configuration (when Q3DB is used)
### Base Unit

<table>
<thead>
<tr>
<th>Base Unit</th>
<th>Main base unit</th>
<th>Extension base unit (Requiring a power supply module)</th>
<th>(Not requiring a power supply module)</th>
</tr>
</thead>
<tbody>
<tr>
<td>With three I/O modules</td>
<td>Power supply: Q33B</td>
<td>Power supply: Q63B</td>
<td>Q52B (For two modules)</td>
</tr>
<tr>
<td>With five I/O modules</td>
<td>Power supply: Q35B</td>
<td>Power supply: Q65B</td>
<td>Q55B</td>
</tr>
<tr>
<td>With eight I/O modules</td>
<td>Power supply: Q38B</td>
<td>Power supply: Q65B</td>
<td>Q68B</td>
</tr>
<tr>
<td>With 12 I/O modules</td>
<td>Power supply: Q312B</td>
<td>Power supply: Q68B</td>
<td>Q612B</td>
</tr>
<tr>
<td>Multiple CPU high speed main base unit</td>
<td>Power supply: Q38DB</td>
<td>Power supply: Q65B</td>
<td>Q612B</td>
</tr>
</tbody>
</table>

- The main roles of the base unit are; fixing the power supply module, CPU module, and I/O modules, supplying 5VDC power from the power supply module to the CPU module and I/O modules, and transmitting the control signals to each module.
### Power Supply Module

<table>
<thead>
<tr>
<th>Module name</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q61P</td>
<td>100V to 240VAC</td>
<td>5VDC 6A</td>
</tr>
<tr>
<td>Q62P</td>
<td>100V to 240VAC</td>
<td>5VDC 3A, 24VDC 0.6A</td>
</tr>
<tr>
<td>Q63P</td>
<td>24VDC</td>
<td>5VDC 6A</td>
</tr>
<tr>
<td>Q64P(N)</td>
<td>100V to 120V/AC200 to 240VAC</td>
<td>5VDC 8.5A</td>
</tr>
<tr>
<td>Q61P-D</td>
<td>100V to 240VAC</td>
<td>5VDC 6A</td>
</tr>
</tbody>
</table>

### CPU Module

<table>
<thead>
<tr>
<th>CPU type</th>
<th>Program capacity (maximum)</th>
<th>Basic instruction processing speed</th>
<th>Maximum I/O points for connecting to a programmable controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q00UJ CPU</td>
<td>10K steps</td>
<td>120ns</td>
<td>256 points</td>
</tr>
<tr>
<td>Q00UCPU</td>
<td>10K steps</td>
<td>80ns</td>
<td>1024 points</td>
</tr>
<tr>
<td>Q01UCPU</td>
<td>15K steps</td>
<td>60ns</td>
<td>1024 points</td>
</tr>
<tr>
<td>Q02UCPU</td>
<td>20K steps</td>
<td>40ns</td>
<td>2048 points</td>
</tr>
<tr>
<td>Q03UD(E)CPU</td>
<td>30K steps</td>
<td>20ns</td>
<td>2048 points</td>
</tr>
<tr>
<td>Q04UD(E)HCPU</td>
<td>40K steps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q06UD(E)HCPU</td>
<td>60K steps</td>
<td>9.5ns</td>
<td>4096 points</td>
</tr>
<tr>
<td>Q10UD(E)HCPU</td>
<td>100K steps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q13UD(E)HCPU</td>
<td>130K steps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q20UD(E)HCPU</td>
<td>200K steps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q26UD(E)HCPU</td>
<td>260K steps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q50UDEHCPU</td>
<td>500K steps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q100UDEHCPU</td>
<td>1000K steps</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### I/O Module

<table>
<thead>
<tr>
<th>Format</th>
<th>8 points</th>
<th>16 points</th>
<th>32 points</th>
<th>64 points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input module</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120VAC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>240VAC</td>
<td>○</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>24VDC (positive common)</td>
<td>-</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>24VDC (high-speed input)</td>
<td>○</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>24VDC (negative common)</td>
<td>-</td>
<td>○</td>
<td>○</td>
<td>-</td>
</tr>
<tr>
<td>5/12VDC</td>
<td>-</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Contact output</td>
<td>-</td>
<td>○</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Independent contact output</td>
<td>○</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Output module</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Triac output</td>
<td>-</td>
<td>○</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Transistor output (sink)</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Transistor output (source)</td>
<td>-</td>
<td>○</td>
<td>○</td>
<td>-</td>
</tr>
<tr>
<td>I/O mixed</td>
<td>○</td>
<td>-</td>
<td>○</td>
<td>-</td>
</tr>
</tbody>
</table>
A QCPU equips a built-in memory as standard for storing parameters and programs, therefore, the programs can be executed without a memory card.

The memory cards are required for the situations in the table below.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM card</td>
<td>Data can be written or changed within the memory capacity.</td>
</tr>
<tr>
<td></td>
<td><em>Example of the usage</em></td>
</tr>
<tr>
<td></td>
<td>• For the boot operation</td>
</tr>
<tr>
<td></td>
<td>• For storing the sampling trace data</td>
</tr>
<tr>
<td></td>
<td>• For storing the SFC trace data</td>
</tr>
<tr>
<td></td>
<td>• For storing the error history data</td>
</tr>
<tr>
<td>Flash card</td>
<td>The contents of the program memory or the specified file can be written at a time. The newly written data replaces all original data. Data can be read by the READ instruction of the sequence program.</td>
</tr>
<tr>
<td></td>
<td><em>Example of the usage</em></td>
</tr>
<tr>
<td></td>
<td>• For the boot operation</td>
</tr>
<tr>
<td></td>
<td>• When the changing the data is unnecessary</td>
</tr>
<tr>
<td>ATA card</td>
<td>Data can be written or changed within the program capacity.</td>
</tr>
<tr>
<td></td>
<td>Programmable controller user data of an ATA card can be accessed by the file access instruction (such as the FWRITE instruction) in a sequence program through a CSV format or binary format.</td>
</tr>
<tr>
<td></td>
<td><em>Example of the usage</em></td>
</tr>
<tr>
<td></td>
<td>• For the boot operation</td>
</tr>
<tr>
<td></td>
<td>• For programmable controller user data (general-purpose data)</td>
</tr>
</tbody>
</table>

- Memory cards are required when the data capacity exceeds the capacity of the built-in program memory, standard RAM, and standard ROM.
- Select the memory card according to the size of the program or the type of the data to be stored.
- Install the enclosed backup battery before using the SRAM-type RAM card first. The SRAM card data cannot be baked up unless the battery is installed.
- Format the memory card before using it.
- Data can be written to a Flash card for 100,000 times, and for an ATA card, data can be written for 1,000,000 times.
The memory of the Universal model QCPU consists of the following blocks.

- **Program memory**: A memory for storing programs and parameters for a CPU module operation. A program operation is executed by transferring a program stored in the program memory to the program cache memory.

- **Program cache memory**: A memory for operating programs. A program operation is executed by transferring a program stored in the program memory to the program cache memory.

- **Standard RAM**: A memory for using file registers, local devices, and sampling trace files without a memory card. Using the standard RAM as the file registers enables the high-speed access as well as data registers. The standard RAM is also used for storing the module error collection file.

- **Standard ROM**: A memory for storing data such as parameters and programs.

- **Memory card (RAM)**: A card for storing the local device, debug data, SFC trace data, and error history data with the parameters and program.

- **Memory card (ROM)**: A Flash card for storing parameters, programs, and file registers. An ATA card stores parameters, programs, and the programmable controller user data (general-purpose files).

*1: A memory card cannot be used for Q00UJ CPU, Q00UCPU, Q01UCPU.
*2: Q00UJ CPU has no standard RAM.
Secure backup by long-term storage
Programs and parameter files are automatically backed up to the program memory (Flash ROM) which does not require a battery backup. This prevents a loss of the program and parameter data due to the flat battery. The battery backup time is also reduced significantly.
In addition, the important data (such as device data) can be backed up to the standard ROM to prevent a loss of the data due to the flat battery in case of consecutive holidays.
The backup data is restored automatically when the power is turned on next time.
### 1.4 External I/O Signal and I/O Number

(1) **Wiring of I/O devices**

The signals output from the external input devices are substituted by the input numbers which are determined by the installation positions and terminal numbers of the connected input module and used in a program.

For the operation results output (coil), use the output numbers which are determined by the installation position and the terminal number of the output module to which the external output module is connected.

- Input numbers are hexadecimal numbers that start with 0. Input/output numbers share the same numbers. "X" at the beginning of the number represents "Input", and "Y" indicates "Output".

- The maximum number of the QCPU (Q mode) input/output number is 4,096.

- The input/output number is sometimes referred to as the I/O number (IN/OUT).

---

**Figure 1.4 Wiring of I/O devices**
(2) I/O numbers of a main base unit

The I/O numbers of I/O modules which are attached to a main base unit are assigned as follows. This configuration applies to both I/O modules and intelligent function modules.

- The I/O numbers of one slot (one module) are assigned in ascending order in 16-point unit (0 to F). As a standard, 16-point modules should be attached to all slots. For example, the following figure shows the I/O numbers of when a 32-point module is attached to the fifth slot.

- The I/O numbers are also assigned to a vacant slot (a slot with no I/O module installed). For example, if the third slot is vacant, the I/O numbers are assigned as shown below. (in the initial setting) The number of assigned points can be changed by the setting.

- For the multiple CPU configuration (two to four CPUs), the I/O numbers are assigned from a slot next to a slot where a CPU is attached.
(3) I/O numbers of an extension base unit
Connect an extension base unit when the number of slots of the main base unit is insufficient.
The I/O numbers are assigned as follows in the initial setting.
This configuration applies to both I/O modules and intelligent function modules.

The slots of the extension base unit are also assigned in ascending order in 16-point unit.
The start I/O number of the extension base unit is assigned from the last number of the main base unit or of the previous extension base unit.
Setting "0" to the parameter can assign the I/O number to the vacant slot or areas with no slot.

The following table shows the number of available extension base units.

<table>
<thead>
<tr>
<th>CPU type</th>
<th>Number of stages (including the ones connected with GOT in bus connection)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal model</td>
<td></td>
</tr>
<tr>
<td>Q00UJ CPU</td>
<td>2</td>
</tr>
<tr>
<td>Q00UCPU, Q01UCPU, Q02UCPU</td>
<td>4</td>
</tr>
<tr>
<td>Other than the above</td>
<td>7</td>
</tr>
</tbody>
</table>
1.5 System Configuration and I/O Number of Demonstration Machine

![Diagram of system configuration]

- **Power supply module**
- **CPU module**
- **Input module**
- **Output module**

**Base unit Q38DB**

**Peripheral device**

**USB cable**

**I/O panel**

**MELSEC-Q**

**D/A OUTPUT**

**A/D INPUT**

**Adaptor**

**ON**

**OFF**

**Y77** **Y76** **Y75** **Y74** **Y73** **Y72** **Y71** **Y70**

**Y7F** **Y7E** **Y7D** **Y7C** **Y7B** **Y7A** **Y79** **Y78**

**X7** **X6** **X5** **X4** **X3** **X2** **X1** **X0**

**XE** **XD** **XC** **XB** **XA** **X9** **X8**

**ON**

**OFF**

**26** **3709** **1402**

**ON**

**OFF**

**D/A OUTPUT**

**Adaptor**

**ON**

**OFF**
CHAPTER 2 OPERATING GX Works2

GX Works2 is a programming tool for designing, debugging, and maintaining programs on Windows®. GX Works2 has improved functionality and operability, with easier-to-use features compared to existing GX Developer.

- Main functions of GX Works2
  GX Works2 can manage programs and parameters in units of projects for each programmable controller CPU.
  - Programming
    Programs can be created in a Simple project in a similar way with existing GX Developer. Structured programming in a Structured project is also available with GX Works2.

- Setting parameters
  The parameters for programmable controller CPUs and network parameters can be set with GX Works2. Intelligent function module parameter can be set as well.

- Writing/reading data to/from a programmable controller CPU
  Created sequence programs can be written to/read from a programmable controller CPU using the Read from PLC/Write to PLC function. Also, with the Online program change function, the sequence programs can be changed even when the programmable controller CPU is in RUN.
- Monitoring/debugging
  Created sequence programs can be written to the programmable controller CPU and device values at operation can be monitored online/offline.
  Programs can be monitored and debugged.

- Diagnostics
  The current error status and error history of the programmable controller CPU can be diagnosed.
  With the diagnostics function, the recovery work is completed in a short time.
  With the System monitor function (for QCPU (Q mode)/LCPU), detailed information on such as intelligent function modules can be obtained. This helps to shorten the recovery work time at error occurrence.

Diagnosing the programmable controller CPU status (PLC diagnostics screen)
2.1 Features of GX Works2

This section explains the features of GX Works2.

(1) Project types in GX Works2
In GX Works2, the project type can be selected from either of Simple project or Structured project.

(a) Simple project
The Simple project creates sequence programs using instructions for Mitsubishi programmable controller CPU. Programs in a Simple project can be created in a similar way to existing GX Developer.

(b) Structured project
In a Structured project, programs can be created by structured programming. By segmenting a whole control process program into common program parts, highly manageable and usable programming (structured programming) is possible.

Programs are created with programmable controller CPU instructions. Created programs can be operated as sequence programs.
(2) Enhanced use of program assets
Projects created with existing GX Developer can be utilized in a Simple project. Utilizing the past assets improves the efficiency of program design.

(3) Sharing Program Organization Unit (POU) registered as libraries
In a Structured project, programs, global labels, and structures frequently used can be registered as user libraries. Utilizing these user libraries reduces time required for creating programs.
(4) Wide variety of programming languages
The wide variety of programming languages available with GX Works2 enables to select the optimum programming language according to control.

- **<Ladder>** Programming similar to existing GX Developer

- **<SFC>** Programming to clarify the procedure

- **<Structured ladder>** Programming a ladder program graphically

- **<ST>** Programming in a text language similar to C language

(5) Other features
(a) Offline debugging
Online debugging using the simulation function is possible with GX Works2. This enables debugging to ensure the normal operation of created sequence programs without connecting GX Works2 to the programmable controller CPU.

Connecting the programmable controller CPU is unnecessary.

Without connecting the programmable controller CPU, programs can be monitored and debugged in the same way with debugging by the programmable controller CPU.
(b) The screen layout can be customized to the user's preference. The docking windows enable to change the screen layout of GX Works2 without restriction.
2.1.1 MELSOFT iQ Works

MELSOFT iQ Works integrates the engineering software (GX Works2, MT Developer2, and GT Designer3).
Sharing the design information such as the system design and programming in the whole control system improves the efficiency of program design and efficiency of programming, which reduces costs.

MELSOFT iQ Works

GT Designer3
(GOT drawing software)

GX Works2
(Programmable controller programming software)

MT Developer2
(Motion controller programming software)

MELSOFT Navigator
(System configuration management tool)

Sharing the design information between the software

Design information data base

POINT

To start MELSOFT Navigator and each engineering software, click the Start button and follow the procedure below.

• MELSOFT Navigator: [MELSOFT Application] → [MELSOFT iQ Works] → [MELSOFT Navigator]
• GX Works2: [MELSOFT Application] → [GX Works2] → [GX Works2]
• MT Developer2: [MELSOFT Application] → [MT Works2] → [MT Developer2]
• GT Designer3: [MELSOFT Application] → [GT Works3] → [GT Designer3]
[Purpose of the engineering environment]

1) Integrating development environment which was independent of each device
2) Sharing the design information in whole development phases (system designing, programming, test/startup, and operation/maintenance)

POINT

iQ Platform is a FA integrated concept of MITSUBISHI ELECTRIC.

Integrated Q/improved Quality/intelligent&Quick/innovation&Quest
2.2 Basic Knowledge Required for Operating GX Works2

2.2.1 Screen configuration in GX Works2

1) Title bar
2) Menu bar
3) Toolbar
4) Tab
5) View contents display area
6) View selection area
7) Edit screen (work window)
8) Output window
9) Status bar
1) Title bar
Title bar displays the name of the active project.
- Resizes or terminates GX Works2.
- Maximizes or restores GX Works2.
- Displays the name and the path of the project.
- Minimizes GX Works2.
- Terminates GX Works2.

2) Menu bar
Menu bar is the most frequently used item when operating GX Works2. Click the menu bar to select a variety of functions from the drop-down menu.

3) Toolbar
Toolbar equips buttons to easily access the commonly-used functions. This enables a quicker operation.

4) Tab
When multiple work windows are open, they are displayed in the tab browser format. Clicking a tab activates the corresponding work window.

5) View contents display area
View contents display area displays the contents of the currently selected view.

6) View selection area
View selection area allows selection of the view to be displayed.

7) Edit screen (work window)
Edit screen displays various screens such as ladder program creation screen and comment creation screen for editing ladder diagrams, comments, and parameters.

8) Output window
Output window displays compilation and check results (such as errors and warnings).

9) Status bar
Status bar displays the status information of GX Works2.
2.2.2 Ladder editor

This section explains the screen display of the GX Works2 ladder editor and its basic operations.

(1) Edit screen

(2) Changing the display size of the edit screen
The display size of the edit screen can be changed.

1) Click [View] → [Zoom].

The Zoom dialog box is displayed.

Change the display size according to the selected zoom ratio.

Change the display size according to the specified zoom ratio. (available range: 50 to 150%)

Adjust the width of the ladder automatically to display the entire ladder.
(3) Changing the text size on the edit screen
The text size displayed on the edit screen can be changed.

1) Select [View] → [Text Size] → [Bigger]/[Smaller].

The text size is changed one step at each setting within the range of 10 steps.

(4) Displaying/hiding comments
Device comments (label comments), notes, and statements can be displayed and hidden.

1) Select [View] → [Comment]/[Statement]/[Note].

POINT
Displaying/hiding comments
Comments also can be displayed or hidden by the following operation.
[Tool] → [Option] → "Program Editor" → "Ladder" → "Comment"

* The details of this operation are explained in the next page.
(5) Setting the number of rows and columns for displaying comments:
The option setting allows switching the number of rows and columns for displaying a device comment.

1) Click [Tool] → [Option].

The Options screen is displayed.

2) Click "Program Editor" → "Ladder" → "Comment".

The screen for setting Device Comment Display Format is displayed.

Comments can be displayed or hidden by this setting in addition to by the method described on the previous page.

(To the next page)
Set the number of display rows in the range from 1 to 4.

Set the number of display columns to 5 or 8.

Example)

4 rows × 8 columns

2 rows × 5 columns
(6) Setting the number of contacts to be displayed in ladder programs

The option setting allows switching the number of contacts to be displayed in a single row.

1) Click "Program Editor" → "Ladder" → "Ladder Diagram" in the Options screen.

The screen for setting Display Format for the ladder diagram is displayed.

Set the number of contacts to be displayed in a single row to 9 or 11 contacts.
(7) Switching the label name display and device display

The display of a program that uses labels can be switched between the label name display and device display.

If label comments or device comments are set, the corresponding comments are displayed.

Devices assigned by the compilation can be checked by switching the program display from the label name display to the device display.

1) Click [View] → [Device Display].

The screen for setting Display Format for the ladder diagram is displayed.

Example)

Label name display

Device display

POINT

Displaying/hiding label comments and device comments

To check the set label comments and device comments, set the setting to display comments. (Refer to section 2.2.2 (4))
(8) Hiding a ladder block
The ladder block after the ladder conversion can be hidden.
The ladder block in which the statements are set is hidden with the statements displayed.

(a) Hiding a ladder block

1) Move the cursor to the ladder block to be hidden.

2) Click [View] → [Non-Display Ladder Block].

(To the next page)
3) The selected ladder blocks are hidden.

(b) Canceling the hidden ladder block.

1) Move the cursor to the hidden ladder block displayed in gray.

2) Click [View] → [Display Ladder Block].
3) The hidden ladder blocks are displayed.

POINT
Displaying/hiding ladder blocks
• Multiple ladder blocks also can be displayed and hidden.
• All ladder blocks can be displayed and hidden by the operation of [View] → [Display All Ladder Block]/[Non-Display All Ladder Block].
• Ladder blocks also can be displayed and hidden by Right-click → [Displaying/hiding ladder blocks].
2.2.3 Project

This section explains the configurations of a project that is displayed in a tree format in the Project view. The display contents differ according to the programmable controller type and the project type. The following is an example for a Simple project of QCPU (Q mode).

< Simple project (without labels) >
- Set various parameters.
- Make settings for the intelligent function modules.
- Set global device comments.
- Set an execution type of each program.
- Create programs.
- Set local device comments.
- Make settings for device memory.
- Set device initial values.

< Simple project (with labels) >
- Set various parameters.
- Make settings for the intelligent function modules.
- Set global device comments.
- Set global labels.
- Set global labels.
- Set an execution type of each program.
- Create programs.
- Set local device comments.
- Make settings for device memory.
- Set device initial values.
1) One project per GX Works2
One GX Works2 can edit only one project unit.
To edit two or more projects at a time, run as many GX Works2 as the number of projects.

2) Device comments
Device comment of GX Works2 is categorized into global device comment and local device comment.

<table>
<thead>
<tr>
<th>Comment type</th>
<th>Number of comments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global comment</td>
<td>1</td>
<td>A device comment created automatically when a new project is created. Global comments are set to use common device comment data among multiple programs.</td>
</tr>
<tr>
<td>Local comment</td>
<td></td>
<td>A device comment created by the user. No local device comment exists when a new project is created. Therefore, create a local device comment if necessary. Set the same names as sequence programs.</td>
</tr>
</tbody>
</table>
2.3 Operation Before Creating Ladder Program

2.3.1 Starting up GX Works2

1) Click the button.
2) Select [All Programs].
3) Select [MELSOFT Application].
4) Select [GX Works2].
   Put the mouse cursor over the items to select the menu.
   (Clicking or double-clicking the mouse is not required.)
5) Click [GX Works2].
6) GX Works2 starts up.
2.3.2 Creating a new project

1) Click on the toolbar or select [Project] → [New Project] \(^{(\text{Ctrl} + \text{N})}\).

2) Click the "Project Type" list button.

3) The "Project Type" list is displayed. Select "Simple Project".

4) Click the "PLC Series" list button.

5) The "PLC Series" list is displayed. Select "QCPU (Q mode)".
6) Click the "PLC Type" list button.

7) The "PLC Type" list is displayed. Select "Q06UDH".

8) Click the OK button.

9) A new project is opened.
2.4 Preparation for Starting Up CPU

Setting switches and formatting the built-in memory are required before writing a program to the CPU.

Connect or set the connectors and the switches of (1) to (3) shown below. (The figures below are example of Q06UDHCPU.)

(1) Connecting a battery
   Connect the battery since the lead wire of the battery connector is disconnected at the factory shipment.

(2) Setting the switches
   Set the RUN/STOP/RESET switch to the STOP position.

(3) Connecting the USB cable
(4) Setting the connection destination

This section explains how to set the connection destination for accessing the programmable controller CPU.

1) Click "Connection Destination" in the view selection area on the navigation window.

2) The Connection Destination view is displayed. Double-click "Connection1" in Current Connection.

   The Transfer Setup dialog box is displayed.

3) Double-click "Serial USB" of PC side I/F.

4) The PC side I/F Serial Setting dialog box is displayed. Check "USB" and click the OK button.

5) Double-click "PLC Module" of PLC side I/F.
6) The PLC side I/F Detailed Setting of PLC Module dialog box is displayed. Select "QCPU (Q mode)" and click the **OK** button.

7) Click the **OK** button.
(5) Formatting the built-in memory of the CPU

This section explains how to format the program memory of the QCPU.

1) Click [Online] → [PLC Memory Operation] → [Format PLC Memory].

2) The Format PLC Memory dialog box is displayed. Select "Program Memory" from the Target Memory drop-down menu.

3) Click the Execute button.

4) Click the Yes button to start formatting.

5) When format is completed, the dialog box on the left is displayed. Click the OK button.

6) Click the Close button to close the dialog box.
(6) Clearing all the device memory from the CPU

This section explains how to clear the device memory of the QCPU.

1) Click [Online] → [PLC Memory Operation] → [Clear PLC Memory].

2) The Clear PLC Memory dialog box is displayed. Check that "Clear Device's whole Memory" is checked.

3) Check "Include Latch".

4) Click the **Execute** button.

5) Click the **Yes** button to clear the latch device.

6) When the clearing the latch device is completed, the dialog box on the left is displayed. Click the **OK** button.

7) Click the **Close** button to close the dialog box.
(7) Clearing the error history in the CPU
This section explains how to clear the error history data stored in the QCPU.

1) Click [Diagnostics] → [PLC Diagnostics].

2) The PLC Diagnostics dialog box is displayed. Click the [Clear History] button.

3) The confirmation dialog box is displayed. Click the [Yes] button.

4) Click the [Close] button to close the dialog box.
(8) Setting the clock on the programmable controller CPU

Setting the year, month, date, time, minute, second, and day of the week to the clock on the programmable controller CPU is available.

To use the clock function, use GX Works2 or a sequence program. Set or read the clock data in GX Works2.

1) Click [Online] → [Set Clock] to display the Set Clock dialog box.

2) Enter a year, month, date, time, minute, second, and day of the week in the Set Clock dialog box.

3) Click the [Execute] button.
2.5 Creating Ladder Program

2.5.1 Creating a ladder program using the function keys

Follow the steps below to create the ladder program as shown on the left.

1) Press the F5 key to open the Enter Symbol window. Enter "X2".
   If any other key is pressed by mistake, press the Esc key and retype.
2) Press the Enter key to confirm the entry.
   • The OK or Exit button also can be used to confirm or cancel the entry.
3) The entered symbol (X2) is displayed.
4) Press the Shift + F5 keys, and enter "X0".
5) Press the Enter key to confirm the entry.
6) The entered symbol (X0) is displayed.
7) Press the F7 key, and enter "Y70".
8) Press the Enter key to confirm the entry.

(To the next page)
9) The entered symbol \(\text{Y70}\) is displayed.

10) Press the \(\text{F6}\) key, and enter "Y70".

11) Press the \(\text{Enter}\) key to confirm the entry.

12) The entered symbol \(\text{Y70}\) is displayed.

13) Move the cursor to the symbol under \(\text{Y70}\).

14) Press the \(\text{F5}\) key, and enter "X3".

15) Press the \(\text{Enter}\) key to confirm the entry.

16) The entered symbol \(\text{X3}\) is displayed.

17) Press the \(\text{F7}\) key, and enter "Y71".

18) Press the \(\text{Enter}\) key to confirm the entry.

19) The entered symbol \(\text{Y71}\) is displayed.

20) The procedure is finished.
2.5.2 Creating a ladder program using the tool buttons

Follow the steps below to create the ladder program as shown on the left.

1) Click on the toolbar to open the Enter Symbol window. Enter "X2". If any other button is pressed by mistake, click the Exit button.

2) Click the OK button to confirm the entry.

3) The entered symbol (X2) is displayed.

4) Click , then enter "X0".

5) Click the OK button.

6) The entered symbol (X0) is displayed.

7) Click , then enter "Y70".

8) Click the OK button.

(To the next page)
9) The entered symbol (Y70) is displayed.

10) Click on the toolbar, and enter "Y70".

11) Click the OK button.

12) The entered symbol (Y70) is displayed.

13) Move the cursor to the symbol under Y70.

14) Click on the toolbar, and enter "X3".

15) Click the OK button.

16) The entered symbol (X3) is displayed.

17) Click on the toolbar, and enter "Y71".

18) Click the OK button.

19) The entered symbol (Y71) is displayed.

20) The procedure is finished.
2.6 Converting Program (Ladder Conversion)

1) Click [Compile] → [Build] (F4).

2) The ladder program has been converted.

If an error occurs during a conversion, the cursor will automatically move to the defective point of the ladder program. Check the point and correct the program as necessary.
2.7 Writing/Reading Data to/from Programmable Controller CPU

(1) Writing data to the CPU

1) Suppose that the ladder program (sequence program) has been created with GX Works2 to proceed to the next step.

2) Set the RUN/STOP/RESET switch on the CPU to STOP.

3) Click on the toolbar or click [Online] → [Write to PLC].

4) From the "PLC Module" tab, click to select the program and parameter to write to the CPU. Or click [Parameter + Program] to select the target program and parameter.

5) Click [Execute] to accept the selection.

6) If a parameter or program has already been written, the confirmation dialog box for overwriting the data is displayed. Click [Yes].

(To the next page)
7) The progress dialog box is displayed.

8) The message "Completed" is displayed when the writing is completed. Click **Close**.

9) Click the **Close** button to close the dialog box.
(2) Reading data from the CPU

1) Click on the toolbar or click [Online] → [Read from PLC].

2) From the "PLC Module" tab, click to select the program and parameter to read from the CPU. Or click Parameter + Program to select the target program and parameter.

Select "Program Memory/Device Memory" for "Target Memory".

3) Click Execute to accept the selection.

4) If a parameter or program exists, the confirmation dialog box for overwriting the data is displayed. Click Yes.

5) The progress dialog box is displayed.

6) The message "Completed" is displayed when the reading is completed. Click Close.
2.8 Monitoring Ladder Program Status

1) Suppose that the ladder program (sequence program) has been written into the programmable controller CPU to proceed to the next step.

2) Set the RUN/STOP/RESET switch on the CPU to RESET once (for about one sec.), return it to STOP, then set it to RUN.

3) Click ![monitor icon] on the toolbar or click [Online] → [Monitor] → [Start Monitoring].

4) Selecting another menu ends the monitor mode.

Operation Practice

1) Confirm that the LED indicator Y70 lights up by turning on the snap switch X2, and that the indicator remains lit after the snap switch is turned off.

2) Confirm that the LED indicator Y70 turns off by pressing (turning on) the push button (snap switch) X0, and that the indicator does not light up when the button (snap switch) is released (turned off).

3) Turning on the snap switch X3 turns on the LED indicator Y71.
(1) In the monitor mode, the Monitor Status dialog box shown below is displayed regardless of the monitor status.

1) Connection status
   Displays the connection status between a programmable controller CPU and personal computer in which the simulation function is started.

2) RUN/STOP status
   Displays the programmable controller CPU status operated by the key switch on the programmable controller CPU or the remote operation from GX Works2.

3) ERR. status (PLC diagnostics)
   Displays the error status of the programmable controller CPU.
   Clicking the icon displays the PLC Diagnostics screen (*1).

4) USER status (PLC diagnostics)
   Displays the user error status of the programmable controller CPU.
   Clicking the icon displays the PLC Diagnostics screen (*1).

5) Scan time
   Displays the maximum scan time of the monitored programmable controller CPU.
   The Q-series programmable controller displays the scan time in units of 0.1msec.

*1: For the PLC diagnostics, refer to section 2.8.

(2) The statuses of the ladder are indicated as shown below.

1) Display of a contact when X0 = OFF
   
   ![Diagram of normally open and closed contacts]
   Normally open contact (not conducting) Normally closed contact (conducting)

2) Display of a coil output instruction, contact-equivalent comparison instruction, and coil-equivalent instruction
   
   ![Diagram of instruction statuses]
   Not executed, conditions not established Executed, conditions established

*: Available contact-equivalent comparison and coil-equivalent instructions are SET, RST, PLS, PLF, SFT, SFTP, MC, FF, DELTA, and DELTAP.
(3) Ladder conversion during the monitoring
This section explains the procedure to convert Y70 into Y72 during the monitoring.

1) Double-click (Y70).

2) The Enter Symbol window is displayed. Enter "Y72".

3) Press the Enter key.

4) Double-click (Y70) and change "Y70" to "Y72".

5) Click [Compile] → [Build] (F4).

6) The conversion is completed.
2.9 Diagnosing Programmable Controller CPU

1) Click [Diagnostics] → [PLC Diagnostics].

2) The PLC Diagnostics screen is displayed.

3) The function menu is displayed from the PLC image.

4) The connection channel list is displayed.

5) Change the window size and position after error info.

6) Error information, PLC status information, Serial communication error.

7) Error history, error details, error history.

Clear Error Error Jump Error Clear Error
<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Monitor Status</td>
<td>Displays the current monitor status.</td>
</tr>
<tr>
<td>2) Connection Channel List</td>
<td>Displays the connection route which has been set.</td>
</tr>
</tbody>
</table>
| 3) CPU operation status | - For single CPU system
- For multiple CPU system
Displays the operation status and switch status of the programmable controller CPU.
- "Uninstallable/Blank" is displayed for a slot with no module mounted. |
| 4) Image of programmable controller CPU | Perform online operations of the programmable controller CPU. (refer to POINT) |
| 5) PLC Status Information | Select this to display the status information of the programmable controller CPU. |
| 6) Error History | Displays the latest error history by clicking the button. |
| 7) Status Icon Legend | Displays the status icons on the screen. |

**POINT**

Online operations

The PLC Memory Operation function and the Remote Operation function can be executed from the image of the programmable controller CPU.

When the cursor is moved to the image of the programmable controller CPU, the function menu is expanded. Click the image of the programmable controller CPU to display the items to be set.

![Diagram of PLC Memory Operation and Remote Operation](image-url)
2.10 Editing Ladder Program

2.10.1 Modifying a part of the ladder program

This section explains how to modify a part of the ladder program shown on the left. (OUT Y71 → OUT Y72)

1) Confirm that "Ovrwrte" is shown at the lower-right portion of the screen.

If "Insert" is shown on the screen, click the Ins key to change the display to "Ovrwrte".

If "Insert" is shown on the screen, contacts or coils are added to the diagram.

<When correcting X2 to X5>

2) Double-click the point to be corrected.

(To the next page)
3) The Enter Symbol window is displayed.

4) Click the edit box and enter "Y72".

5) Click the OK button to accept the change.

6) The modified ladder program is displayed.

7) To convert the edited ladder program, click [Compile] → [Build] (F4).
2.10.2 Drawing/deleting lines

(1) Drawing lines

This section explains how to add a line to the ladder program shown on the left.

1) Click entrances (Alt + F10) on the toolbar.

2) Drag the mouse from the start position to the end position.

A vertical line is created to the left of the cursor.

3) A line is created when the left button of the mouse is released.

(To the next page)
4) Click on the toolbar, and enter "Y73".

5) Click the OK button.

6) The entered symbol (Y73) is displayed.

7) To convert the edited ladder program, click [Compile] → [Build] (F4).
Perform the following steps to delete the line from the ladder shown on the left.

1) Click \((\text{Alt} + F9)\) on the toolbar.

2) Drag the mouse from the start position to the end position.

3) The line is deleted when the left button of the mouse is released.

4) Press the \(\text{Delete}\) key to delete \(Y73\).

5) To convert the edited ladder program, click \([\text{Compile}] \rightarrow [\text{Build}] \((F4)\).
2.10.3 Inserting/deleting rows

(1) Inserting rows

This section explains how to add a row to the ladder program shown on the left.

1) Click on any point of the row to move the cursor.

2) Right-click on any point on the ladder program creation screen to display the menu.

(To the next page)
3) Select the [Edit] → [Insert Row] (Shift + Ins).

4) A new row is inserted above the selected row.

5) Click on the toolbar to open the Enter Symbol window. Enter "X7".

6) Click the OK button to accept the entry.
7) The entered symbol (X7) is displayed.

8) Click on the toolbar, and enter "Y77".

9) Click the OK button.

10) The entered symbol (Y77) is displayed.

11) To convert the edited ladder program, click [Compile] → [Build] (F4).
This section explains how to delete the row from the ladder program shown on the left.

1) Click on any point of the row to be deleted to move the cursor.

2) Right-click on any point on the ladder program creation screen to display the menu.

(To the next page)
3) Select the [Edit] → [Delete Row] (Shift + Del).

3) Click!

4) The selected row is deleted.

4) The row is deleted!

5) To convert the edited ladder program, click [Compile] → [Build] (F4).
2.10.4 Cutting/copying ladder program

This section explains how to copy and cut the ladder program shown on the left.

1) Click on the start point of the ladder program to be cut to move the cursor.

2) Drag the mouse over the ladder to specify the area. The selected area is highlighted.
   Click the step numbers and drag the mouse vertically to specify the area in ladder block units.

3) Click \( \boxed{\text{X}} \) on the toolbar or select [Edit] \( \rightarrow \) [Cut] (Ctrl + X) to cut the specified area.
4) Click on the start point of the ladder program to be copied to move the cursor.

5) Drag the mouse over the ladder to specify the area. The selected area is highlighted. Click the step numbers and drag the mouse vertically to specify the area in ladder block units.

6) Click on the toolbar or select [Edit] → [Copy] (Ctrl + C ) to copy the specified area.

7) Click any ladder block to move the cursor to the ladder. The copied ladder is pasted above the row with the cursor.
8) Click on the toolbar or select [Edit] → [Paste] (Ctrl + V) to paste the cut or copied area.

9) The cut or copied ladder is pasted.
2.11 Verifying Data

This section explains how to verify the open project against the data on the programmable controller CPU. The verification function is used to compare the contents of two projects or to locate program changes made in the programs.

1) Click [Online] → [Verify with PLC].

2) The Online Data Operation dialog box is displayed.
   Click the Parameter + Program button.

3) Click the Execute button.

4) The verification result is displayed in the Verify Result window.
   To check the detail of the data, double-click the corresponding row.

5) The detail of the result is displayed.
2.12 Saving Ladder Program

2.12.1 Saving newly-created or overwritten projects

1) Click on the toolbar or select [Project] → [Save] (Ctrl + S).

(Saving the existing project is completed at this step.)

2) Specify the location to store the project.
3) Set a workspace name.
4) Set a project name.
5) Set a title as necessary.
6) Click the Save button to accept the entry.

(Only when a newly-created project is saved)

7) Click the Yes button.
The new project is saved.
POINT

- **Workspace**
  Workspace enables GX Works2 to manage several projects with one name.

- **When the save destination exists**
  When the save destination (workspace and project) exists, the folder where the workspace is saved can be specified in "Workspace/Project List".

- **Number of the characters for a workspace name, project name, and title**
  Specify a workspace name, project name, and title within 128 characters each. However, the total number of the characters of the save destination path name + workspace name + project name must be within 150.

2.12.2 Saving a project with another name

1) Click [Project] → [Save as].

2) Specify the location to store the project.

3) Set a workspace name.

4) Set a project name.

5) Set a title as necessary.

6) Click the **Save** button to accept the entry.

7) Click the **Yes** button.
   The new project is saved.
2.13 Reading the saved project

1) Click \(\text{Project} \rightarrow \text{Open} \) (Ctrl + O).

2) Specify the location where the project to be read is stored.

3) Double-click the workspace to be read.

4) Click the project to be read.

5) Click the button to start reading the specified project.

Each confirmation dialog box below is displayed in the following cases:

(When another project has been open)

- Yes ........ Terminates the project.
- No ........ Keeps the project open.

(When another project has been open without being converted)

- Yes ........ Terminates the project without converting the project.
- No ........ Keeps the project open.

(Continue editing the ladder program.)

(When another project has been open without being saved)

- Yes ........ Terminates the project after saving it.
- No ........ Terminates the project without saving it.
- Cancel .... Keeps the project open.
2.14 Opening Projects in Different Format

This section explains how to open a project created with GX Developer in GX Works2.

1) Click [Project] → [Open Other Data] → [Open Other Project].

2) The Open Other Project dialog box is displayed. Specify the project and click the Open button.

3) The message on the left is displayed. Click the Yes button.

4) The project created with GX Developer is read.
2.15 Saving Projects in Different Format

This section explains how to save a Simple project of GX Works2 in the GX Developer format.

1) Click [Project] → [Export to GX Developer Format File].

2) The Export to GX Developer Format File dialog box is displayed. Specify the destination to save the project.

3) Enter a project name and click the Save button.

4) The message on the left is displayed. Click the Yes button.

5) The project is saved in the GX Developer format.
### 3.1 Device

A device is an imaginary element for programming in the programmable controller CPU, as well as the components (such as contacts and coils) that compose a program.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Input Sends commands and data to a programmable controller through external devices such as push buttons, selector switches, limit switches, and digital switches.</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>Output Outputs control results to solenoids, electromagnetic switches, signal lights, and digital indicators.</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Internal relay Auxiliary relay inside a programmable controller that cannot output directly to external devices.</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>Latch relay Uninterruptible auxiliary relay inside the programmable controller that cannot output directly to external devices.</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Link relay Internal relay for data link that cannot output directly to external devices. The area not assigned by initial link information setting can be used as an internal relay.</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Annunciator Used for failure detection. Create a failure detection program beforehand and turn on the program while the programmable controller is running to store numerical values in the special register D.</td>
<td>Bit device • Mainly handles on/off signals.</td>
</tr>
<tr>
<td>V</td>
<td>Edge relay Internal relay that stores an operation result (on/off information) from the top of a circuit block.</td>
<td></td>
</tr>
<tr>
<td>SM</td>
<td>Special relay Internal relay that stores CPU statuses.</td>
<td></td>
</tr>
<tr>
<td>SB</td>
<td>Special link relay Internal relay for data link that indicates a communication status and errors.</td>
<td></td>
</tr>
<tr>
<td>FX</td>
<td>Function input Internal relay that captures the on/off data specified by a subroutine call instructions with arguments in a subroutine program.</td>
<td></td>
</tr>
<tr>
<td>FY</td>
<td>Function output Internal relay that passes an operation result (on/off data) in a subroutine program to a subroutine program call source.</td>
<td></td>
</tr>
<tr>
<td>T(ST)</td>
<td>Timer Accumulative timers of four types: low-speed timer, high-speed timer, low-speed integrator, and high-speed integrator.</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Counter Accumulative counters of two types: counters for sequence programs and counters for interruption sequence programs.</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Data register Memory that stores data in the programmable controller.</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>Link register Data register for data link.</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>File register Register for an extensive use of data registers, which uses the standard RAM or memory card.</td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>Special register Register that stores CPU statuses.</td>
<td></td>
</tr>
<tr>
<td>SW</td>
<td>Link data register Data register for data link that stores a communication status and failure information.</td>
<td></td>
</tr>
<tr>
<td>FD</td>
<td>Function register Register for the exchange data between a subroutine call source and a subroutine program.</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>Index register Register for modification to the devices (X, Y, M, L, B, F, T, C, D, W, R, K, H, and P)</td>
<td></td>
</tr>
</tbody>
</table>

Bit device • Word device

Word device • Mainly handles data. • One word consists of 16 bits. • Can be specified by entering ~.* (* = 0 to F (hexadecimal)).
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD</td>
<td>Function register</td>
<td>Register for the exchange data between a subroutine call source and a subroutine program.</td>
</tr>
<tr>
<td>N</td>
<td>Nesting</td>
<td>Shows the nesting (nested structure) of the master control.</td>
</tr>
<tr>
<td>P</td>
<td>Pointer</td>
<td>Locates the jump addresses of the branch instructions (CJ, SCJ, CALL and JMP).</td>
</tr>
<tr>
<td>I</td>
<td>Interruption pointer</td>
<td>Locates a jump address that corresponds to the factor of the interruption when an interruption occurs.</td>
</tr>
<tr>
<td>J</td>
<td>Network No. specification device</td>
<td>Used to specify the network number in the data link instructions.</td>
</tr>
<tr>
<td>U</td>
<td>I/O No. specification device</td>
<td>Used to specify the I/O number in the intelligent function module dedicated instructions.</td>
</tr>
<tr>
<td>K</td>
<td>Decimal constant</td>
<td>Used to specify the following; timer counter set value, pointer number, interruption pointer number, number of digits of bit device, and basic/application instruction values.</td>
</tr>
<tr>
<td>H</td>
<td>Hexadecimal constant</td>
<td>Used to specify the basic/application instruction values.</td>
</tr>
<tr>
<td>E</td>
<td>Real number constant</td>
<td>Used to specify real numbers as instructions.</td>
</tr>
<tr>
<td>&quot;Character String&quot;</td>
<td>Character string constant</td>
<td>Used to specify character strings as instructions.</td>
</tr>
<tr>
<td>Jn\X</td>
<td>Link direct device</td>
<td>Device that can access directly to a link device of a network module (The refresh parameter setting is not required.).</td>
</tr>
<tr>
<td>Jn\Y</td>
<td>Link direct device</td>
<td></td>
</tr>
<tr>
<td>Jn\B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jn\SB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jn\W</td>
<td>Intelligent function module device</td>
<td>Device that can access directly to the buffer memory of an intelligent function module.</td>
</tr>
<tr>
<td>Jn\SW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Un\G</td>
<td></td>
<td>Bit device Mainly handles on/off signals.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Word device Mainly handles data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>One word consists of 16 bits.</td>
</tr>
</tbody>
</table>
### 3.2 Parameter

The parameters are basic setting values applied to a programmable controller in order to control objects as planned. The parameters are divided into the PLC parameter, network parameter, and remote password as shown below.

* A shaded area in the following table indicates the items to be set in this textbook.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PLC name</strong></td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>Sets a label (name and application) of a programmable controller CPU.</td>
</tr>
<tr>
<td>Comment</td>
<td>Sets a comment for the label of a programmable controller CPU.</td>
</tr>
<tr>
<td><strong>Timer limit setting</strong></td>
<td>Sets the time limit of the low-speed or high-speed timer.</td>
</tr>
<tr>
<td><strong>RUN-PAUSE contacts</strong></td>
<td>Sets contacts for controlling RUN and PAUSE of a programmable controller CPU.</td>
</tr>
<tr>
<td><strong>Latch data backup operation valid contact</strong></td>
<td>Sets contact devices in order to execute the latch data backup operation. (Only for Universal model QCPU)</td>
</tr>
<tr>
<td>Remote reset</td>
<td>Sets whether to allow a remote reset operation from GX Works2.</td>
</tr>
<tr>
<td><strong>Output mode at STOP to RUN</strong></td>
<td>Sets the status of an output (Y) when the programmable controller is switched from STOP to RUN.</td>
</tr>
<tr>
<td><strong>Floating point arithmetic processing</strong></td>
<td>Sets whether to execute floating-point processing in double precision. (Only for high performance model QCPU)</td>
</tr>
<tr>
<td><strong>Intelligent function module setting</strong></td>
<td>Sets the interruption pointer assignment of a module.</td>
</tr>
<tr>
<td></td>
<td>Sets the start I/O number and start SI number.</td>
</tr>
<tr>
<td><strong>Common pointer No.</strong></td>
<td>Sets the start number of the pointer used as a common pointer.</td>
</tr>
<tr>
<td><strong>Points occupied by empty slot</strong></td>
<td>Sets the number of empty slots for the main or extension base unit.</td>
</tr>
<tr>
<td><strong>System interrupt settings</strong></td>
<td>Sets the start number of the interrupt counters.</td>
</tr>
<tr>
<td></td>
<td>Sets the execution interval for the interrupt pointers.</td>
</tr>
<tr>
<td><strong>Interrupt program/fixed scan program setting</strong></td>
<td>Set whether to execute high-speed execution of an interrupt program.</td>
</tr>
<tr>
<td><strong>Module synchronization</strong></td>
<td>Set whether to synchronize the start-up of the programmable controller CPU with that of the intelligent function module.</td>
</tr>
<tr>
<td><strong>A-PLC compatibility setting</strong></td>
<td>Set whether to use the MELSEC-A series special relays/special registers.</td>
</tr>
<tr>
<td><strong>Service processing setting</strong></td>
<td>Sets the processing time and the number of times of service processing. (Only for Universal model QCPU)</td>
</tr>
<tr>
<td><strong>PLC module change setting</strong></td>
<td>Set this parameter to replace the CPU module using a memory card (Only for Universal model QCPU).</td>
</tr>
<tr>
<td><strong>File register</strong></td>
<td>• Sets the file register file to be used in a program.</td>
</tr>
<tr>
<td></td>
<td>• Sets whether to transfer data to the standard ROM at a latch data backup operation. (Only for Universal model QCPU)</td>
</tr>
<tr>
<td><strong>Comment file used in a command</strong></td>
<td>Sets the device comment file to be used in a program. (Only for Universal model QCPU)</td>
</tr>
<tr>
<td><strong>Initial device value</strong></td>
<td>Sets the device initial value file to be used on the programmable controller CPU.</td>
</tr>
<tr>
<td><strong>File for local device</strong></td>
<td>Sets the local device file to be used in a program.</td>
</tr>
<tr>
<td><strong>File used for SP.DEVST/S.DEVLD instruction</strong></td>
<td>Sets the device data ROM write/read instruction file to be used in a program. (Only for Universal model QCPU)</td>
</tr>
<tr>
<td><strong>WDT (watchdog timer) setting</strong></td>
<td>Sets the WDT of the programmable controller CPU.</td>
</tr>
<tr>
<td><strong>Error check</strong></td>
<td>Sets whether to detect specified errors.</td>
</tr>
<tr>
<td><strong>Operation mode when there is an error</strong></td>
<td>Sets the programmable controller CPU operation mode when an error is detected.</td>
</tr>
<tr>
<td><strong>Constant scanning</strong></td>
<td>Sets the constant scan time.</td>
</tr>
<tr>
<td><strong>Breakdown history</strong></td>
<td>Sets the storage destination for error histories of the programmable controller CPU. (Only for high performance model QCPU)</td>
</tr>
<tr>
<td><strong>Low speed program execution time</strong></td>
<td>Sets the execution time of a low-speed program in every scan. (Only for high performance model QCPU)</td>
</tr>
<tr>
<td>Item</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Boot file</strong></td>
<td></td>
</tr>
<tr>
<td>Boot option</td>
<td>Sets whether to clear the program memory when booting up.</td>
</tr>
<tr>
<td>Boot file setting</td>
<td>Sets the type, data name, transfer source drive, and transfer destination drive of the boot file.</td>
</tr>
<tr>
<td><strong>Program</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sets the file name and execution type (execution condition) of the program when several programs are written to the programmable controller CPU.</td>
</tr>
<tr>
<td><strong>SFC</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sets the startup mode and startup condition of an SFC program and the output mode at block stop.</td>
</tr>
<tr>
<td><strong>Device</strong></td>
<td></td>
</tr>
<tr>
<td>Device points</td>
<td>Sets the number of points used for each device of the programmable controller CPU.</td>
</tr>
<tr>
<td>Latch (1) start/end</td>
<td>Sets the latch range (start device number/end device number) clearable with the RESET/L.CLR switch or a remote latch clear operation.</td>
</tr>
<tr>
<td>Latch (2) start/end</td>
<td>Sets the latch range (start device number/end device number) not clearable with the RESET/L.CLR switch or a remote latch clear operation.</td>
</tr>
<tr>
<td>Local device start/end setting</td>
<td>Sets the range (start device number/end device number) of devices used as a local device.</td>
</tr>
<tr>
<td>File register extended setting</td>
<td>Sets the extended data register and extended link register. (Only for Universal model QCPU)</td>
</tr>
<tr>
<td>Indexing setting for ZR device</td>
<td>Sets the number of Z to be 32-bit indexed, or use the index register ZZ for 32-bit index setting. (Only for Universal model QCPU)</td>
</tr>
<tr>
<td><strong>I/O assignment</strong></td>
<td></td>
</tr>
<tr>
<td>I/O assignment</td>
<td>Sets the type, model, number of occupied I/O points, and start I/O number of each module mounted on the base unit.</td>
</tr>
<tr>
<td>Basic setting</td>
<td>Sets the model and the number of slots of the base unit, the model of the power supply module, and the model of the extension cable.</td>
</tr>
<tr>
<td><strong>Multiple CPU setting</strong></td>
<td></td>
</tr>
<tr>
<td>No. of PLC</td>
<td>Sets the number of programmable controller CPUs used in the multiple CPU system.</td>
</tr>
<tr>
<td>Operation mode</td>
<td>Sets the operation mode of the multiple CPU system when a stop error occurs in any of the programmable controller CPU No. 2 to No. 4.</td>
</tr>
<tr>
<td>Host station</td>
<td>Sets the CPU number for the host CPU.</td>
</tr>
<tr>
<td>Multiple CPU synchronous startup setting</td>
<td>Selects the CPU modules to be started up synchronously.</td>
</tr>
<tr>
<td>Online module change</td>
<td>Sets whether to allow the online module change in the multiple CPU system.</td>
</tr>
<tr>
<td>I/O sharing when using multiple CPUs</td>
<td>Sets whether to retrieve the I/O status of the I/O module or intelligent function module controlled by other programmable controller CPUs.</td>
</tr>
<tr>
<td>Communication area setting (refresh setting)</td>
<td>Sets the CPU shared memory to enable data sharing among multiple CPUs.</td>
</tr>
<tr>
<td>Multiple CPU high speed transmission area setting</td>
<td>Sets the user setting area, auto refresh, assignment confirmation, and system area.</td>
</tr>
<tr>
<td><strong>Built-in Ethernet port setting</strong></td>
<td></td>
</tr>
<tr>
<td>IP address setting</td>
<td>Sets the IP address and the input format of the IP address.</td>
</tr>
<tr>
<td>Communication data code</td>
<td>Selects the Binary code or ASCII code for communication.</td>
</tr>
<tr>
<td>Open setting button</td>
<td>Sets the protocol, open system, and host station port number.</td>
</tr>
<tr>
<td>FTP setting button</td>
<td>Selects whether to use the FTP function.</td>
</tr>
<tr>
<td>Time setting button</td>
<td>Sets whether to use the SNTP function and the timing of setting the time.</td>
</tr>
<tr>
<td><strong>Serial communication</strong></td>
<td></td>
</tr>
<tr>
<td>Transmission speed</td>
<td>Sets the transmission speed.</td>
</tr>
<tr>
<td>Sum check</td>
<td>Sets the sum check.</td>
</tr>
<tr>
<td>Transmission wait time</td>
<td>Sets the transmission wait time.</td>
</tr>
<tr>
<td>Online change</td>
<td>Sets whether to allow the online program change.</td>
</tr>
<tr>
<td><strong>Network parameter</strong></td>
<td></td>
</tr>
<tr>
<td>Ethernet/CC IE/MELSECNET</td>
<td>Sets the network parameters for Ethernet, MELSECNET/I0, MELSECNET/H, and CC-Link IE controller network.</td>
</tr>
<tr>
<td>CC-Link</td>
<td>Sets the parameters for CC-Link.</td>
</tr>
<tr>
<td>Remote password</td>
<td>Sets the password that limits the access via the Ethernet or serial communication modules.</td>
</tr>
</tbody>
</table>
• When GX Works2 starts, it employs the preset values as the parameters. These values are called the default (initial values).
• The programmable controller can run with those values unchanged, however, modify them within a specified range as necessary.

**Operation example: Changing the operation mode when an error exists**

When a computation error is caused, the programmable controller CPU changes to the STOP status at the default value, however, changing the parameters continues the programmable controller CPU to run.

**Computation error example**
• In the division instruction, the processing to divide by 0 is executed.

1) Double-click "PLC parameter" on the navigation window.

2) The Q Parameter Setting dialog box is displayed. Click the "PLC RAS" tab.

3) Change the setting of "Computation Error" in "Operation Mode When There Is an Error" to "Continue"

4) Click the End button.
4.1 List of Instruction Explained in this Chapter

This chapter explains the sequence instructions and basic instructions as shown below.

<table>
<thead>
<tr>
<th>Instruction symbol (Name)</th>
<th>Function</th>
<th>Drawing (devices to be used)</th>
<th>Instruction symbol (Name)</th>
<th>Function</th>
<th>Drawing (devices to be used)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OUT</strong> Out</td>
<td>Coil output</td>
<td><img src="image1" alt="OUT Diagram" /></td>
<td><strong>CJ</strong></td>
<td>Conditional jump (non-delay)</td>
<td><img src="image2" alt="CJ Diagram" /></td>
</tr>
<tr>
<td><strong>MC</strong> Master control</td>
<td>Starting master control*1</td>
<td><img src="image3" alt="MC Diagram" /></td>
<td><strong>SCJ</strong></td>
<td>Conditional jump (non-delay)</td>
<td><img src="image4" alt="SCJ Diagram" /></td>
</tr>
<tr>
<td><strong>MCR</strong> Master control reset</td>
<td>Terminating master control</td>
<td><img src="image5" alt="MCR Diagram" /></td>
<td><strong>CALL</strong></td>
<td>Calling subroutine program</td>
<td><img src="image6" alt="CALL Diagram" /></td>
</tr>
<tr>
<td><strong>SET</strong> Set</td>
<td>Setting devices</td>
<td><img src="image7" alt="SET Diagram" /></td>
<td><strong>CALLP</strong></td>
<td>Calling subroutine program (pulsing operation)</td>
<td><img src="image8" alt="CALLP Diagram" /></td>
</tr>
<tr>
<td><strong>RST</strong> Reset</td>
<td>Resetting devices</td>
<td><img src="image9" alt="RST Diagram" /></td>
<td><strong>RET</strong></td>
<td>Returning from a subroutine program</td>
<td><img src="image10" alt="RET Diagram" /></td>
</tr>
<tr>
<td><strong>PLS</strong> Pulse</td>
<td>Pulse Generating the pulses for one program cycle when a signal turns off</td>
<td><img src="image11" alt="PLS Diagram" /></td>
<td><strong>FEND</strong></td>
<td>Terminating a main routine program</td>
<td><img src="image12" alt="FEND Diagram" /></td>
</tr>
<tr>
<td><strong>PLF</strong> Pulf</td>
<td>Pulse Generating the pulses for one program cycle when a signal turns off</td>
<td><img src="image13" alt="PLF Diagram" /></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*1: In GX Works2, the on/off status of the master control is displayed in the title tag on the monitor screen.
"Introduction: PLC Course" covers the instructions shown below. The conventional A series also support them. Refer to "MELSEC-Q/L Programming Manual Common Instruction" for more details.

<table>
<thead>
<tr>
<th>Instruction symbol (Name)</th>
<th>Function</th>
<th>Drawing (devices to be used)</th>
<th>Instruction symbol (Name)</th>
<th>Function</th>
<th>Drawing (devices to be used)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD Load</td>
<td>Starting a logical operation (Starting to operate a normally open contact)</td>
<td><img src="image" alt="LD Diagram" /></td>
<td>MRD Lead Intermediate branching</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDI Load inverse</td>
<td>Starting a logical inverse operation (Starting to operate a normally closed contact)</td>
<td><img src="image" alt="LDI Diagram" /></td>
<td>MPP Pop Terminating branching</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AND And</td>
<td>Logical AND operation (Series connection of normally open contacts)</td>
<td><img src="image" alt="AND Diagram" /></td>
<td>NOP Nop Ignored For a space or deleting a program</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANI And inverse</td>
<td>Logical AND inverse operation (Series connection of normally closed contacts)</td>
<td><img src="image" alt="ANI Diagram" /></td>
<td>END End END processing of terminating a program Must be used as an end of a program</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OR Or</td>
<td>Logical OR operation (Parallel connection of normally open contacts)</td>
<td><img src="image" alt="OR Diagram" /></td>
<td>STOP Stopping operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORI Or inverse</td>
<td>Logical OR inverse operation (Parallel connection of normally closed contacts)</td>
<td><img src="image" alt="ORI Diagram" /></td>
<td>SFT Shift 1-bit shift for devices Specifies a bit of a bit device or word device</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANB And block</td>
<td>AND operation between logical blocks (Series connection of blocks)</td>
<td><img src="image" alt="ANB Diagram" /></td>
<td>SFTP Shift P 1-bit shift for devices (pulsing operation) Specifies a bit of a bit device or word device</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORB Or block</td>
<td>OR operation between logical blocks (Parallel connection of blocks)</td>
<td><img src="image" alt="ORB Diagram" /></td>
<td>NOPLF Ignored (for a page break at printing)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPS Push</td>
<td>Starting a branch</td>
<td><img src="image" alt="MPS Diagram" /></td>
<td>PAGE PAGE n Ignored (Recognized as zero step of n-page)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
<List of instructions not explained in this chapter: part 2>
The instructions listed below are intended for the Q series and not supported by the A series.
Some of them are explained in "Q Programming Applied Course".
Refer to "MELSEC-Q/L Programming Manual Common Instruction" for more details.

<table>
<thead>
<tr>
<th>Instruction symbol (Name)</th>
<th>Function</th>
<th>Drawing (devices to be used)</th>
<th>Instruction symbol (Name)</th>
<th>Function</th>
<th>Drawing (devices to be used)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LDP</strong></td>
<td>Load P</td>
<td>Starting to operate a rising pulse</td>
<td></td>
<td><strong>MEF</strong></td>
<td>Converting a result into a falling pulse</td>
</tr>
<tr>
<td></td>
<td>Specifies a bit of a bit device or word device.</td>
<td></td>
<td></td>
<td></td>
<td>Specifies a bit of a bit device or word device.</td>
</tr>
<tr>
<td><strong>LDL</strong></td>
<td>Load F</td>
<td>Starting to operate a falling pulse</td>
<td></td>
<td><strong>INV</strong></td>
<td>Inverting the operation results</td>
</tr>
<tr>
<td></td>
<td>Specifies a bit of a bit device or word device.</td>
<td></td>
<td></td>
<td></td>
<td>Specifies a bit of a bit device or word device.</td>
</tr>
<tr>
<td><strong>ANDP</strong></td>
<td>And P</td>
<td>Series connection of rising pulses</td>
<td></td>
<td><strong>EGP</strong></td>
<td>Converting a result into a rising pulse (Memorized by Vn)</td>
</tr>
<tr>
<td></td>
<td>Specifies a bit of a bit device or word device.</td>
<td></td>
<td></td>
<td></td>
<td>Specifies a bit of a bit device or word device.</td>
</tr>
<tr>
<td><strong>ANDF</strong></td>
<td>And F</td>
<td>Series connection of falling pulses</td>
<td></td>
<td><strong>EGF</strong></td>
<td>Converting a result into a falling pulse (Memorized by Vn)</td>
</tr>
<tr>
<td></td>
<td>Specifies a bit of a bit device or word device.</td>
<td></td>
<td></td>
<td></td>
<td>Specifies a bit of a bit device or word device.</td>
</tr>
<tr>
<td><strong>ORP</strong></td>
<td>Or P</td>
<td>Parallel connection of rising pulses</td>
<td></td>
<td><strong>FF</strong></td>
<td>Inverting a device output</td>
</tr>
<tr>
<td></td>
<td>Specifies a bit of a bit device or word device.</td>
<td></td>
<td></td>
<td></td>
<td>Specifies a bit of a bit device or word device.</td>
</tr>
<tr>
<td><strong>ORF</strong></td>
<td>Or F</td>
<td>Parallel connection of falling pulses</td>
<td></td>
<td><strong>DELTA</strong></td>
<td>Converting a direct output into a pulse</td>
</tr>
<tr>
<td></td>
<td>Specifies a bit of a bit device or word device.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>MEP</strong></td>
<td>Converting a result into a rising pulse</td>
<td></td>
<td></td>
<td><strong>DELTA P</strong></td>
<td>Converting a direct output into a pulse</td>
</tr>
<tr>
<td></td>
<td>Specifies a bit of a bit device or word device.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.2 Differences between **OUT** and **SET / RST**

**OUT instruction**

- The OUT instruction turns the specified device on when the input condition turns on, and turns the device off when the condition turns off.

[Timing chart]

**SET/RST instruction**

- The SET instruction turns the specified device on when the input condition turns on, and holds the on status of the device even when the condition turns off. To turn off the device, use the RST instruction.

[Timing chart]
4.3 Measuring Timer

<table>
<thead>
<tr>
<th>Project name</th>
<th>QB-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

- The timer contact operates delaying by a set time after the coil is energized. (On delay timer)
- The timer setting range is from K1 to K32767.
  - Low-speed (100ms) timer: 0.1 to 3276.7sec.
  - High-speed (10ms) timer: 0.01 to 327.67sec.
- When the timer setting value is set to 0, it turns on (time-out) by the execution of the instruction.

- The following four types of timer are available.

<table>
<thead>
<tr>
<th>Type</th>
<th>Timer No. (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low-speed timer</td>
<td>Counts time in units of 100ms.</td>
</tr>
<tr>
<td>High-speed timer</td>
<td>Counts time in units of 10ms.</td>
</tr>
<tr>
<td>Low-speed retentive timer</td>
<td>Accumulates time in units of 100ms.</td>
</tr>
<tr>
<td>High-speed retentive timer</td>
<td>Accumulates time in units of 10ms.</td>
</tr>
</tbody>
</table>

- Change the output instruction (OUT) to OUTH to select the high-speed timer or high-speed retentive timer.
- To use the retentive timer, set the device points for the retentive timer in the device setting of the PLC parameter.

Refer to section 6.4 for explanation on the retentive timers.
4.4 Counting by Counter

<table>
<thead>
<tr>
<th>Project name</th>
<th>QB-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

- **Set value in counter**
  - X1
  - C20
  - Y72

- **RST C20**
  - X7

*: OUT C is a 4-step instruction.

[Timing chart]

- Contact X1
- Coil C20
- Contact C20, coil Y72
- Contact X7 (input of RST instruction)

- The counter counts when an input signal rises.
- After the count, the subsequent input signals are not counted.
- Once the counter counts, the contact status and the current counter value do not change until the RST instruction is executed.
- Executing the RST instruction before the count returns the counter to 0.
- The counter setting range is from K0 and K32767. (K0 turns on (counts up) by the execution of the instruction.)

- In addition to the direct specification using K, indirect specification using D (data register) is available.

- The counter C30 counts when the number of rising edges on the input signal X0 becomes the same as the number (such as 24) specified by the data register D10.
- This indirect specification is useful for applying a value specified with an external digital switch to the counter.

- The indirect specification using the data register D is also available for the timer.
Ladder example
When the conveyor belt operation start switch (X0) is turned on, the buzzer (Y70) beeps for three seconds and the conveyor belt (Y71) starts to operate. The conveyor belt automatically stops when the sensor (X1) detects that six packages have passed through.

Create the following ladder and check that it operates properly.

```
0  X0  C0  <M0  > During operation
M0  Y71  Y70  K30  > Buzzer
4  T0  Y71  T0  > 3-sec. timer
7  X1  C0  K6  > Operating the conveyor
12  Y71  RST  C0  > Counter for counting the number of packages
14
19

Sensor (X1)
```
Operating Procedure

(1) Creating a new project

(a) Click the toolbar.

(b) The New Project dialog box is displayed. Set "Project Type" to "Simple Project", "PLC Series" to "QCPU (Q mode)", and "PLC Type" to "Q06UDH". Then click the OK button.

(c) If the project in preparation exists, the confirmation dialog box for saving the project is displayed. Click the No button.

(d) The screen shifts to the new project creation mode.
(2) Creating a program

[Using the keyboard]

F5 X 0 Shift + F5 C 0 F7 M 0

Conversion

[Using the tool buttons]

(a) Click on the toolbar to open the Enter Symbol window.

(b) Enter "X0" with the keyboard and click the OK button.

(c) Click on the toolbar to open the Enter Symbol window.

(d) Enter "C0" with the keyboard and click the OK button.

(e) Click on the toolbar to open the Enter Symbol window.

(f) Enter "M0" with the keyboard and click the OK button.

(g) When creating the circuit is finished, click [Compile] → [Build].
(3) Writing the project to the programmable controller

(a) Write the created ladder to the memory on the programmable controller.

Set the RUN/STOP switch of the CPU to STOP.

Click on the toolbar. The Online Data Operation dialog box is displayed.

(b) Click the Parameter + Program button. Checkboxes for the target program and the target parameter displayed in the window are automatically marked (✓).

(c) Click the Execute button.
(d) If parameters have been already written, the confirmation dialog box for overwriting the parameters is displayed. Click the **Yes** button.

![Parameter overwrite confirmation dialog box]

(e) The Write to PLC dialog box is displayed.

![Write to PLC dialog box]

(f) If a program has already been written, the confirmation dialog box for overwriting the program is displayed. Click the **Yes** button.

![Program overwrite confirmation dialog box]
(g) Writing the program to the programmable controller is finished.
(4) Monitoring the ladder
Monitor the ladder.

Hold the RESET/STOP/RUN switch on the CPU at the RESET position for one second or more, then set the switch to RUN.

(a) Click on the toolbar.

(b) The ladder (write) screen is used to monitor the ladder.

**Operation Practice**

1) Turning on the push button switch (X0) turns on Y70 and starts T0 at the same time.

2) When the timer T0 counts three seconds (time-out), Y70 turns off and Y71 turns on at the same time.

3) Turn on or off (push or release) the push button switch (X1). The counter C0 counts the number of ON to turn off Y71 after counting on six times.
4.5 **PLS** Pulse (turns on the specified device for one scan at rising edge of an input condition.)

**PLF** Pulf (turns on the specified device for one scan at falling edge of an input condition.)

<table>
<thead>
<tr>
<th>Project name</th>
<th>QB-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

- The PLS instruction turns on the specified device only for one scan when the execution command is turned on from off.

- The PLF instruction turns on the specified device only for one scan when the execution command is turned off from on.
**Application**

- The instructions can be used in the standby program that waits for the operation condition.

![Execution command diagram]

**[Timing chart]**

<table>
<thead>
<tr>
<th>Execution command</th>
<th>X0</th>
<th>M0</th>
<th>PL5</th>
<th>M5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution condition</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

![Timing chart diagram]

- **Applicable device**

<table>
<thead>
<tr>
<th>Internal device (system or user)</th>
<th>File register</th>
<th>MELSENET/10 (H) Direct J n</th>
<th>Intelligent function module</th>
<th>Index register</th>
<th>Constant</th>
<th>Pointer</th>
<th>Level</th>
<th>Digit</th>
<th>Number of basic steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>Word</td>
<td>R</td>
<td>Bit</td>
<td>Word</td>
<td>UnG</td>
<td>Z</td>
<td>K</td>
<td>H</td>
<td>P</td>
</tr>
<tr>
<td>PLS</td>
<td>⑤</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>PLF</td>
<td>②</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>
• The instructions can be used for a program that detects passage of moving objects. After the passage of a product is detected, the next process for the product is started.

\[\text{Timing chart}\]

\(X_0\)

\(M_0\)

\(Y_{70}\)

---

**Other Useful Ways of PLS and PLF**

Part 1

• The instructions can be used for a program that executes the output operation for a set period of time when the input signal changes from on to off.

\[\text{Timing chart}\]

Input (\(X_0\))

Output (\(Y_{76}\))

Set time limit 10sec. Pulse duration

---

**Program example**

<table>
<thead>
<tr>
<th>Project name</th>
<th>QB-6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

\[\begin{array}{c}
0 & X_0 \\
3 & M_1 & T_{16} \\
& Y_{76} \\
\end{array}\]

\[\begin{array}{c}
& \text{PLF} & \text{M}_1 \\
& K_{100} & T_{16} \\
& Y_{76} \\
\end{array}\]
Other Useful Ways of PLS and PLF  Part 2

- The program for the repeated operation such as switching on/off status alternately by pressing the push button switch can be made with the instructions.

If the PLS instruction is used in the above program, the rising edge caused when the push button switch is pressed triggers the program. If the PLF instruction is used, the falling edge caused when the switch is released is the trigger.

[Timing chart]

[Program example]
Create the following ladder and check that it operates properly.

```
   0   X2   Y70
   1 M0   X0   Y70
   2
   3   Y70
   4 X3   M1   Y71
   5
   6   X1   Y71
   7
   8   X0
   9   X3
  10  M1
```

**REFERENCE**

The following is a timing chart of a lockup ladder programmed using the OUT instruction. Compare this with the lockup ladder created using the PLS instruction.

```
X2
Y70
X0
```

```
X2
Y70
X0
```
Operating Procedure
The following procedures are the same as the Operating Procedure in section 4.4.

1. Creating a new project
2. Creating a program
3. Writing the project to the programmable controller
4. Monitoring the ladder

Operation Practice
- Turning on X2 turns on Y70, and turning on X0 turns off Y70. (Even when X2 stays on, turning on X0 turns off Y70.)
- Turning on X3 turns on Y71, and turning on X1 turns off Y71.

Related Exercise --- Exercise 3

REMARK
Input pulse processing is not required for the QCPU as it uses a derivative contact (\( \frac{d}{dt} \)).

[For A/AnSCPU]

\[
\begin{array}{cccc}
X0 & & & PLS \quad M0 \\
M0 & & & SET \quad M5
\end{array}
\]

[For QCPU]

\[
\begin{array}{cccc}
X0 & & & SET \quad M5
\end{array}
\]

Supported instructions are; LDP, LDF, ANDP, ANDF, ORP, and ORF.
4.6 **MC** Master Control (Start)

**MCR** Master Control Reset (End)

---

**Application**

• The above program is a basic one.

• **MC N M** to MCR N** (indicated as "MC to MCR" hereafter.)

  The available nesting (N) numbers for "MC to MCR" are from N0 and N14.

• The scan time skipped by "MC to MCR" hardly changes.

  The device status of the program skipped by "MC to MCR" becomes as follows;
  - All the devices in the OUT instruction are turned off.
  - The devices in the SET, RST, and SFT instructions, the counter, and retentive timer keep their statuses.
  - The 100ms timer and 10ms timer are reset to 0.

---

<table>
<thead>
<tr>
<th>Applicable device</th>
<th>Internal device (system or user)</th>
<th>File register</th>
<th>MELSECNET/10 (H) Direct J n</th>
<th>Intelligent function module Un(G)</th>
<th>Index register</th>
<th>Constant</th>
<th>Pointer</th>
<th>Level</th>
<th>Digit</th>
<th>Number of basic steps</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC n</strong></td>
<td>Bit</td>
<td>Word</td>
<td>R</td>
<td>Bit</td>
<td>Word</td>
<td>Z</td>
<td>K</td>
<td>H</td>
<td>P</td>
<td>I</td>
</tr>
<tr>
<td><strong>MCR n</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

The number of basic steps of the MC instruction is two, and that of the MCR instruction is one.
Nested "MC to MCR" Program Example

• The MC and MCR instructions can be nested as shown below.

<table>
<thead>
<tr>
<th>Project name</th>
<th>QB-9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

![Diagram of nested MC to MCR program]

- The "MC to MCR" program (a) is nested under the "MC to MCR" program (b). (It is called "nested structure"). In this case;
  1) Assign the nesting number (N) of the MC instructions in ascending order.
  2) Assign the nesting number (N) of the MCR instructions used for the MC in descending order.

- The "MC to MCR" program (a) can be independent from the (c) program. The same nesting numbers (N) can be used in the both programs.
  • The internal relay number (M) must be changed for each MC instruction.

- As shown in the (c) program, the internal relay number M of MC can be used as a contact.

Note) In GX Works2, the on/off status of the master control is displayed in the title tag on the monitor screen.
Ladder example
The following program switches between manual and automatic operations using the MC and MCR instructions.

- When the manual operation is selected by turning off X7:
  1) Turning X2 sets the system to the low-speed operation mode.
  2) Turning X3 sets the system to the high-speed operation mode.

- When the automatic operation is selected by turning on X7, the system operates in the low-speed mode for 3sec. after X0 is turned on. Then the system operates in the high-speed mode for 10sec. and stops.

Note) In GX Works2, the on/off status of the master control is displayed in the title tag on the monitor screen.
Operating Procedure

The following procedures are the same as the Operating Procedure in section 4.4.

1. Creating a new project
2. Creating a program
3. Writing the project to the programmable controller
4. Monitoring the ladder

Operation Practice

• The manual operation is selected by turning off the X7 switch. When the X2 switch is turned on, Y71 lights and the low-speed operation is executed. To select the high-speed operation, turn on the X3 switch. Y72 lights and the high-speed operation starts.

• The automatic operation is selected by turning on the X7 switch. When the X0 switch is turned on, Y70 lights indicating that the automatic operation is activated. At the same time, Y71 also lights for 3sec. indicating the system is in the low-speed mode. After the 3sec. have elapsed, Y72 lights for 10sec. indicating that the system is in high-speed mode. Then the operation is stopped. (Y70, Y71, and Y72 have stopped lighting at the end.)

NOTE

For the MCR instructions in one nested program block, all master controls in the program can be terminated with the lowest nesting (N) number only.
4.7 FEND  CJ  SCJ  CALL  RET

4.7.1 FEND Fend

FEND is a 1-step instruction.

- Use the FEND instruction as the END instruction under the following conditions;
  1) When a sequence program must be executed and terminated in each program block.
     For example, use the FEND instruction with the CJ and SCJ instructions.
  2) When using subroutine programs (CALL and RET instructions).
  3) When using an interrupt program.
- After each execution of the FEND instruction, the programmable controller processes the current value of the timer and counter and executes self-diagnostic check, and then re-operates from the step 0.

(a) When operating in each program block by the CJ instruction

(b) When using the subroutine and interrupt programs

NOTE
- There is no limit on the number of the FEND instructions in a sequence program, however, they cannot be used in the subroutine and interrupt programs.
- The FEND instruction cannot be used to terminate the main or sub sequence program.
Make sure to use the END instruction for the end of a whole program.

REFERENCE
The interrupt program stops the current process and processes an interrupt upon receiving an interrupt request while a normal program is being processed.
Ladder example
Create the following ladder with GX Works2 and write it to the CPU of the demonstration machine. Then check that the FEND instruction operates properly.

Operating Procedure
The following procedures are the same as the Operating Procedure in section 4.4.

(1) Creating a new project
(2) Creating a program
(3) Writing the project to the programmable controller
(4) Monitoring the ladder
Operation Practice

Verify the operation of the ladder, which was created with GX Works2 and written to the CPU of the demonstration machine, by monitoring the ladder on the screen.

(1) When X3 is off
(a) The operation is executed from 0 to FEND.
(b) Turning on or off X4 turns on or off Y70.
(c) Turning on or off X5 does not change Y72.

(2) When X3 is on
(a) The program jumps to the pointer P10 by the CJ instruction.
(b) Turning on or off X4 does not change Y70.
(c) Turning on or off X5 turns on or off Y72.

Related Exercise: Exercise 4
4.7.2 CJ (Conditional jump: instantaneous execution condition jump)

SCJ (S conditional jump: execution condition jump after one scan)

- The CJ instruction instantaneously executes a program jumping it to the specified address (pointer number) when the execution command is on. When the command is off, the program is not jumped.

- The SCJ instruction executes a program without jumping it for the scan when the execution command is turned on. From the next scan, the instruction executes the program jumping it to the specified address (pointer number). When the command is off, the program is not jumped.

- The SCJ instruction is used when some operations must be executed before jumping the program. For example, when the output needs to be on or reset in advance.

[Timing chart]

- Input condition (X0, X1)
- CJ executes every scan
- SCJ executes every scan
- One scan
NOTE

• The pointer numbers available for both CJ and SCJ instructions are P0 to P4095.
• Use the FEND instruction as shown below when a program using the CJ and SCJ instructions must be concluded in each program block. (Refer to section 4.7.1 for FEND.)

![Diagram](image)

- The status of ladders skipped by the CJ instruction remains unchanged.

(Before CJ execution)

(During CJ execution)

Because X0 is on, all instructions within this area are not executed. Hence Y72 remains on even after X2 is turned off.

• After the timer coil has turned on, jumping the timer of a coil that is on using the CJ, SCJ, or JMP instruction interrupts an accurate measurement.

<table>
<thead>
<tr>
<th></th>
<th>Internal device (system or user)</th>
<th>File register</th>
<th>MELSECNET/10 (H) Direct</th>
<th>Intelligent function module</th>
<th>Index register</th>
<th>Constant</th>
<th>Pointer</th>
<th>Level</th>
<th>Digit</th>
<th>Number of basic steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>CJ</td>
<td>p**</td>
<td>Bit</td>
<td>Word</td>
<td>R</td>
<td>Bit</td>
<td>Word</td>
<td>UnG</td>
<td>Z</td>
<td>K</td>
<td>H</td>
</tr>
<tr>
<td>SCJ</td>
<td>p**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Ladder example
Create the following ladder with GX Works2 and write it to the CPU of the demonstration machine. Then check the difference between the CJ and SCJ instructions.

Operating Procedure
The following procedures are the same as the Operating Procedure in section 4.4.

1. Creating a new project
2. Creating a program
3. Writing the project to the programmable controller
4. Monitoring the ladder
(1) When X0 and X1 are off, the CJ and SCJ instructions are not executed. Therefore, Y70 is on.

(2) When X0 is turned on, the CJ instruction is executed and the program jumps to P10. Therefore, Y70 remains on.

(3) Turning off X0 and on X1 executes the SCJ instruction and jumps the program to P10 from the second scan. Therefore, Y70 turns off.

(4) Y71 is turned on or off when the CJ and SCJ instructions are executed.
   - The following lists explain the difference between the CJ and SCJ instructions.

- The related exercise is Exercise 4.

**Operation Practice**

**Before CJ and SCJ execution**

**During CJ execution**

**During SCJ execution**
4.7.3 CALL(P) Call
        RET Return

Executing a subroutine program

- The above program is a basic style to execute the subroutine program using the CALL and RET instructions. Keep this structure, otherwise an error occurs and the programmable controller stops.

- A subroutine program consists of the ladders for executing the same data many times in one program.
  A subroutine program starts at a pointer P and ends with the RET instruction.

- The pointer P number is from 0 to 4095. (Same as the pointer numbers used for the CJ and SCJ instructions.)

- A subroutine program is executed as shown in the following diagrams.

![Diagram of subroutine program execution](image-url)
Nesting

• The CALL (P) instructions can be nested up to 16 levels.

The number of basic steps of CALL (P) is 2tn, and that of the RET instruction is one. ("n" is an argument passed to the subroutine.)
Ladder example
Create the following ladder with GX Works2 and write it to the CPU of the demonstration machine. Then check that the CALL and RET instructions operate properly.

```
<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P10</td>
<td>X4</td>
<td></td>
<td>FEND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y70</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y71</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Operating Procedure
The following procedures are the same as the Operating Procedure in section 4.4.

1. Creating a new project
2. Creating a program
3. Writing the project to the programmable controller
4. Monitoring the ladder
Operation Practice

Verify the operation of the ladder, which was created with GX Works2 and written to the CPU of the demonstration machine, by monitoring the ladder on the screen.

(1) When X2 is off
   (a) The operation is executed from 0 to FEND.
   (b) Turning on or off X3 turns on or off Y70.
   (c) Turning on or off X4 does not change Y71.

(2) When X2 is on
   (a) After the subroutine of P10 is executed, the operation from step 3 to FEND is executed.
   (b) Turning on or off X3 turns on or off Y70.
   (c) Turning on or off X4 turns on or off Y71.

Related Exercise --- Exercise 4
4.8 Exercise

4.8.1 Exercise 1

LD to NOP
When X0 turns on, Y70 is self-maintained, and Y74 and Y77 flicker alternately every 0.5 sec.
When X1 turns on, Y70 turns off and flickering of Y74 and Y77 also stops.

[Timing chart]

Create the following program with GX Works2 filling in the blanks [______]. Then, check the operation using the demonstration machine.
SET, RST
When X0 is turned on, Y70 starts to flicker at one-second intervals and stops the flickering for five seconds after flickering 10 times, then restarts flickering. The flickering of Y70 can be stopped by turning on X1.

Create the following program with GX Works2 filling in the blanks [____]. Then, check the operation using the demonstration machine.
(1) The following shows the timing chart of the program.

Counter of C0 1.  2.  ⋮  ⋮  ⋮  ⋮  10.  1.  2.0.

(2) The following shows the basic flickering ladder and its timing chart.

REFERENCE

• The flickering ladder can be created using the special relay that generates clock as shown below.
  SM413 (2-sec. clock)

In addition to the SM413 (2-sec. clock) on the left, the following can be used.
  SM409 (0.01-sec. clock)
  SM410 (0.1-sec. clock)
  SM411 (0.2-sec. clock)
  SM412 (1-sec. clock)
  SM414 (2n-sec. clock)
  SM415 (2n-msec. clock)

The clock starts from OFF when the programmable controller is reset or the power is turned on.
PLS, PLF
Y70 starts to switch between ON and OFF alternately when X0 is turned on, and turning off X0 triggers Y71 to operate in the same way as Y70 does.

[Timing chart]

Create the following program with GX Works2 filling in the blanks [____]. Then, check the operation using the demonstration machine.

1) ______________________
2) ______________________
CJ, CALL, RET, FEND

Y70 and Y71 flicker for 0.5sec. alternately when X7 is off, and when X7 is on, Y72 and Y73 flicker for 1.0sec. alternately. Turning on X0 resets the currently flickering Y70 to Y73.

Fill in the blanks [____]. Then, check the operation using the demonstration machine.
START

X7 ON?

Y

< CJ P0 >

N

Y72,Y73 Reset

Y70,Y71 Reset

Y70,Y71 0.5-sec. flickering

Y72,Y73 1-sec. flickering

X0 ON?

Y

N

Y70,Y73 Reset

Subroutine program

Y70,Y73

Reset

P10

< FEND >

END

1) ________________
2) ________________
3) ________________
4) ________________
5) ________________
6) ________________
# Answers for the exercises in Chapter 4

<table>
<thead>
<tr>
<th>Exercise No.</th>
<th>Answer</th>
</tr>
</thead>
</table>
| 1            | 1) Y70  
|              | 2) X1   
|              | 3) T1   
|              | 4) T0   
|              | 5) Y74  
| 2            | 1) SET M0  
|              | 2) C0   
|              | 3) Y70  
|              | 4) SET M1  
|              | 5) RST C0  
|              | 6) RST M0  
|              | 7) RST C0  
|              | 8) RST M1  
| 3            | 1) PLS  
|              | 2) PLF  
| 4            | 1) P0  
|              | 2) CALL  
|              | 3) FEND  
|              | 4) CALL  
|              | 5) FEND  
|              | 6) RET  

4 - 41
The programmable controller CPU converts all input signals into ON or OFF signals (logical 1 or 0, respectively) to store and process them. Therefore, the programmable controller executes the numeric operation using the numeric values stored with the logical 1 or 0 (binary numbers = BIN).

In daily life, a decimal number is regarded as the most commonly and the easiest system. Therefore, the decimal-to-binary conversion or the reverse conversion is required when values are written or read (monitored) to or from the programmable controller. The programming system and some instructions have the function for the decimal-to-binary conversion and the binary-to-decimal conversion.

This section explains how to express values (data) in decimal, binary, hexadecimal and binary-coded decimal notation (BCD), and how to convert them.

- A decimal number system consists of ten symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 which represent the order and size (amount).
  - After a digit reaches 9, an increment is reset to 0 and the next increment of the next digit (to the left) is incremented.
- The following shows how a decimal number (in this case 153) is represented.
  \[
  153 = 100 + 50 + 3 = 1 \times 10^2 + 5 \times 10^1 + 3 \times 10^0
  \]

\[
\begin{align*}
  n &: \text{Digit number (0, 1, 2...)} \\
  10 &: \text{Decimal}
\end{align*}
\]

- In the MELSEC-Q series programmable controller, symbol "K" is used for expressing values in decimal.
The binary number system consists of two symbols: 0 and 1 which represent the order and size (amount). After a digit reaches 1, an increment is reset to 0 and the next digit (to the left) is incremented. The two digits 0 and 1 are called bits.

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>111</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

The following example explains how to convert a binary number into a decimal number.

"10011101"

The diagram below shows the binary number with the powers of two.

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6 5 4 3 2 1 0</td>
<td>1 0 0 1 1 1 0 1</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

\[ 1 \times 128 + 0 \times 64 + 0 \times 32 + 1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 \]
\[ = 128 + 16 + 8 + 4 + 1 \]
\[ = 157 \]

The binary number can be calculated by adding each weight of bits whose codes are 1.

5 - 2
The hexadecimal number system consists of 16 symbols: 0 to 9 and A to F which represent the order and size (amount). After a digit reaches F, an increment is reset to 0 and the next digit (to the left) is incremented.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hexadecimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>111</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>1010</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>1011</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>1100</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>1101</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>1110</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>1111</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>10000</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>10001</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>10010</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>A</td>
<td>10100</td>
</tr>
<tr>
<td>20</td>
<td>9</td>
<td>10101</td>
</tr>
<tr>
<td>21</td>
<td>D</td>
<td>11010</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Four bits of a binary number equal to one digit of a hexadecimal number.

In the MELSEC-Q series programmable controller, symbol "H" is used for indicating a hexadecimal number.

The hexadecimal system is used to represent the following device numbers:
- Input and output (X, Y)
- Function input and output (FX, FY)
- Link relay (B)
- Link register (W)
- Link special relay (SB)
- Link special register (SW)
- Link direct device (J n\X, J n\Y, J n\B, J n\SB, J n\W, J n\SW)
**Binary Coded Decimal (BCD)**

- The binary-coded decimal is "a numerical system using a binary number to represent a decimal number".

  A decimal number 157, for example, is expressed as shown below.

  
<table>
<thead>
<tr>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

  ← Digit number

  (100) (10) (1) ← Power of digit

  0001 : 0101 : 0111  ← BCD

  8421 : 1001 : 1010  ← Binary bit weights

- In BCD, decimal numbers 0 to 9999 (the biggest 4-digit number) can be represented by 16 bits.

  The diagram below shows the bit weights of BCD.

  ![Bit weights of BCD diagram]

  - BCD is used for the following signals.
    1) Output signals of digital switches
    2) Signals of seven-element display (digital display)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1(0)</td>
<td>2(0)</td>
<td>3(0)</td>
<td>4(0)</td>
<td>5(0)</td>
<td>6(0)</td>
<td>7(0)</td>
<td>8(0)</td>
<td>9(0)</td>
</tr>
<tr>
<td>1</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
</tr>
<tr>
<td>2</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
</tr>
<tr>
<td>3</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
</tr>
<tr>
<td>4</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
</tr>
<tr>
<td>5</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
</tr>
<tr>
<td>6</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
</tr>
<tr>
<td>7</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
</tr>
<tr>
<td>8</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
</tr>
<tr>
<td>9</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
<td>O(0)</td>
</tr>
</tbody>
</table>

  BCD code digital switch
How to convert the decimal number into the binary number

In the example below, a decimal number 157 is converted into the binary number.
1)

\[ \begin{array}{c|c|c|c|c|c|c|c}
128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\
\hline
1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
\end{array} \]

2)

\[ \begin{array}{c|c|c|c|c|c|c|c}
128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\
\hline
1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
\end{array} \]

How to convert the decimal number into the hexadecimal number

In the example below, a decimal number 157 is converted into the hexadecimal number.
1)

\[ \begin{array}{c}
157 \div 16 = 9 \text{ R } 13 \\
\end{array} \]

9 \leftrightarrow \text{ D}
Numerical values used by MELSEC-Q series programmable controller

- Usually, 8 bits are called 1 byte, and 16 bits (2 bytes) are called 1 word.

- Registers of each word device in the MELSEC-Q series programmable controller consist of 16 bits.
  - Data register D
  - Current value of timer T
  - Current value of counter C
  - File register R
  - Link register W

- The following two ranges of numbers can be processed in 16 bits (1 word).
  1) 0 to 65535
  2) -32768 to +32767

- The range 2) is available for the MELSEC-Q programmable controller.
  The negative numbers adopt two's complement against the positive numbers (1 to +32767).

- In the two's complement, each binary bit is inverted, and 1 is added to the least significant bit.
  Example
  How to calculate the two's complement against 1
  -1 ...
  \[
  \begin{array}{c}
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  1 \\
  \end{array}
  \]
  Invert all bits.

  \[
  \begin{array}{c}
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  0 \\
  \end{array}
  \]
  Add 1 to the least significant bit.

  \[
  \begin{array}{c}
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  1 \\
  0 \\
  \end{array}
  \]
  The most significant bit of "1" means negative.

  The most significant bit corresponds to the sign of a signed binary number.
<table>
<thead>
<tr>
<th>BCD (binary coded decimal)</th>
<th>BIN (binary)</th>
<th>K (decimal)</th>
<th>H (hexadecimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000 00000000</td>
<td>00000000 00000000</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>00000000 00000001</td>
<td>00000000 00000001</td>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>00000000 00000010</td>
<td>00000000 00000010</td>
<td>2</td>
<td>0002</td>
</tr>
<tr>
<td>00000000 00000011</td>
<td>00000000 00000011</td>
<td>3</td>
<td>0003</td>
</tr>
<tr>
<td>00000000 00000100</td>
<td>00000000 00000100</td>
<td>4</td>
<td>0004</td>
</tr>
<tr>
<td>00000000 00000101</td>
<td>00000000 00000101</td>
<td>5</td>
<td>0005</td>
</tr>
<tr>
<td>00000000 00000110</td>
<td>00000000 00000110</td>
<td>6</td>
<td>0006</td>
</tr>
<tr>
<td>00000000 00000111</td>
<td>00000000 00000111</td>
<td>7</td>
<td>0007</td>
</tr>
<tr>
<td>00000000 00001000</td>
<td>00000000 00001000</td>
<td>8</td>
<td>0008</td>
</tr>
<tr>
<td>00000000 00001001</td>
<td>00000000 00001001</td>
<td>9</td>
<td>0009</td>
</tr>
<tr>
<td>00000000 00010000</td>
<td>00000000 00010000</td>
<td>10</td>
<td>00A</td>
</tr>
<tr>
<td>00000000 00010001</td>
<td>00000000 00010001</td>
<td>11</td>
<td>00B</td>
</tr>
<tr>
<td>00000000 00010010</td>
<td>00000000 00010010</td>
<td>12</td>
<td>00C</td>
</tr>
<tr>
<td>00000000 00010011</td>
<td>00000000 00010011</td>
<td>13</td>
<td>00D</td>
</tr>
<tr>
<td>00000000 00010100</td>
<td>00000000 00010100</td>
<td>14</td>
<td>00E</td>
</tr>
<tr>
<td>00000000 00010101</td>
<td>00000000 00010101</td>
<td>15</td>
<td>00F</td>
</tr>
<tr>
<td>00000000 00010110</td>
<td>00000000 00010100</td>
<td>16</td>
<td>010</td>
</tr>
<tr>
<td>00000000 00010111</td>
<td>00000000 00010101</td>
<td>17</td>
<td>011</td>
</tr>
<tr>
<td>00000000 00011000</td>
<td>00000000 00011000</td>
<td>18</td>
<td>012</td>
</tr>
<tr>
<td>00000000 00011001</td>
<td>00000000 00011001</td>
<td>19</td>
<td>013</td>
</tr>
<tr>
<td>00000000 00100000</td>
<td>00000000 00100000</td>
<td>20</td>
<td>014</td>
</tr>
<tr>
<td>00000000 00100001</td>
<td>00000000 00100001</td>
<td>21</td>
<td>015</td>
</tr>
<tr>
<td>00000000 00100010</td>
<td>00000000 00100010</td>
<td>22</td>
<td>016</td>
</tr>
<tr>
<td>00000000 00100011</td>
<td>00000000 00100011</td>
<td>23</td>
<td>017</td>
</tr>
<tr>
<td>00000010 00000000</td>
<td>00000000 00110000</td>
<td>100</td>
<td>064</td>
</tr>
<tr>
<td>00000010 00100111</td>
<td>00000000 01111111</td>
<td>127</td>
<td>07F</td>
</tr>
<tr>
<td>00000100 01010101</td>
<td>00000000 11111111</td>
<td>255</td>
<td>0FF</td>
</tr>
<tr>
<td>00100000 00000000</td>
<td>00000011 11101000</td>
<td>1000</td>
<td>03E8</td>
</tr>
<tr>
<td>00100000 01000111</td>
<td>00000111 11111111</td>
<td>2047</td>
<td>07FF</td>
</tr>
<tr>
<td>01000000 10010101</td>
<td>00011111 11111111</td>
<td>4095</td>
<td>0FF</td>
</tr>
<tr>
<td>00100111 00010000</td>
<td>00011111 11111111</td>
<td>1000</td>
<td>2710</td>
</tr>
<tr>
<td>01111111 11111111</td>
<td>01111111 11111111</td>
<td>32767</td>
<td>7FFF</td>
</tr>
<tr>
<td>11111111 11111111</td>
<td>11111111 11111111</td>
<td>-1</td>
<td>FFFF</td>
</tr>
<tr>
<td>11111111 11111111</td>
<td>11111111 11111111</td>
<td>-2</td>
<td>FFFE</td>
</tr>
<tr>
<td>10000000 00000000</td>
<td>10000000 00000000</td>
<td>-32768</td>
<td>8000</td>
</tr>
</tbody>
</table>
System configuration and I/O number of demonstration machine

- Power supply module
- CPU module
- Input module
- Output module
- Base unit Q38DB

Peripheral device

USB cable

I/O panel

CPU module

Output module

Vacant slot

QX

42

(64 points)

QY

42P

(64 points)

Q64

AD

(16 points)

Q62

DAN

(16 points)

X0 to X3F

Y40 to Y7F

Peripheral device

Y77 Y76 Y75 Y74 Y73 Y72 Y71 Y70

Y7F Y7E Y7D Y7C Y7B Y7A Y79 Y78

Y77 Y76 Y75 Y74 Y73 Y72 Y71 Y70

Y7F Y7E Y7D Y7C Y7B Y7A Y79 Y78

X7 X6 X5 X4 X3 X2 X1 X0

ON OFF

ON OFF

ON OFF

ON OFF

X8 X7 X6 X5 X4 X3 X2 X1 X0

ON OFF

ON OFF

ON OFF

X8 X7 X6 X5 X4 X3 X2 X1 X0

ON OFF

ON OFF

ON OFF

AD INPUT

D/A OUTPUT

Y6F Y6E Y6D Y6C Y6B Y6A Y69 Y68

Y6F Y6E Y6D Y6C Y6B Y6A Y69 Y68

26

3709

1402

X3F X3D X3C X3B X3A X39 X38

ON OFF

ON OFF

1 9 4 2 4 1 3 6

MELSEC-Q

5 - 8
5.2 Transfer Instruction

5.2.1 **MOV (P)**  16-bit data transfer

- When the input condition turns on, the current value of the timer T0 is transferred to the data register D0.
  - Source
  - Destination
- The current value of T0 is stored in the register in binary (BIN code). And the value is transferred to the data register D0 in binary (The code is not converted at the transfer.)

```
T0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 45
```

```
D0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1
```

- When the input condition turns on, the decimal number 157 is transferred to the data register D2. And the value is stored in the register in binary. The decimal number (K) is converted into binary automatically, then transferred.

```
D2 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1
```

```
K157
```

```
D2 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1
```

---

<table>
<thead>
<tr>
<th>Project name</th>
<th>QB-11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>
When the input condition turns on, the hexadecimal number 4A9D is transferred to the data register D3.

### Difference between MOV and MOVP

The P of MOVP stands for a pulse.

- **Input condition**
  - MOV: Data is transferred every scan while the input condition is on.
  - MOVP: Data is transferred only one scan after the input condition turns on. (For only once)

- Use the **MOV** instruction when reading the changing data for all the time.
- Use the **MOVP** instruction to transfer data instantaneously such as setting data or reading data at an error.
- Both of the following ladder programs function similarly.

<table>
<thead>
<tr>
<th>MOV</th>
<th>MOVP</th>
</tr>
</thead>
<tbody>
<tr>
<td>X4</td>
<td>K157</td>
</tr>
<tr>
<td>X4</td>
<td>PLS</td>
</tr>
</tbody>
</table>

### Applicable device

<table>
<thead>
<tr>
<th>Internal device (system or user)</th>
<th>File register</th>
<th>MELSECNET/10 (H) Direct J n</th>
<th>Intelligent function module Un/G</th>
<th>Index register</th>
<th>Constant</th>
<th>Pointer</th>
<th>Level</th>
<th>Digit</th>
<th>Number of basic steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Word</td>
<td>R Bit Word</td>
<td>Z K H P I N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td>S D</td>
<td>O O O O O O O O O O O</td>
<td></td>
<td></td>
<td></td>
<td>5 6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*: The number of steps varies depending on the device to be used.
- Monitor the contents of the data registers D0 to D3.
  - After writing the data to the programmable controller, click [Online] → [Monitor] → [Device/Buffer Memory Batch]. The Device/Buffer Memory Batch Monitor dialog box is displayed.
  - Enter "D0" in the Device Name column of the Device/Buffer Memory Batch Monitor dialog box and press the Enter key.

The CPU is running.
The inputs X2, X3, X4, X5, and X7 are on.
Indicates that a decimal number 157 (K157) is stored.

Current values of a timer and counter are monitored. (They are changing.)

This is a decimal number equivalent to a hexadecimal 4A9D.

Indicates the word devices in the on/off of the bit units.  

<table>
<thead>
<tr>
<th>D3</th>
<th>4</th>
<th>A</th>
<th>9</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Hexadecimal numbers: H4A9D

Binary bit weights:
- 16
- 8
- 4
- 2
- 1

Sign bit:

19101
• Click the **Display Format** button.
• Change the display of the numerical value in the monitor to the hexadecimal notation.
• Select “HEX” for the device batch monitoring.

[Device/Buffer Memory Batch Monitor screen]

<table>
<thead>
<tr>
<th>Device</th>
<th>D0</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>02</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>03</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>04</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>05</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

• Change the display of the numerical value in the monitor to the binary notation.
Select “Word Multi-point” in Monitor Format for the device batch monitoring.

[Device/Buffer Memory Batch Monitor screen]
Ladder example
Create the following ladder with GX Works2 and write it to the CPU of the demonstration machine. Then check that the MOV instruction works properly.

Operating Procedure
The following procedures are the same as the **Operating Procedure** in section 4.4.

1. Creating a new project
2. Creating a program
3. Writing the project to the programmable controller
4. Monitoring the ladder

* How to modify the transfer instruction
To modify the transfer instruction, follow the procedures below.
Example: Change the transfer data K200 of [MOV K200 D0] to K100
1) Double-click the instruction to be modified.

```
MOV K200 D0
```

2) The Enter Symbol window is displayed.

![Enter Symbol](image)

3) Write "1" over "2" of "MOV K200 D0".
4) Click the **OK** button on the Enter Symbol window.

```
MOV K200 D0
```

All data in **{}** can be modified with the above method.
When "Insert" is displayed, press the **Insert** key to change it to "Ovrwrte" before modifying.

```
Ovrwrte  CAP  NUM
```

5) After finishing modifications, click [Compile] → [Build].
Operation Practice

Check that "200" is displayed under both D0 and D1 on the monitor screen when X0 on the control panel of the demonstration machine is turned on.

When X0 turns on, the current values of D0 and D1 become 200.

Related Exercise ---- Exercise 5
5.2.2 **BIN (P) BCD → BIN data conversion instruction**

Operations to read and write data after step 35

```
  0   X7  TO  K50
  30  X0  TO  K1500
  34  X0  BINP  K4X20  D5
        MOV  K4X20  D6
```

- When the input condition is turned on, the data in the device specified in (5) is recognized as a BCD code, converted into binary (BIN code), and transferred to the device specified in (6).

<table>
<thead>
<tr>
<th>Thousand digits</th>
<th>Hundred digits</th>
<th>Ten digits</th>
<th>Unit digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(5) side BCD 9999

- The ordinary digital switches generate BCD codes. Therefore, the BIN instruction is required for writing data from the digital switches to the programmable controller.

- When the BCD code is input without converted.

```
  4660 128 16 4

0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0
```

D6

- When the BCD code is input after converted into the binary code.

```
  1234 16 8 4 2 1

0 0 0 0 0 1 0 1 0 1 1 0 1 0 0 0 1 0
```

D5

Check the difference from the BIN instruction.
- Word devices D (data register), T (timer current value), and C (counter current value) consist of 16 bits (1 word), and data is basically transferred among the units of one device.
- Collecting 16 bit devices (such as X, Y, and M) means processing the word device. The device numbers allocated to the bit devices must be in consecutive order.
- In the bit device, data are processed in units of four.

Specify these devices to read two-digit data "12":

K2X2B  (X2F to X28)
K1X20  (X23 to X20)
K4X20  (X2F to X20)

As long as four bit devices are in consecutive order, any bit device can be specified as the first.

* Other bit devices can be processed in the same way as described above.

* A sample program using a digital switch to import data is provided in page App.- 46.
5.2.3 **BCD (P) BIN → BCD data conversion instruction**

- When the input condition is turned on, the data in the device specified in \( \textcircled{5} \) is recognized as a binary (BIN code), converted into a binary coded decimal (BCD code), and transferred to the device specified in \( \textcircled{1} \).

**BIN side** (9999)

- Must be set to "0".

**BCD side** (9999)

- The ordinary digital displays display numbers in the BCD code. Therefore, the BCD instruction is required for displaying data of the programmable controller (current values of timers and counters, data register values of operation results).
Displayable Range with BCD Instruction

- The displayable range of data with the BCD instruction (to be converted from BIN into BCD) is between 0 and 9999. Any data which is outside the range causes an error.
  (Error code 4100: OPERATION ERROR)
- To display a timer current value more than 9,999, use the DBCD instruction. The instruction can handle 8-digit values (up to 99,999,999).

---

Programmable controller Output module

Output power supply

Applicable device

<table>
<thead>
<tr>
<th>Internal device (system or user)</th>
<th>File register</th>
<th>MELSECNET/10 (H) Direct</th>
<th>Intelligent function module</th>
<th>Index register</th>
<th>Constant</th>
<th>Pointer</th>
<th>Level</th>
<th>Number of basic steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>⑤</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>K1 to K4</td>
<td>3</td>
</tr>
<tr>
<td>⑨</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Ladder example
Create the following ladder with GX Works2 and write it to the CPU of the demonstration machine. Then check that the BCD instruction works properly.

Operating Procedure
The following procedures are the same as the Operating Procedure in section 4.4.

1. Creating a new project
2. Creating a program
3. Writing the project to the programmable controller
4. Monitoring the ladder

Operation Practice
Check that turning on X0 on the control panel for several times displays the value of C0 on the BCD digital displays of Y40 to Y47. Turning on X1 resets C0.

Related Exercise ---- Exercise 6
### Example of specifying digit for bit devices and transferring data

#### Program example

<table>
<thead>
<tr>
<th>When the destination data (D) is a word device</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
</tbody>
</table>

- **Source**: Source device
- **Destination**: Destination device

#### When the source data (S) is a word device

| ![Diagram](image3) | ![Diagram](image4) |

#### When the source data (S) is a constant

| ![Diagram](image5) | ![Diagram](image6) |

#### When the source data (S) is a bit device

| ![Diagram](image7) | ![Diagram](image8) |
5.2.5 **FMOV (P)** FMOV (Batch transfer of the same data)

**BMOV (P)** BMOV (Batch transfer of the block data)

### Operation Explanation

**FMOV**

- When the input condition is turned on, the FMOV instruction transfers the data in the device specified in \( S \) to the devices starting from the device specified in \( D \) (the number of target devices is specified by \( n \)).

**Example** The FMOV instruction executes the following operation when X3 is turned on.

- The FMOV instruction is useful for clearing many data sets in batch.

**Example**

The FMOV instruction can substitute the RST instructions as shown above.
When the input condition is turned on, the BMOV instruction transfers the data in the devices starting from the device specified in \( S \) to the devices starting from the device specified in \( D \) in batch (the numbers of source devices and target devices are specified by \( n \)).

**Example**  The BMOV instruction executes the following operation when X5 is turned on.

\[
\begin{align*}
\text{S365} & \quad \text{D0} \quad \text{S365} \\
\text{D7} & \quad \text{D32} \\
\text{D8} & \quad \text{D32} \\
\text{D15} & \quad \text{D32}
\end{align*}
\]

The BMOV instruction is useful for the following:

- Saving important data (such as automatic operation data and measured data) into the latch area. This can prevent a data loss caused by a power failure.

---

**Applicable device**

<table>
<thead>
<tr>
<th>Internal device (system or user)</th>
<th>File register</th>
<th>MELSECNET/10 (H) DirectJ nL</th>
<th>Intelligent function module UnG</th>
<th>Index register</th>
<th>Constant</th>
<th>Pointer</th>
<th>Level</th>
<th>Digit</th>
<th>Number of basic steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>Word</td>
<td>R</td>
<td>Bit</td>
<td>Word</td>
<td>Z</td>
<td>K</td>
<td>H</td>
<td>P</td>
<td>I</td>
</tr>
<tr>
<td>FMOV  ( S )  ( D )  ( O )</td>
<td>( O )</td>
<td>( O )</td>
<td>( O )</td>
<td>( O )</td>
<td>( O )</td>
<td>( O )</td>
<td>( O )</td>
<td>( O )</td>
<td>( O )</td>
</tr>
<tr>
<td>BMOV  ( S )  ( D )  ( D )</td>
<td>( O )</td>
<td>( O )</td>
<td>( O )</td>
<td>( O )</td>
<td>( O )</td>
<td>( O )</td>
<td>( O )</td>
<td>( O )</td>
<td>( O )</td>
</tr>
</tbody>
</table>

(Note) Not available in the BMOV instruction.
Operation Practice

- Write the program on the previous page to the CPU, then run the CPU.
- Follow the procedures below to execute the device batch monitoring. The contents of D0 to D47 can be monitored.

Write the program to the programmable controller → Click [Online] → [Monitor] → [Device/Buffer Memory Batch].

Enter “D0” in the Device/Buffer Memory Batch Monitor dialog box and press the Enter key.

- Click the Display Format button and select "Word Multi-point" for Monitor Format.

→ Click the OK button.

[Monitor screen]

<table>
<thead>
<tr>
<th>Device</th>
<th>+9</th>
<th>+1</th>
<th>+2</th>
<th>+3</th>
<th>+4</th>
<th>+5</th>
<th>+6</th>
<th>+7</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0</td>
<td>365</td>
<td>365</td>
<td>365</td>
<td>365</td>
<td>365</td>
<td>365</td>
<td>365</td>
<td>365</td>
</tr>
<tr>
<td>D1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

1) Turn on X3. The numeric data 365 is sent to eight registers of D0 to D7 in batch.

2) Turn on X4. The numeric data 7000 is sent to 16 registers of D8 to D23 in batch.

3) Turn on X5. The contents of the 16 registers of D0 to D15 are sent to the 16 registers of D32 to D47 in batch.

4) Turn on X6. "0" is sent to the all 48 registers of D0 to D47 in batch. This means that all the 48 registers are cleared.
If $D$ is a bit device, the operation becomes as follows:

**FMOV instruction**

As $D$ specifies a two-digit number, these data are ignored.

- Among the device of Y40 to Y5F, the devices specified as "1" are output first.
- In the program shown below, turning on the input condition 1) turns on all the outputs Y40 to Y5F and turning on the input condition 2) turns them off.

In units of four bits, to turn off:
- 16 bit devices or less $\Rightarrow$ MOV instruction
- 32 bit devices or less $\Rightarrow$ DMOV instruction
- More than 32 bit devices $\Rightarrow$ FMOV instruction

**BMOV instruction**

As $D$ specifies a two-digit number, these data are ignored.

- In the example above, the devices D0 to D3 store the product code (16 bits). The BMOV instruction is useful for displaying and monitoring the last two digits representing their types.
Ladder example
Create the following ladder with GX Works2 and write it to the CPU of the demonstration machine. Then check that the FMOV instruction works properly.

```
X0  FMOV K200  D0  K5  3
X1  FMOV K0   D0  K5  3
```

Operating Procedure
The following procedures are the same as the Operating Procedure in section 4.4.

1. Creating a new project
2. Creating a program
3. Writing the project to the programmable controller
4. Monitoring the ladder

Operation Practice
Check that the contents of the devices of D0 to D4 become 200 on the batch monitor screen by turning on X0 on the control panel of the demonstration machine. Turning on X1 clears the data in the devices.

```
<table>
<thead>
<tr>
<th>Device</th>
<th>+0</th>
<th>+1</th>
<th>+2</th>
<th>+3</th>
<th>+4</th>
<th>+5</th>
<th>+6</th>
<th>+7</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D40</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D51</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Change the setting of the device batch monitor as shown below to display the numbers in decimal, hexadecimal, or binary notation.

- **Value: DEC** displays numbers in decimal.
- **Value: HEX** displays numbers in hexadecimal.
- **Monitor Format: Bit Multi-point** displays numbers in binary.

Related Exercise ---- Exercise 7
5.3 Comparison Operation Instruction

- The comparison operation instruction compares the data of source 1 (S₁) and source 2 (S₂), and brings the devices into conduction when the conditions are met.
- The instruction can be regarded as one normally open contact (−−−−) since it is conducted only when the conditions are met.

- = $S₁$ $S₂$ …… Becomes conducted when source 1 and source 2 match.
- < $S₁$ $S₂$ …… Becomes conducted when source 1 is smaller than source 2.
- > $S₁$ $S₂$ …… Becomes conducted when source 1 is larger than source 2.
- <= $S₁$ $S₂$ …… Becomes conducted when source 1 and source 2 match or when source 1 is smaller than source 2.
- >= $S₁$ $S₂$ …… Becomes conducted when source 1 and source 2 match or when source 1 is larger than source 2.
- <> $S₁$ $S₂$ …… Becomes conducted when source 1 and source 2 do not match.
Operation Practice

- Write the program to the CPU.
- Turn on X3 and X4.
- C10 starts to count (one count every two sec.) The current counter value is displayed on the digital display (Y40 to Y4F).
- Make sure that the devices Y70 to Y76 turn on as follows.

<table>
<thead>
<tr>
<th>Y70</th>
<th>Y71</th>
<th>Y72</th>
<th>Y73</th>
<th>Y74</th>
<th>Y75</th>
<th>Y76</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

Output

The range where Y70 to Y76 turn on

0 1 2 3 4 5

Count (the current value of the counter C10)

Differences between $K50 \ C10 > = 49$

$K50 \ C10 > = 50$

- The counter is designed to be reset every 200 sec.
- In this way, the comparison instruction does not only compare one data but also specifies the range. This function is commonly used for the program to judge the acceptances of products.

<table>
<thead>
<tr>
<th>Internal device (system or user)</th>
<th>File register</th>
<th>MELSECNET /10 (H) Direct J n\</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>Word</td>
<td>R Bit Word</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index register</th>
<th>Constant</th>
<th>Pointer</th>
<th>Level</th>
<th>Digit</th>
<th>Number of basic steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z K H P I N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Comparison instruction

$S1$ $S2$

$S1$ $S2$
Ladder example
Read the following ladder and write it to the CPU of the demonstration machine. Then check that the > and < instructions work properly.

0sec. ≤ T0 < 3sec. → Y70: ON, 2.7sec. < T0 < 3.3sec. → Y71: ON, 3sec. < T0 ≤ 6sec. → Y72: ON

Operating Procedure
(1) Reading data
Read the project data.
- Click on the toolbar.
- The Open dialog box is displayed. Specify the save destination.
- Double-click the displayed workspace "SCHOOL".

![Image of the Open dialog box]

- Click "QEX10" and click the Open button.

![Image of the workspace selection]

The following procedures are the same as the Operating Procedure in section 4.4.

(2) Writing the project to the programmable controller
(3) Monitoring the ladder
Operation Practice

- Turn on X0 and check that the program works properly.
5.4 Arithmetic Operation Instruction

5.4.1 **BIN 16-bit data addition**

- **+P**

**BIN 16-bit data subtraction**

- **-P**

Every time the input condition is turned on, the content of the device specified in \( D \) is added to the content of the device specified in \( S \) and the result is stored in the device specified in \( D \).

\[
\begin{array}{ccc}
D_0 & + & S_5 \\
\text{(Input condition)} & & \\
0 \text{ (example)} & + & 5 \\
5 \text{ (example)} & + & 5 \\
10 \text{ (example)} & + & 5 \\
\end{array}
\]

The content of \( D_0 \) is changed.

When the input condition is turned on, the content of the device specified in \( S_3 \) is added to the content of the device specified in \( S_2 \) and the result is stored in the device specified in \( D \).

\[
\begin{array}{ccc}
D_0 & + & S_3 \\
\text{(Input condition)} & & \\
15 \text{ (example)} & + & 100 \\
\end{array}
\]

The content of \( D_0 \) is not changed.

**CAUTION**

- **+P** or **-P** must be used for the addition or subtraction instructions.
- When **+** or **-** is used, an addition or subtraction operation is executed every scanning. To use **+** or **-**, operands must be converted into pulse in advance.

**REFERENCE**

- The following two instructions work on the same principle in the addition or subtraction operation.
Every time the input condition is turned on, the content of the device specified in \( S \) is subtracted from the device specified in \( D \) and the result is stored in the device specified in \( D \).

\[
\begin{array}{c|c|c}
\text{Input condition} & \text{Action} & \text{Result} \\
\hline
\text{First ON} & 1000 \text{ (example)} - 10 & 990 \\
\text{Second ON} & 990 - 10 & 980 \\
\text{Third ON} & 980 - 10 & 970 \\
\end{array}
\]

The content of \( D \) is changed.

When the input condition is turned on, the content of the device specified in \( S2 \) is subtracted from the content of the device specified in \( S1 \) and the result is stored in the device specified in \( D \).

\[
\begin{array}{c|c|c}
\text{Input condition} & \text{Action} & \text{Result} \\
\hline
\text{ON} & 970 - 50 & 920 \\
\end{array}
\]

The content of \( D \) is not changed.

The number of basic steps is four for \( S1 \) to \( S2 \).
Ladder example
Create the following ladder with GX Works2 and write it to the CPU of the demonstration machine. Then check that the addition and subtraction instructions operate properly.

**Operating Procedure**
The following procedures are the same as the **Operating Procedure** in section 4.4.

1. Creating a new project
2. Creating a program
3. Writing the project to the programmable controller
4. Monitoring the ladder
Operation Practice

(1) When X0 is turned on, the data in X30 to 3F and X20 to 2F are added, and the result is output to Y40 to Y53.

(2) When X1 is turned on, the data in X30 to 3F is subtracted from the data in X20 to 2F, and the result is output to Y40 to Y53. When the result is a negative value, Y70 is turned on and Y40 to Y53 are cleared to 0.

Related Exercise  Exercise 9
5.4.2 *(P) BIN 16-bit data multiplication

/ (P) BIN 16-bit data division

* When the input condition is turned on, the content of the device specified in $S_1$ is multiplied by the content of the device specified in $S_2$ and the result is stored in the device specified in $D$.

\[
S_1 \times S_2 = D
\]

To store the result of 16-bit data $\times$ 16-bit data, 16 bits (1 word) is not enough. Therefore, $D10$ which is specified in the program and the next number $D11$ work as the holder of the result.

This device is regarded as a 32-bit register to hold the result. Left-most bit of $D10$ (B15) is not a bit to determine positive and negative. It is regarded as a part of the data.

The instructions must be regarded as 32 bits for programming with the calculation result of the *(P)* instruction. (such as the DMOV instruction and the DBCD instruction)
When the input condition is turned on, the content of the device specified in $S_1$ is divided by the content of the device specified in $S_2$ and the result is stored in the device specified in $D$.

\[
\begin{align*}
S_1 & \quad D_0 \\
2000 & \div 600 \\
D_20 & = 3 \\
D_21 & = 200
\end{align*}
\]

The quotient is stored to $D_{20}$, which is specified in the program. and The remainder is stored to $D_{21}$, which is the next device number.

Values after the decimal point of the operation result are ignored.

- When a bit device is specified in $D$, the quotient is stored, but the remainder is not stored.
- The following shows examples for processing negative values.

**Example**

- $-5 \div (-3) = 1$, remainder: -2
- $5 \div (-3) = -1$, remainder: 2

- The following shows examples for dividing a number by 0, or dividing 0 by a number.

**Example**

- $0 \div 0$
- $1 \div 0$
- $0 \div 1$, Both quotient and remainder are 0.

Error "OPERATION ERROR"

Operation Practice

- Write the program to the CPU and run it.
- Turn on X0 and store "2000" (BIN value) in $D_0$.
- Turn on X2. The following operation is executed.

\[
\begin{align*}
S_1 & \quad D_0 \\
2000 & \times 600 \\
D_11 & = 3 \\
D_10 & = 200
\end{align*}
\]

- Turn on X3.

The number of basic steps for the multiplication instruction is three or four, and that for division instruction is four.

*: The multiplication instruction varies depending on the device to be used.
How to monitor 32-bit integral number data

When the operation result of the multiplication instruction is outside the range from 0 to 32,767, the result cannot be displayed properly even though the number is regarded as 16-bit integral number and the contents of the lower register are monitored in ladder. To monitor those numbers properly, follow the procedures below.

- **Click the Display Format button on the Device/Buffer Memory Batch Monitor dialog box and select "32bit Integer" of "Display".**
- Click the OK button.

The data is monitored properly.
Ladder example
Create the following ladder with GX Works2 and write it to the CPU of the demonstration machine. Then check that the multiplication and division instructions operate properly.

```
0 X0
  BINP K4X30 D0
  BINP K4X20 D1
  D0  D1  D10
  /P   
  D10  D10
  D10  D10

13 X1
  BINP K4X30 D0
  BINP K4X20 D1
  D0  D1  D20
  /P   
  D20  D20
  D20  D20
```

Operating Procedure
The following procedures are the same as the Operating Procedure in section 4.4.

1. Creating a new project
2. Creating a program
3. Writing the project to the programmable controller
4. Monitoring the ladder
(1) When X0 is turned on, the data in X20 to X2F is multiplied by the data in X30 to 3F, and the result is output to Y40 to 5F.

(2) When X1 is turned on, the data in X30 to X3F is divided by the data in X20 to 2F. The quotient is output to Y50 to 5F, and the remainder is output to Y40 to 4F.

Related Exercise: Exercise 10, Exercise 11
5.4.3 32-bit data instructions and their necessity

- The minimum unit in the data memory of the Q-series programmable controller is 1 word which consists of 16 bits. Therefore, in general, data is processed in 1-word basis at the transfer, comparison, and arithmetic operation.

- The Q-series programmable controller can process data in 2-word (32-bit) basis. In that case, "D" is added at the head of each instruction to indicate that the instruction is regarded as 2-word. The following shows the examples.

<table>
<thead>
<tr>
<th>Data Instruction</th>
<th>1 word 16 bits</th>
<th>2 words 32 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer</td>
<td>MOV(P)</td>
<td>DMOV(P)</td>
</tr>
<tr>
<td></td>
<td>BIN(P)</td>
<td>DBIN(P)</td>
</tr>
<tr>
<td></td>
<td>BCD(P)</td>
<td>DBCD(P)</td>
</tr>
<tr>
<td>Comparison</td>
<td>&lt;, &gt;, &lt;=</td>
<td>D&lt;, D&gt;, D&lt;=</td>
</tr>
<tr>
<td></td>
<td>&gt;=, =, &lt;&gt;</td>
<td>D&gt;=, D=, D&lt;&gt;</td>
</tr>
<tr>
<td>Arithmetic</td>
<td>+ (P)</td>
<td>D + (P)</td>
</tr>
<tr>
<td>operations</td>
<td>- (P)</td>
<td>D - (P)</td>
</tr>
<tr>
<td></td>
<td>* (P)</td>
<td>D * (P)</td>
</tr>
<tr>
<td></td>
<td>/ (P)</td>
<td>D / (P)</td>
</tr>
</tbody>
</table>

Available range for values

-32,768 to 32,767
Values in parentheses are for BIN(P), BCD(P) instructions.

Available range for digits

K1 to K4

-2,147,483,648 to 2,147,483,647
Values in parentheses are for DBIN(P), DBCD(P) instructions.

- The bit weights of the 32-bit configuration are as follows:

```
B31 ................. B16
B15 .................. B0
```

As the case of 16-bit data processing, the programmable controller processes a 32-bit negative value in two’s complement. Therefore, the most significant bit B31 (B15 for 16-bit data), is a sign bit.

```
B31 ................. B0
```

Available range for numbers

-2147483648 to 2147483647

When the bit is 0, the number is interpreted as a positive number.
When the bit is 1, the number is interpreted as a negative number.
Whether the data is processed in 2-word (32-bit) basis or not depends on the size of the data.

In the following cases, 2-word instructions must be used.

1) When the data size exceeds the range (-32768 to 32767) in which data can be processed as 1-word.

![Diagram](image1)

2) When the result of the 16-bit multiplication instruction (1-word instruction) is transferred.

![Diagram](image2)

3) When the result of 32-bit division instruction is used.

![Diagram](image3)
5.4.4 Calculation examples for multiplication and division including decimal points (when the multiplication or division is used)

**Example 1** Calculation example to determine a circumference

\[
\text{Digitalswitch value} \times 3.14 \rightarrow \text{Integral part and Decimal part}
\]

(K4X30) (Circular constant) (K8Y50) (K2Y48)

- Programming method
  Handle the circular constant as 314 (3.14 × 100), and divide the result by 100 afterward.

**Example 2** Calculation example to handle values after decimal point (division example)

\[
\text{Digitalswitch value} \div 0.006 \rightarrow \text{Quotient and Remainder}
\]

(K4X30) (K8Y50) (K4Y40)

- Programming method
  To handle 0.006 as an integer 6, multiply both the dividend and divisor by 1000.

**REMARK**

QCPU has instructions which can process actual number (floating decimal point) operation data for highly accurate operations. As long as the instructions are used, careful attentions for the place of the decimal point as shown above are unnecessary.
5.5 Index Register and File Register

5.5.1 How to use index register Z

- The index register (Zn) is used to indirectly specify the device number. The result of an addition of data in the index register and the directly specified device number can be specified as the device number.

**Example**

\[
\text{D0Z0} \rightarrow \text{Can be interpreted as D (0+Z0)}
\]

Device number

For example, when Z0 is 0, the device number becomes D0.
when Z0 is 50, the device number becomes D50.

- Z0 to Z19 can be used as the index register.
- The index register (Zn) is a word device which consists of 16 bits. Therefore, the available data size range is -32768 to +32767.
- The following devices can be used for the indexing.
  - Bit device: X, Y, M, S, B, J nX, J nY, J nB, J nSB
  - Word device: T (Note), C (Note), D, R, W, J nW, J nSW, J nG
  - Constant: K, H
  - Pointer: P

(Note) Only the current value can be used for timer and counter.

The following restrictions are provided for using the index register for contact or coil.

<table>
<thead>
<tr>
<th>Device</th>
<th>Description</th>
<th>Application example</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>Only Z0 and Z1 are available for a contact and coil of a timer.</td>
<td>T0Z0 [\text{K100 T1Z1}]</td>
</tr>
<tr>
<td>C</td>
<td>Only Z0 and Z1 are available for a contact and coil of a counter.</td>
<td>C0Z1 [\text{K100 C1Z0}]</td>
</tr>
</tbody>
</table>

**REMARK**

When the index register is used with 32-bit data instructions

Zn and Zn+1 are targets to be processed.
The lower 16 bits correspond to the specified index register number (Zn), and the higher 16 bits correspond to the specified index register number + 1.

32-bit indexing
(Only for Universal model QCPU (except for Q00UJ CPU))

A method for specifying index registers for 32-bit indexing can be selected from following two methods.
- Specifying the index range used for 32-bit indexing
- Specifying the 32-bit indexing using “ZZ” specification

Refer to appendix 8 for the detail of indexing.
Application Example

- Write the data to the data register with number which is specified with the digital switch.

<table>
<thead>
<tr>
<th>Project name</th>
<th>Index register</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td></td>
</tr>
</tbody>
</table>

- Check the operation of the ladder executing the device batch monitoring. The operation procedure is the same as the one in section 5.2.1. Set any two-digit number in the digital switch column (X27 to X20) and turn on X0.

```
<table>
<thead>
<tr>
<th>Device</th>
<th>+0</th>
<th>+1</th>
<th>+2</th>
<th>+3</th>
<th>+4</th>
<th>+5</th>
<th>+6</th>
<th>+7</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E32</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E40</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E50</td>
<td>0</td>
<td>0</td>
<td>2030</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E64</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

The current value of T2 is transferred to D50.
5.5.2 How to use file register R

- The file register (R) consists of 16 bits as well as the data register (D).
- Set the file register in the standard RAM of the QCPU or a memory card (SRAM card and Flash card). The file register to be stored in the Flash card can be read from the program only. The data cannot be changed with the program.

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Memory</td>
<td>Stores parameters, programs, device comments, and device initial values.</td>
</tr>
<tr>
<td></td>
<td>(File registers cannot be stored.)</td>
</tr>
<tr>
<td>Standard RAM</td>
<td>Stores file registers of 1K to 640K (the capacity depends on the CPU type).</td>
</tr>
<tr>
<td></td>
<td>Stores parameters, programs, device comments, and device initial values.</td>
</tr>
<tr>
<td></td>
<td>(File registers cannot be stored.)</td>
</tr>
<tr>
<td>Standard ROM</td>
<td>Stores file registers of 1K to 640K (the capacity depends on the CPU type).</td>
</tr>
<tr>
<td></td>
<td>Stores parameters, programs, device comments, and device initial values.</td>
</tr>
<tr>
<td></td>
<td>(File registers cannot be stored.)</td>
</tr>
<tr>
<td>Memory card</td>
<td>Stores file registers of 1K to 4086K, (The maximum number of storable file</td>
</tr>
<tr>
<td></td>
<td>registers depends on the memory card to be used.)</td>
</tr>
</tbody>
</table>

- The data in the file register remains after the reset operation or after the power is turned off.
- To clear the data, write 0 to the file register with the MOV(P) instruction or GX Works2.
- Use [Write to PLC] of GX Works2 or a sequence program to write data to the standard RAM or SRAM card.
- Use [Export to ROM Format] of GX Works2 to copy data in the standard ROM or Flash card.
- Specify the area of the file register in 1K (1024 point) basis with the parameter.

**Application Example**

- Set 32K points of the file register R0 to R32767 to use in the program.
  Follow the procedures below to register the file register to the parameter.

1) Double-click "Parameter" in the project list.

2) "PLC Parameter", "Network Parameter", and "Remote Password" are displayed.
   Double-click "PLC Parameter"
3) The Q Parameter Setting dialog box is displayed. Click the PLC File tab.

4) Check the "Use the following file" check box and select "Memory Card (RAM) (Drive 1)" for "Corresponding Memory".

Enter the following items in "File Name" and "Capacity".

[Setting contents]
File Name : R
Capacity : 32

5) After the setting is completed, click the **End** button.

6) The message on the left is displayed. Click the **Yes** button.
7) Click [Online] → [Write to PLC] to display the Online Data Operation dialog box. Select "Parameter" in the PLC Module tab.

8) Click the **Execute** button to write the data.

- To clear the file register data with the program, write the following program. For the operation procedure, refer to section 4.4.
  Turning on X0 can write the data, and turning on X1 can clear the data.

<table>
<thead>
<tr>
<th>Project name</th>
<th>File register</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

```
X0  MOV  K173  R5
X1  MOV  K0   R5
X0  MOV  K173  D5
```

The file register data of the standard RAM is kept by the battery. Resetting or turning off the power cannot clear the data. To clear the data, write "0".

For comparison
5.6 External Setting of Timer/Counter Set Value and External Display of Current Value

The timer and counter can be specified by \( K \) (decimal constant) directly or by \( D \) (data register) indirectly. In the program shown below, the external digital switch can change the set value.

- After reading the program to GX Works2, write it to the programmable controller to check that it works properly.

Operating Procedure

The step (1) of the following procedure is the same as Operating Procedure in section 5.3.
The steps (2) to (4) of the following procedure are the same as Operating Procedure in section 4.4.

(1) Reading the data
(2) Creating a program
(3) Writing the project to the programmable controller
(4) Monitoring the ladder
Operation Practice

(1) External setting of the timer set value and display of the current value
   - Set the timer set value in the digital switch (X20 to 2F), and turn on the switch X0.
   - When the switch X4 is turned on, Y70 turns on after the time specified with the digital switch. (For example, Y70 turns on after 123.4sec. when \([1 \ 2 \ 3 \ 4]\) is set.)
   - The digital display (Y40 to 4F) displays the current value of the timer T10.

(2) External setting of the counter set value and display of the current value
   - Set the counter set value in the digital switch (X30 to 3F), and turn on the switch X1.
   - Turn on and off the switch X5 repeatedly. When X5 has turned on for the times specified with the digital switch (count up), Y71 turns on.
   - The digital display (Y50 to 5F) displays the current value of the counter C10 (the number of times that X5 is turned on).
   - Turning on the switch X6 clears the counter C10 to 0. When the contact C10 is already turned on (count up), the contact is released.
5.7 Exercise

5.7.1 Exercise 1

Transfer the eight input statuses (X0 to X7) to D0 once then output them to Y70 to Y77. (For example, Y70 turns on when X0 turns on.)

\[
\begin{align*}
X0 & \rightarrow Y70 \\
X1 & \rightarrow Y71 \\
X2 & \rightarrow Y72 \\
X3 & \rightarrow Y73 \\
X4 & \rightarrow Y74 \\
X5 & \rightarrow Y75 \\
X6 & \rightarrow Y76 \\
X7 & \rightarrow Y77
\end{align*}
\]

Create the following program with GX Works2 filling in the blanks [ ] . Then, check the operation using the demonstration machine.

**Hint**

The CPU imports the input signal as "1" when it is on, and imports as "0" when it is off. The output module turns on when the CPU outputs "1", and turns off when the CPU outputs "0".

**Comparison**

The following shows a program which is created with the sequence instructions, not with the MOV instruction.
5.7.2 **Exercise 2**  BIN and BCD conversion

Output the number of times that X1 is turned on on the display connected to Y40 to Y4F in BCD. As a precondition, the set value of the counter (C0) can be input with the digital switch (X20 to X2F) and the setting will be available by turning on X0.

Create the following program with GX Works2 filling in the blanks [ ] . Then, check the operation using the demonstration machine.

<table>
<thead>
<tr>
<th>X1</th>
<th>X0</th>
<th>SM401</th>
<th>X2</th>
<th>C0</th>
<th>Y70</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Hint**

BCD value

BCD digital switch X20 to X2F (K4X20)

1) 2) 3) 4)
Create a program in which turning on X0 turns on the 64 outputs Y40 to Y7F and turning off X0 turns off the 64 outputs Y40 to Y7F.

Create the following program with GX Works2 filling in the blanks. Then, check the operation using the demonstration machine.

X0

0

5

FMOV K255

FMOV 1)

K255FMOV 1)

K2Y40

K8

Comparison

The following shows a program which is created with the sequence instructions, not with the FMOV instruction. The 130 steps are used.
5.7.4 **Exercise 4** Comparison instruction

Using the two BCD digital switches, execute the calculation of (A - B) and display the result on the BCD digital display (Y40 to Y4F).

**Display the result of the calculation of A - B on the BCD display of Y40 to Y4F.**

When the result is a negative number, make sure that the display displays 0 and the LED of Y70 turns on.

Fill in the blanks [ ].

Then, check the operation using the demonstration machine.

**Hint**

The operation result is always output from the CPU in binary.

\[ -D0 \rightarrow D1 = D1 - D0 \rightarrow D1 \]

1)
2)
3)
4)
5.7.5 **Exercise 5**  Addition and subtraction instructions

Create a program that:
1) Imports the values specified by the digital switches (X20 to X2F) to D3 and D2 (32-bit data) when X0 is turned on, adds them to D1 and D0, and displays the result on the displays (Y40 to Y5F).
2) Imports the values specified by the digital switches (X20 to X2F) to D5 and D4 when X1 is turned on, subtracts them from D1 and D0, and displays the result.
3) When the result is a negative number, Y77 is turned on, the two's complement is determined from the result to obtain the absolute value, and displayed.

Fill in the blanks [________] in the following.
Then, check the operation using the demonstration machine.

---

### Reference

**Complement (deny transfer)**

Before DCML execution (negative number) | After DCML execution | After D + P execution (absolute value)
---|---|---
| | | |

The absolute value is determined by a calculation of two's complement of D0 and D1 (32-bit data).

**REMARK**
The CML instruction inverts the bit pattern of \( \overline{S} \) and transfers the data to \( \overline{D} \) when the input condition is turned on.
5.7.6 Exercise 6 Multiplication and division instructions

Create a program that:
1) Sets data for multiplication and division when X0 is turned on.
2) Multiplies the value specified by the digital switches X20 to X27 by the value specified by the digital switches X30 to X37 in binary when X2 is turned on.
3) Divides the value specified by the digital switches X30 to X37 by the value specified by the digital switches X20 to X27 in binary when X3 is turned on.
4) Outputs the result of the multiplication or division to the BCD displays Y40 to Y4F and the remainder to the BCD displays Y60 to Y67.

\[(X_{30} \text{ to } X_{37}) \times (X_{20} \text{ to } X_{27}) \rightarrow (Y_{40} \text{ to } Y_{4F})\]
\[(X_{30} \text{ to } X_{37}) / (X_{20} \text{ to } X_{27}) \rightarrow (Y_{40} \text{ to } Y_{4F}) \ldots (Y_{60} \text{ to } Y_{67})\]

Create the program with GX Works2 filling in the blanks \[\underline{\text{[ ]}}\] in the following. Then, check the operation using the demonstration machine.
5.7.7 **Exercise 7**  D-multiplication and D-division

Create a program that:

1) Multiplies the value set by the 5-digit digital switches (X20 to X33) by 1,100 in binary when X2 is turned on. When the result is 99,999,999 or less, it is displayed on the displays (Y40 to Y5F).

2) Divides the value set by the 8-digit digital switches (X20 to X3F) by 40,000 in binary when X3 is turned on. When X4 is on, the quotient is displayed on the displays (Y40 to Y5F). When X4 is off, the remainder is displayed on the displays (Y40 to Y5F).

\[(X20 \text{ to } X33) \times 1100 \rightarrow (Y40 \text{ to } Y5F)\]
\[(X20 \text{ to } X3F) / 40000 \rightarrow \text{Quotient (Y40 to Y5F)} \quad X4: \text{ON}\]
\[\text{Remainder (Y40 to Y5F)} \quad X4: \text{OFF}\]

Create the program with GX Works2 filling in the blanks [ ] in the following. Then, check the operation using the demonstration machine.
<table>
<thead>
<tr>
<th>Exercise No.</th>
<th>Answer</th>
</tr>
</thead>
</table>
| 1           | 1) K2X0  
2) K2Y70 | |
| 2           | 1) D0  
2) K4X20  
3) BCD  
4) K4Y40 | |
| 3           | 1) K2Y40  
2) KB  
3) K0 | |
| 4           | 1) BINP  
2) BINP  
3) >  
4) <= | |
| 5           | 1) D + P  
2) D - P  
1) BINP | |
| 6           | 2) BINP  
3) *P  
4) /P  
5) BCD  
6) BCD | |
| 7           | 1) DMOVP  
2) D*P  
3) DBINP  
4) DMOVP  
5) D/P  
6) D14  
7) D16 | |
CHAPTER 6  HOW TO USE OTHER FUNCTIONS

6.1  Test Function at Online

As a preparation, follow the procedure below.

1) Read a project with GX Works2.
2) Write the parameter and program of the read project to the CPU (programmable controller). (The CPU must be stopped.)
3) Set GX Works2 to the monitor mode.
4) Confirm the program displayed on the screen.

For details on the operation method, refer to chapter 2.
6.1.1 Turning on and off the device "Y" forcibly

Stop the CPU before this operation.

1) Click [Debug] → [Modify Value].

2) The Modify Value dialog box is displayed. Enter "Y70" in the "Device/Label" list box.

3) Click the [ON] or [OFF] button to turn on or off "Y70" forcibly.

---

Check with demonstration machine

1) Confirm that the on and off statuses on the Execution Result area switches according to the clicking of the [ON] or [OFF] button. Also, confirm that the LED of Y70 on the demonstration machine turns on and off according to the operation.

---

NOTE

When the CPU is in the RUN state, the operation results of programs are displayed preferentially. Therefore, stop the CPU first before the confirmation with the demonstration machine.
The test function during ladder monitoring of GX Works2 is also available for setting and resetting contacts, changing current values, and outputting forcibly word devices.

Double-clicking a contact (pressing the Enter key) holding the Shift key in the ladder monitoring screen of GX Works2 switches the contact open or close forcibly.

To display the Modify Value dialog box, double-click a word device (press the Enter key) holding the Shift key in the ladder monitoring screen of GX Works2.
6.1.2 Setting and resetting the device "M"

Activate the CPU before this operation.

1) Click [Debug] → [Modify Value].

2) The Modify Value dialog box is displayed. Enter "M10" in the "Device/Label" list box.

3) Click the [ON] or [OFF] button to set or reset "M10".

Check with demonstration machine

Turn off X4 and check the following.

1) When M10 is set, M10 becomes non-conductive and the current value of the timer T0 is cleared to 0.
   Check that the value on the digital display (Y50-Y5F) does not change.

2) When M10 is reset, M10 is conducted and the timer T0 starts counting from 0. This count value increases every 10 seconds.
   Confirm that the value on the display (Y50-Y5F) increases every 10 seconds.

POINT

With the same procedure, bit devices other than the internal relay (M) also can be set or reset forcibly.
### 6.1.3 Changing the current value of the device "T"

1. **Activate the CPU before this operation.**

2. 1) Click [Debug] → [Modify Value].

3. 2) The Modify Value dialog box is displayed. Enter "T0" in the "Device/Label" list box.

4. 3) Select "Word[Signed]" from the "Data Type" list box.

5. 4) Enter "1000" in the "Value" column.

6. 5) After the setting is completed, click the [Set] button to change the current value of T0 to 1000 forcibly.

### Check with demonstration machine

1. Confirm that the value on the digital display (Y50-Y5F) is 1000 when the key is pressed.

### POINT

With the same procedure, the current values of word devices other than the timer (T) also can be changed.
6.1.4 Reading error steps

1) Click [Diagnostics] → [PLC Diagnostics].

2) The PLC Diagnostics dialog box is displayed. Click the [Error Jump] button to jump to the highlighted sequence program step number where the selected error occurred.

- An error number is displayed if an error occurred.
- "No Error" is displayed if no error occurred.
6.1.5 Remote STOP and RUN

1) Click [Online] → [Remote Operation].

2) The Remote Operation dialog box is displayed. Select "STOP" from the list in the Operation area.

3) After the setting is completed, click the [Execute] button.

4) The message "Do you want to execute the operation(STOP)?" is displayed. Click the [Yes] button.

The operation of the CPU stops.

5) Select "RUN" in step 2), and perform steps 2) to 4) again.

The CPU, which was stopped in the above operation, starts the operation again.

Activate the CPU before this operation.
6.2 Forced I/O Assignment by Parameter Settings

1) Double-click "Parameter" in the project list.

2) "PLC Parameter", the "Network Parameter" folder, and "Remote Password" are displayed. Double-click the "PLC Parameter".

3) The Q Parameter Setting dialog box is displayed. Click the "I/O Assignment" tab.

4) Select "Input" from the list box of the "Type" column.

5) Enter "QX42" in the "Model Name" column.

6) Select "32Points" from the list box of "Points" column.

7) Enter "0000" in the "Start XY" column.

8) After the setting is completed, click the End button.

After this exercise is finished, initialize the settings by the following procedure.

1) Click the Default button in the Q Parameter Setting dialog box to initialize the parameter settings.

2) Click on the toolbar and write only the parameters to the CPU.
Check with demonstration machine

Stop the CPU and click on the toolbar.
The Online Data Operation dialog box is displayed. Click the parameter of the currently edited data, and click the button to write only the parameters to the CPU. Then, activate the CPU and check the following.

1) The current value of the timer T0 disappears from the digital display (Y50 to Y5F). Then, the LEDs of Y70 to Y77 start flashing until the set values of Y70 to Y77 reach each set device value.
2) Turning on X6 to output the signal to Y70 and Y74 does not turn the LEDs of Y70 and Y74.

[I/O numbers before the forced assignment]

<table>
<thead>
<tr>
<th>Q61P Power supply module</th>
<th>Q06 CPU</th>
<th>Vacant slot</th>
<th>Q42</th>
<th>QY42P</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

Sixteen points occupy one address. Sixty-four points occupy four addresses.

[X is set to 32 points, therefore only two addresses are available. Hereafter, the addresses become smaller by two.]

[Slot 0]
Address 0, Address 2, Address 1, Address 3

[Slot 1]
Address 4, Address 6, Address 5, Address 7

Sixty-four points occupy four addresses.

[Digital display]

[BCD T0 K4Y50]

Y5F-Y50

Y77 Y76 Y75 Y74 Y73 Y72 Y71 Y70

Name plate

[I/O numbers after the forced assignment]

<table>
<thead>
<tr>
<th>Q61P Power supply module</th>
<th>Q06 CPU</th>
<th>Vacant slot</th>
<th>Q42</th>
<th>QY42P</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

X is set to 32 points, therefore only two addresses are available. Hereafter, the addresses become smaller by two.

[Digital display]

[BCD T0 K4Y50]

Y5F-Y50

Y77 Y76 Y75 Y74 Y73 Y72 Y71 Y70

Name plate

POINT

- The address 7 is replaced with the address 5. Therefore, the current value of the timer T0 is output to the newly assigned address 5, and LEDs of Y70 to Y77 which are connected to the address 5 flash.
- Results of outputting the signals to Y70 or Y74 are not displayed on any displays since the address 7 of the output modules no longer exists.
- To display the device numbers normally, change the device number K4Y50 K4Y30, and Y70 to Y77 Y50 to Y57.
6.3 How to Use Retentive Timers

When an input condition is turned on, the coil is energized. Then the value of a retentive timer starts increasing. When the current value reaches the set value, the retentive timer goes time-out and the contact turns on. When the input condition is turned off during the increasing, the coil is de-energized but the current value is kept. When the input condition is turned on again, the coil is re-energized and the current value is accumulated.

<table>
<thead>
<tr>
<th>Project name</th>
<th>Retentive timer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

In the example below, the retentive timer is set to ST0 to ST31.

1) Double-click "Parameter" in the project list.

2) "PLC Parameter", the "Network Parameter" folder, and "Remote Password" are displayed. Double-click the "PLC Parameter".

(To the next page)
3) The Q Parameter Setting dialog box is displayed. Click the "Device" tab.

4) Click "Device Points" in the "Retentive Timer" row, and enter "32".

5) After the setting is completed, click the End button.
6.4 Device Batch Replacement

6.4.1 Batch replacement of device numbers

This section explains how to replace Y40 to Y7F (64 devices) with Y20 to Y5F (64 devices) in batch.

1) Click [Find/Replace] → [Device Batch Replace].

2) The Find/Replace dialog box is displayed. Enter "Y40" in the "Find Device" column.

3) Enter "Y20" in the "Replace Device" column.

4) Enter "64" in the "Points" column.

5) After the setting is completed, click the Execute button.

6) Confirm that the target device numbers are replaced.
6.4.2 Batch change of specified devices between normally open contacts and normally closed contacts

This section explains how to change the normally open contacts of the specified devices to the normally closed contact and vice versa in batch.

1) Click [Find/Replace] → [Change Open/Close Contact].

2) The Find/Replace dialog box is displayed. Enter "X4" in the "Replace Device" list box.

3) After the setting is completed, click the All Replace button.

4) Confirm that the normally open contact is changed to the normally closed contact and vice versa.

NOTE

Before exercising section 6.5 after this section, write the program in the personal computer to the CPU.
For the write operation, refer to section 2.7.
6.5 Online Program Change

This function is used to write programs to the CPU that is running.

Activate the CPU before this operation.

1) Change the ladder.
   (In the example, change "X1" to "X0").

2) After the change, click [Compile] → [Online Program Change].

3) The dialog box for "Caution" is displayed. Click the [Yes] button to accept the change.

4) The message "Online change has completed." is displayed. Click the OK button.

NOTE

Online program change cannot be executed when the program in the programmable controller CPU and the program in GX Works2 before the modification do not match. Therefore, when whether the programs match or not is unclear, verify them before the modification with GX Works2, and execute the online program change.
6.6 Registering Devices

This section explains how to register multiple devices or labels in one screen and to monitor them at the same time.

1) Click [View] → [Docking Window] → [Watch(1 to 4)].

* In this example, select "1".

2) The Watch 1 window is displayed. Select a row to be edited. Enter "T0" in the Device Label column.

3) The input device or label is registered.

4) Click [Online] → [Watch] → [Start Watching].

The current value of the registered device or label is displayed in the window.
6.7 How to Create Comments

The following is an example of a printed out ladder with comments.

Use the keyboard to input the program above or read it from a folder on the desktop.
(1) Flowchart of when creating comments

Set the device range on which comments are attached.*

Double-click the comment file on the workspace.

Create comments.

Save the project.

Read and confirm the ladder with comments.

*: This procedure is necessary for specifying the device comment range.

POINT

Comments are used for displaying functions or applications of each device. Up to 32 characters are available.
(2) Creating comments

1) Double-click "Global Device Comment" in the project list. The Device Comment screen is displayed.

2) Click a comment area and enter a comment as shown on the left.

3) Enter "Y70" in the "Device Name" list box.

4) Press the Enter key.

5) Click a comment area and enter a comment as shown on the left.

(To the next page)
6) Enter "M1" in the "Device Name" list box.

7) Press the Enter key.

8) Click a comment area and enter a comment as shown left.

9) Enter "T0" in the "Device Name" list box.

10) Press the Enter key.

11) Click a comment area and enter a comment as shown left.

12) Enter "C2" in the "Device name" list box.

13) Press the Enter key.

14) Click a comment area and enter a comment as shown left.
(3) Saving comments

1) Click [Project] → [Save As].

2) The Save As dialog box is displayed. Specify (or select) a workspace name and click the **Save** button.
(4) Displaying a ladder with comments on GX Works2 screens

1) Click [View] → [Comment].

2) Comments are displayed on the ladder screen.
In addition to device comments, statements and notes can be created on the ladder screen.

- **Statement**: Comment for explaining functions or applications for the ladder block. Up to 64 characters are available.
- **Note**: Comment for explaining functions or applications for outputs and commands. Up to 32 characters are available.

**Creating statements**

Click and double-click a symbol where a comment is to be attached. The Enter Line Statements dialog box is displayed. Enter a comment and click the OK button.

**Creating notes**

Click and double-click an output or a command where a comment is to be attached. The Enter Symbol dialog box is displayed. Enter a comment and click the OK button.

- Select "In PLC" or "In Peripheral" for statements and notes.
  - "In PLC" : The data of statements and notes is stored as a part of a program. This enables the data to be stored in CPUs at factories. However, a lot of program memory capacity of the programmable controller CPU is required.
  - "In Peripheral" : The data of statements and notes is stored in the peripheral device (personal computer) separated from the program. Since a program requires only one extra step per one location, less program memory capacity is required on the programmable controller CPUs. However, when programs are modified at factories, programs in GX Works2 in the peripheral device (personal computer) and those in programmable controller CPUs do not match. Carefully handle the data.
6.8 Setting Security for Projects

This section explains how to set security for projects to protect the projects and the data in the projects.

Setting security restricts accesses to projects.

Also, setting security prevents data such as POUs, device comments, and parameters, which are created by the user, from erroneous modifications or disclosures to unauthorized users.

### POINT

Access levels and access authority

Setting an access level to each user restricts accesses to each data.

An access level is an operating authority given to a login user of the project.

The following five levels are available as the access levels. Data that can be edited by a user having lower access level can also be edited by a user having higher access level.

<table>
<thead>
<tr>
<th>Access level</th>
<th>Operating authority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Higher</td>
<td></td>
</tr>
<tr>
<td>Administrators</td>
<td><strong>&lt;Administrator level&gt;</strong> All operations are possible.</td>
</tr>
<tr>
<td>Developers (Level 3)</td>
<td><strong>&lt;Developer level&gt;</strong> Security setting, data access, and a part of operations are restricted.</td>
</tr>
<tr>
<td>Developers (Level 2)</td>
<td></td>
</tr>
<tr>
<td>Developers (Level 1)</td>
<td></td>
</tr>
<tr>
<td>Lower</td>
<td></td>
</tr>
<tr>
<td>Users</td>
<td><strong>&lt;Operator level&gt;</strong> Only access to project data is possible. Data cannot be read from the programmable controller CPU.</td>
</tr>
</tbody>
</table>

**<Example>**
The data with access authority of Developers (Level 2) can be edited by login users with the access level of Developers (Level 2) or higher (Administrators, Developers (Level 3), or Developers (Level 2)).
6.8.1 Setting and resetting security for projects

This section explains how to set security for an open project and how to reset the security.

(1) Setting security for projects
Set a security for a project.
Once security is set for a project, user authentication is required when the project is opened again.

1) Click [Project] → [Security] → [User Management].

2) The Use Addition dialog box is displayed. Enter the following items.

   User Name : MITSUBISHI
   Password   : MITSUBISHI
   Re-enter Password : MITSUBISHI

   * When the user name or login password is lost, logging in to the project is disabled. Do not enter any other user name or password other than the above.

3) After entering them, click the OK button.

   Security is set for the project.

(2) Resetting security for projects
Deleting all users resets the set security of a project and returns the project to the status without security. (Refer to section 6.8.2.)
6.8.2 Managing (adding, deleting, and changing) users

This section explains how to manage the registered statuses of users for a project with security and how to add, delete, and change users. This function is available only when a user logs in a project with the access level of "Administrators" or "Developers".

[Displaying the User Management screen]

1) Click [Project] → [Security] → [User Management].

2) The User Management dialog box is displayed.

The methods for adding users, changing user information, changing passwords, and deleting users are explained from the next page.
[Adding users]
Add a user to a project with security.
A user whose access level is higher than that of the login user cannot be added.

1) Click the Add... button on the User Management screen.

2) The User Addition dialog box is displayed. Enter the following items.
   - User Name: Developers
   - Access Level: Developers(Level3)
   - Password: Developers
   - Re-enter Password: Developers

3) After entering them, click the OK button.

4) The user (Developers(Level3)) is added.
[Changing user information]
Change the access level of the user added on the previous page from "Developers(Level3)" to "Users".
The information of the login user and of a user whose access level is higher than that of the login user cannot be changed.

1) Select the user name "Developers".

2) Click the Change button.

3) The Change User Data dialog box is displayed. Select "Users" from the "Access Level" list box.

4) After selecting it, click the OK button.

5) The access level of the user "Developers" is changed.
[Changing passwords]
Change the password of a user selected in the list on the User Management screen. The password of the login user and of a user whose access level is higher than that of the login user cannot be changed.
To change the password of the login user, click [Project] → [Security] → [Change Password].

1) Select the user name "Developers".
2) Click the [Password Setup] button.

3) The Change Password dialog box is displayed. Enter the following items.
New Password : Users1
Re-enter Password : Users1
4) After entering them, click the [OK] button.

The password of the user "Developers" is changed.

[Deleting users]
Delete a user selected in the list with the [Delete...] button on the User Management dialog box.
The current login user cannot be deleted.
However, when the registered user is only "Administrators" and no other users to be deleted exist, the current login user can be deleted.
When all users are deleted, security is reset.
6.8.3 Logging in projects

A user authentication is required for opening a project with security.

1) When a project with security is opened, the User Authentication screen is displayed. Enter a user name and a password for log-in, and click the OK button.

Enter the following user name and password, which are set in section 6.8.1.

User Name : MITSUBISHI
Password : MITSUBISHI

2) The project is displayed.
6.8.4 Changing access authority for each access level

This section explains how to set an authorization of displaying and saving data for each access level. The access authority of access levels higher than that of the login user cannot be changed. When the access level of the current login user is "Users", the access authority cannot be changed.

1) Click [Project] → [Security] → [Data Security Setting].

2) The Data Security Setting dialog box is displayed.

3) Select a target item from Access Object.

4) Set "Enable" or "Disable" for reading and writing data from Access Authority for each access level by moving the slider.

5) Click the OK button.
6.9 Sampling Trace Function

This function is used to acquire data at the specified timing to find how device values change during program operation and to trace the changes displayed in time series. For details of the sampling trace function, refer to the manuals of each CPU module.

In this example, the device value at an error occurrence is acquired.

As a preparation, follow the procedure below.

1) Click the "PLC RAS" tab on the Q Parameter Setting dialog box.

2) Select "Continue" from the Computation Error list box in the Operating Mode When There is an Error area.

3) Click the End button.

4) Write parameters and programs to the CPU.
(1) Setting the sampling trace

1) Click [Debug] → [Sampling Trace] → [Open Sampling Trace].

2) The Sampling Trace screen is displayed.

3) Click [Debug] → [Sampling Trace] → [Trace Setting].

(To the next page)
4) The Trace Setting dialog box is displayed. Select “Standard RAM” from the “Target Memory” list box.

5) Click the Condition Setting tab.

6) Check “Detail Setting” in the Trigger Condition Setting area and click the [Setting Change] button.
7) The Detail Setting - Trigger Condition dialog box is displayed. Set the following items. In this example, set error occurrence as trigger condition.

Device/Label : SM0
Condition : -P-

8) Click the End Setting button.

9) The Detail Setting - Trigger Condition dialog box disappears. Click the End Setting button to close the Trace Setting dialog box.

10) Set devices to be traced on the Sampling Trace screen as shown on the left.

11) Check the check box to display the trend graph of SD0.

12) “SD0” is displayed in the trend graph area on the Sampling Trace screen.
(2) Starting the sampling trace

1) Click [Debug] → [Sampling Trace] → [Start Trace].

2) The message shown on the left is displayed. Click the **Yes** button.

3) The Trace Data Storage Status screen is displayed when the sampling trace is started.

   After confirming that the total data reaches 100%, operate digital switches to generate an error.

4) The trace result is displayed on the Sampling Trace screen.
(3) Checking the trace result

1) Scroll the trend graph screen to the trigger point to check the device value at an error occurrence.

POINT

Saving trace data to a personal computer
Click [Debug] → [Sampling Trace] → [Export CSV Data]. The following dialog box is displayed.

After entering a file name, click the Save button.
CHAPTER 7  PROGRAMMING INTELLIGENT FUNCTION MODULE

7.1  Intelligent Function Module

(1) Intelligent function module type
On programmable controller CPUs (hereinafter referred to as QCPUs), some functions are not supported or are limited in use. Intelligent function modules support those functions instead of QCPUs. Therefore users need to select an intelligent function module that is appropriate for the purpose involved.
QCPUs are compatible with QCPU-compatible intelligent function modules. The following table shows examples of the intelligent function modules.

Table 7.1  Example of intelligent function module

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of I/O occupied points</th>
<th>Function</th>
<th>Module current consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog-digital converter module</td>
<td>16 points</td>
<td>Input module that converts; 0 to 20mA → 0 to 4000 (in standard resolution mode), 0 to ±10V → 0 to ±4000 (in standard resolution mode)</td>
<td>5VDC 0.63A</td>
</tr>
<tr>
<td>module (Q64AD)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Digital-analog converter module</td>
<td>16 points</td>
<td>Output module that converts; 0 to 4000 → 0 to 20mA (in standard resolution mode), 0 to ±4000 → 0 to ±10V (in standard resolution mode)</td>
<td>5VDC 24VDC 0.33A 0.12A</td>
</tr>
<tr>
<td>module (Q62DAN)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(2) Using intelligent function modules with CPUs
An intelligent function module can be installed on any I/O slots on a main base unit and extension base unit.
7.2 Data Communication between Intelligent Function Modules and CPUs

An intelligent function module and a CPU exchange mainly two formats of data. 

Bit data ----------- Signals that use input Xs and output Ys 

Word data -------- 16-bit data or 32-bit data

Figure 7.2 Internal configuration of the intelligent function module
7.2.1 I/O signals to CPUs

For 1-bit signals exchanged between a QCPU and an intelligent function module, input Xs and output Ys are used. Xs and Ys here do not mean external I/Os but symbols that are used in a sequence program to exclusively represent I/O signals of intelligent function modules. Also note that I/O numbers are assigned according to the slot where the intelligent function module is installed.

[X]

![Figure 7.3](image)

Xs in a sequence program represent signals that are input to a QCPU from an intelligent function module. These signals are generated on an intelligent function module. Note that the Xs are used as contacts in a program. The following is examples of the signals.

1) READY signal
   This signal notifies a QCPU that an intelligent function module started up normally at power-on and is ready for operation.

2) Operating condition setting completed
   This signal is used as an interlock condition for turning Operating condition setting request (Y9) on/off when the following settings are changed.
   - A/D conversion enable/disable setting (buffer memory address 0: Un\G0)
   - CH\ Average time/average number of times
     (buffer memory addresses 1 to 8: Un\G1 to Un\G8)
   - Averaging process setting
     (buffer memory address 9: Un\G0)

[Y]

![Figure 7.4](image)

SETs, RSTs, or OUT-Ys represent output signals transmitted from a QCPU to an intelligent function module. These signals are generated on a QCPU. Note that they are used as coils or contacts in a program.

(Example) D/A converter modules output an enable instruction (output enable) before outputting analog values that were converted from digital values.
7.2.2 Data communication with intelligent function modules

Data is transmitted or received in 16-bit or 32-bit units. Intelligent function modules have a buffer memory to store those data.

<table>
<thead>
<tr>
<th>Buffer memory address</th>
<th>D/A conversion enable/disable</th>
<th>CH1 Digital value</th>
<th>CH2 Digital value</th>
<th>System area (use prohibited)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 to 12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 7.5 Buffer memory

(1) QCPUs can read and write data to and from the buffer memory. Also note that some modules can write data to buffer memory from peripheral device via an interface.

(2) In a buffer memory, space of one word (16 bits) is reserved for each intelligent function module's unique address. The smallest address is 0, and these addresses are used to specify a target module to read or write. The minimum unit is one word. Data of 17 bits to 32 bits is treated as 2-word (32-bit) data.

Figure 7.6 Example image of buffer memory content (D/A converter module)

Figure 7.6 shows 16 bits of the buffer memory of a D/A converter module where digital quantities have been written. The number is obtained from digital quantity that a QCPU wrote to the buffer memory within the range from -4096 to +4095 in signed binary (16 bits long).

(3) A buffer memory is a RAM.
7.3 Communication with Intelligent Function Module

7.3.1 Communication methods with intelligent function modules

The following table shows the communication methods between a QCPU and an intelligent function module.

**Table 7.2 Communication method with intelligent function modules**

<table>
<thead>
<tr>
<th>Communication method</th>
<th>Function</th>
<th>Setting method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial setting, Auto refresh setting</td>
<td>Performs initial settings and auto refresh settings of intelligent function modules. These settings allow writing/reading data to/from intelligent function modules regardless of communication program creation or buffer memory address.</td>
<td>Use GX Works2.</td>
</tr>
<tr>
<td>Ex.) When A/D converter module Q64AD is used</td>
<td>• Initial setting : • A/D conversion enable/disable setting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Sampling/averaging processing specification,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Time average/number of times average specification,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Average time/average number of times specification</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Set data in auto refresh settings is stored to the intelligent function module parameter on a QCPU.)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Auto refresh setting : Set a device on a QCPU to store the following data to.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Digital output from Q64AD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Maximum and minimum values of Q64AD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Error code</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Set data in auto refresh settings is stored to the intelligent function module parameter on a QCPU.)</td>
<td></td>
</tr>
<tr>
<td>Device initial value</td>
<td>Writes set data in device initial settings of intelligent function modules to the intelligent function modules at the following timings.</td>
<td>Use GX Works2 to specify the range for intelligent function module devices (U□G□).</td>
</tr>
<tr>
<td></td>
<td>• At power-on of a QCPU</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• At reset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• At switching from STOP to RUN</td>
<td></td>
</tr>
<tr>
<td>FROM/TO instruction</td>
<td>Read or write data from or to the buffer memory on an intelligent function module.</td>
<td>Use this instruction in a sequence program.</td>
</tr>
<tr>
<td>Intelligent function module device (U□G□)</td>
<td>Directly handles the buffer memory on an intelligent function module as a device of a QCPU. Unlike &quot;FROM/TO instruction&quot;, this requires only one instruction for processing data that is read from an intelligent function module.</td>
<td>Specify this device as a device in a sequence program.</td>
</tr>
<tr>
<td>Intelligent function module dedicated</td>
<td>Used to simplify programming for using the functions of intelligent function modules.</td>
<td>Use this instruction in a sequence program.</td>
</tr>
<tr>
<td>instruction</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7.4 Intelligent Function Module System in Demonstration Machine

Use an A/D or D/A converter module to convert analog signals/digital data that are input with the volume or digital switch on the demonstration machine.
7.5 Q64AD Analog/Digital Converter Module

7.5.1 Names of parts

The following explains the parts of Q64AD. For details, refer to the User’s Manual.

---

<table>
<thead>
<tr>
<th>No.</th>
<th>Name and appearance</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1)  | RUN LED             | Indicating the operation status of the A/D converter module.  
ON    : In normal operation  
Flicker: In offset/gain setting mode  
OFF   : 5V power failure or watchdog timer error occurred |
| 2)  | ERROR LED           | Indicating errors and the status of the A/D converter module.  
ON    : Error occurred  
OFF   : In normal operation  
Flicker: Switch setting error occurred  
Values other than 0 has been set to the switch 5 on the intelligent function module. |
7.5.2 A/D conversion characteristics

(1) A/D conversion characteristics on voltage inputs
(For analog input range from -10 to 10V in a standard resolution mode)

![Graph showing A/D conversion characteristics for voltage inputs.]

Figure 7.12 A/D conversion characteristics (voltage input)

A/D converter modules convert analog values input from other devices to digital quantities so that CPUs can operate those values. On voltage inputs, for example, A/D converter modules convert -10V to a quantity of -4000 and 10V to 4000. This means that the modules convert an input voltage of 2.5mV to a digital quantity of 1, and abandon values smaller than 2.5mV.

(2) A/D conversion characteristics on current inputs
(For analog input range from 0 to 20mA in a standard resolution mode)

![Graph showing A/D conversion characteristics for current inputs.]

Figure 7.13 A/D conversion characteristics (current input)

The modules convert current an input of 0mA to 0 for an output, and 20mA to 4000. This means that the modules convert an input current of 5μA to a digital quantity of 1, and abandon values smaller than 5μA.

**REMARK**

A voltage or current value that is equivalent to a digital value of 1 through A/D conversion (maximum resolution) differs depending on the setting of the resolution mode (1/4000, 1/12000, 1/16000) or the output range.
7.5.3 List of I/O signals and buffer memory assignment

(1) List of I/O signals

The following shows a list of the I/O signals for the A/D converter modules. Note that I/O numbers (X/Y) shown in this section and thereafter are the values when the start I/O number for the A/D converter module is set to 0.

<table>
<thead>
<tr>
<th>Device No. (input)</th>
<th>Signal name</th>
<th>Device No. (output)</th>
<th>Signal name</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0</td>
<td>Module READY</td>
<td>Y0</td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>Temperature drift compensation flag</td>
<td>Y1</td>
<td></td>
</tr>
<tr>
<td>X2</td>
<td></td>
<td>Y2</td>
<td></td>
</tr>
<tr>
<td>X3</td>
<td></td>
<td>Y3</td>
<td></td>
</tr>
<tr>
<td>X4</td>
<td></td>
<td>Y4</td>
<td></td>
</tr>
<tr>
<td>X5</td>
<td></td>
<td>Y5</td>
<td></td>
</tr>
<tr>
<td>X6</td>
<td></td>
<td>Y6</td>
<td></td>
</tr>
<tr>
<td>X7</td>
<td></td>
<td>Y7</td>
<td></td>
</tr>
<tr>
<td>X8</td>
<td>High resolution mode status flag</td>
<td>Y8</td>
<td></td>
</tr>
<tr>
<td>X9</td>
<td>Operating condition setting completed flag</td>
<td>Y9</td>
<td>Operating condition setting request</td>
</tr>
<tr>
<td>XA</td>
<td>Offset/gain setting mode flag</td>
<td>YA</td>
<td>User range writing request</td>
</tr>
<tr>
<td>XB</td>
<td>Channel change completed flag</td>
<td>YB</td>
<td>Channel change request</td>
</tr>
<tr>
<td>XC</td>
<td>Use prohibited*1</td>
<td>YC</td>
<td>Use prohibited*1</td>
</tr>
<tr>
<td>XD</td>
<td>Maximum value/minimum value reset completed flag</td>
<td>YD</td>
<td>Maximum value/minimum value reset request</td>
</tr>
<tr>
<td>XE</td>
<td>A/D conversion completed flag</td>
<td>YE</td>
<td>Use prohibited*1</td>
</tr>
<tr>
<td>XF</td>
<td>Error flag</td>
<td>YF</td>
<td>Error clear request</td>
</tr>
</tbody>
</table>

POINT

*1: These signals cannot be used by the user since they are for system use only. If these are turned on/off by the sequence program, the functioning of the A/D converter module cannot be guaranteed.
(2) Buffer memory assignment (Q64AD)
This section explains the assignment of the Q64AD buffer memory.

**POINT**
Do not write data to the system areas or areas to which writing data from a sequence program is disabled. Doing so may cause malfunction.

Buffer memory assignment (Q64AD) (1/2)

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Default</th>
<th>Read/write*1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0H</td>
<td>A/D conversion enable/disable setting</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>1H</td>
<td>CH1 Average time/average number of times</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>2H</td>
<td>CH2 Average time/average number of times</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>3H</td>
<td>CH3 Average time/average number of times</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>4H</td>
<td>CH4 Average time/average number of times</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>5H</td>
<td>Averaging process setting</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>6H</td>
<td>A/D conversion completed flag</td>
<td>0</td>
<td>R</td>
</tr>
<tr>
<td>7H</td>
<td>CH1 Digital output value</td>
<td>0</td>
<td>R</td>
</tr>
<tr>
<td>8H</td>
<td>CH2 Digital output value</td>
<td>0</td>
<td>R</td>
</tr>
<tr>
<td>9H</td>
<td>CH3 Digital output value</td>
<td>0</td>
<td>R</td>
</tr>
<tr>
<td>AH</td>
<td>CH4 Digital output value</td>
<td>0</td>
<td>R</td>
</tr>
<tr>
<td>BH</td>
<td>Offset/gain setting mode Offset specification</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>CH</td>
<td>Offset/gain setting mode Gain specification</td>
<td>0</td>
<td>R/W</td>
</tr>
</tbody>
</table>

*1: Indicates whether reading from and writing to a sequence program are enabled.
R: Read enabled
W: Write enabled
## Buffer memory assignment (Q64AD) (2/2)

<table>
<thead>
<tr>
<th>Address</th>
<th>Hexadecimal</th>
<th>Decimal</th>
<th>Description</th>
<th>Default</th>
<th>Read/Write*1</th>
</tr>
</thead>
<tbody>
<tr>
<td>18H</td>
<td>24</td>
<td></td>
<td>System area</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1DH</td>
<td>29</td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1EH</td>
<td>30</td>
<td>24</td>
<td>CH1 Maximum value</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>1FH</td>
<td>31</td>
<td>25</td>
<td>CH1 Maximum value</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>20H</td>
<td>32</td>
<td>26</td>
<td>CH2 Maximum value</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>21H</td>
<td>33</td>
<td>27</td>
<td>CH2 Maximum value</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>22H</td>
<td>34</td>
<td>28</td>
<td>CH3 Maximum value</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>23H</td>
<td>35</td>
<td>29</td>
<td>CH3 Maximum value</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>24H</td>
<td>36</td>
<td>30</td>
<td>CH4 Maximum value</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>25H</td>
<td>37</td>
<td>31</td>
<td>CH4 Maximum value</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>26H</td>
<td>38</td>
<td></td>
<td>System area</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9DH</td>
<td>157</td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9EH</td>
<td>158</td>
<td></td>
<td>Mode switching setting</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>9FH</td>
<td>159</td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A0H</td>
<td>160</td>
<td></td>
<td>System area</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C7H</td>
<td>199</td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CBH</td>
<td>200</td>
<td>160</td>
<td>Pass data classification setting*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>C9H</td>
<td>201</td>
<td>161</td>
<td>System area</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CAH</td>
<td>202</td>
<td>162</td>
<td>CH1 Industrial shipment settings offset value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>CBH</td>
<td>203</td>
<td>163</td>
<td>CH1 Industrial shipment settings gain value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>CCH</td>
<td>204</td>
<td>164</td>
<td>CH2 Industrial shipment settings offset value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>CDH</td>
<td>205</td>
<td>165</td>
<td>CH2 Industrial shipment settings gain value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>CEH</td>
<td>206</td>
<td>166</td>
<td>CH3 Industrial shipment settings offset value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>CFH</td>
<td>207</td>
<td>167</td>
<td>CH3 Industrial shipment settings gain value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>D0H</td>
<td>208</td>
<td>168</td>
<td>CH4 Industrial shipment settings offset value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>D1H</td>
<td>209</td>
<td>169</td>
<td>CH4 Industrial shipment settings gain value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>D2H</td>
<td>210</td>
<td>170</td>
<td>CH1 User range settings offset value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>D3H</td>
<td>211</td>
<td>171</td>
<td>CH1 User range settings gain value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>D4H</td>
<td>212</td>
<td>172</td>
<td>CH2 User range settings offset value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>D5H</td>
<td>213</td>
<td>173</td>
<td>CH2 User range settings gain value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>D6H</td>
<td>214</td>
<td>174</td>
<td>CH3 User range settings offset value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>D7H</td>
<td>215</td>
<td>175</td>
<td>CH3 User range settings gain value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>D8H</td>
<td>216</td>
<td>176</td>
<td>CH4 User range settings offset value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>D9H</td>
<td>217</td>
<td>177</td>
<td>CH4 User range settings gain value*2</td>
<td>0</td>
<td>R/W</td>
</tr>
</tbody>
</table>

*1: Indicates whether reading from and writing to a sequence program are enabled.
R: Read enabled
W: Write enabled

*2: Areas used to restore the user range settings offset/gain values when online module change is made.
7.5.4 Adding or setting intelligent function module data

This section explains how to set the intelligent function module data. After an intelligent function module is added to a project, the data settings (parameters and switch settings) of the intelligent function module can be set.

1) Click [Project] → [Intelligent Function Module] → [New Module].

2) The New Module dialog box is displayed.

3) Set the A/D converter module setting as follows.
   - Module Type: Analog Module
   - Module Name: Q64AD
   - Mounted Slot No.: 3
   (Specify start XY address: 0080)

4) Click the OK button.

5) The specified intelligent function module data are added to the Project window.
6) Double-click Switch Setting.

7) The Switch Setting screen is displayed. Set Input range for CH1 to "0 to 10V".

8) Click the **OK** button.

9) Double-click Parameter.

10) The Parameter screen is displayed. Set "A/D conversion enable/disable setting" for CH2 to CH4 to "1:Disable". (Only CH1 is used.)
11) Double-click Auto Refresh.

12) The Auto Refresh screen is displayed. Set Digital output value for CH1 to "D10".

13) Click [Project] → [Intelligent Function Module] → [Intelligent Function Module Parameter List].
14) Check that "Setting Exist" is checked in Initialization (Count) and Auto Refresh (Count) for Q64AD in the Intelligent Function Module Parameter List dialog box.

15) Click the **Close** button.
7.5.5 Exercise with the demonstration machine

(1) Sequence program
The sequence program executes a sampling processing on analog voltages input through CH1 of Q64AD, and then converts the analog values to digital values.
Set the start XY of Q64AD to 80 as explained before.

<table>
<thead>
<tr>
<th>Project name</th>
<th>Q64AD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

X3: Displays digital conversion value of CH1 on LED

X80: Module READY signal
X8E: A/D conversion completed flag

At power-on or reset of a programmable controller CPU, this flag turns on if A/D conversion is ready to be executed. A/D conversion is executed once this flag turned on.

(2) Operation of the demonstration machine
Stop the CPU and click on the toolbar.
The Online Data Operation dialog box is displayed. Click the Parameter + Program button, then click the Execute button to write data to the CPU. After that, activate the CPU and check the following items.

(a) Turn on X3, and change input voltages for an A/D converter module with the volume on the demonstration machine.
Analog values that have been input to the channel 1 (CH1) of Q64AD are stored to the buffer memory (in digital value). With the auto refresh settings, the QCPU reads the stored digital values and stores them in its data register D10.

(b) Whenever an analog value is "-1" or smaller, 0 is set.

(c) The digital values are displayed on the digital display (Y50 to Y5F).
7.6 Q62DAN Digital/Analog Converter Module

7.6.1 Names of parts

The following explains the parts of Q62DAN. For details, refer to the User's Manual.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name and appearance</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1)  | RUN LED                           | Indicates the operation status of the D/A converter module.  
|     |                                   | ON : In normal operation  
|     |                                   | Flicker : In offset/gain setting mode  
|     |                                   | OFF : 5V power failure or watchdog timer error occurred                                                                                   |
| 2)  | ERROR LED                         | Indicates errors and the status of the D/A converter module.  
|     |                                   | ON : Error occurred  
|     |                                   | OFF : In normal operation  
|     |                                   | Flicker : Switch settings error occurred  
|     |                                   | Values other than 0 has been set to the switch 5 on the intelligent function module.                                                     |
| 3)  | External power supply terminal    | Terminal for connecting a 24VDC external power supply                                                                                        |
7.6.2 D/A conversion characteristics

(1) D/A conversion characteristics on voltage outputs
(For analog output range from -10 to 10V in a standard resolution mode)

D/A converter modules convert digital quantities that are input from a QCPU into analog values, and then output them. For example, the modules convert a digital quantity of -4000 to a analog quantity of -10V and 4000 to 10V before output. This means that the modules convert the digital input value of 1 to an analog quantity of 2.5mV, and abandon digital input values in decimal places.

Figure 7.14  D/A conversion characteristics (current output)

(2) D/A conversion characteristics on current outputs
(For analog output range from 0 to 20mA in a standard resolution mode)

For current outputs, the modules convert a digital value 0 to 0mA and 4000 to 20mA. This means that the modules convert the digital input value of 1 to an analog quantity of 5μA, and abandon digital input values in decimal places.

Figure 7.15  D/A conversion characteristics (current output)

REMARK

A voltage or current value that is equivalent to a digital value of 1 through D/A conversion (maximum resolution) differs depending on the setting of the resolution mode (1/4000, 1/12000, 1/16000) or the output range.
7.6.3 List of I/O signals and buffer memory assignment

(1) List of I/O signals
The following shows a list of the I/O signals for the D/A converter modules.
The following explanation is mentioned based on the Q68DAVN, Q68DAIN, Q68DAV and Q68DAI with 8-channel analog output (CH1 to CH8).
Note that I/O numbers (X/Y) shown in this section and thereafter are the values when the start I/O number for the D/A converter module is set to 0.

<table>
<thead>
<tr>
<th>Signal direction</th>
<th>D/A converter module → CPU module</th>
<th>Signal direction</th>
<th>CPU module → D/A converter module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device No.</td>
<td>Signal name</td>
<td>Device No.</td>
<td>Signal name 2</td>
</tr>
<tr>
<td>X0</td>
<td>Module READY</td>
<td>Y0</td>
<td>Use prohibited 1</td>
</tr>
<tr>
<td>X1</td>
<td>Use prohibited 1</td>
<td>Y1</td>
<td>CH1 Output enable/disable flag</td>
</tr>
<tr>
<td>X2</td>
<td></td>
<td>Y2</td>
<td>CH2 Output enable/disable flag</td>
</tr>
<tr>
<td>X3</td>
<td></td>
<td>Y3 2</td>
<td>CH3 Output enable/disable flag</td>
</tr>
<tr>
<td>X4</td>
<td></td>
<td>Y4 2</td>
<td>CH4 Output enable/disable flag</td>
</tr>
<tr>
<td>X5</td>
<td></td>
<td>Y5 2</td>
<td>CH5 Output enable/disable flag</td>
</tr>
<tr>
<td>X6</td>
<td></td>
<td>Y6 2</td>
<td>CH6 Output enable/disable flag</td>
</tr>
<tr>
<td>X7</td>
<td></td>
<td>Y7 2</td>
<td>CH7 Output enable/disable flag</td>
</tr>
<tr>
<td>X8</td>
<td>High resolution mode status flag</td>
<td>Y8 2</td>
<td>CH8 Output enable/disable flag</td>
</tr>
<tr>
<td>X9</td>
<td>Operating condition setting</td>
<td>Y9</td>
<td>Operating condition setting request</td>
</tr>
<tr>
<td>XA</td>
<td>Offset/gain setting mode flag</td>
<td>YA</td>
<td>User range writing request</td>
</tr>
<tr>
<td>XB</td>
<td>Channel change completed flag</td>
<td>YB</td>
<td>Channel change request</td>
</tr>
<tr>
<td>XC</td>
<td>Set value change completed flag</td>
<td>YC</td>
<td>Set value change request</td>
</tr>
<tr>
<td>XD</td>
<td>Synchronous output mode flag</td>
<td>YD</td>
<td>Synchronous output request</td>
</tr>
<tr>
<td>XE</td>
<td>Use prohibited 1</td>
<td>YE</td>
<td>Use prohibited 1</td>
</tr>
<tr>
<td>XF</td>
<td>Error flag</td>
<td>YF</td>
<td>Error clear request</td>
</tr>
</tbody>
</table>

POINT

*1: These signals cannot be used by the user since they are for system use only.
If these are turned on/off by the sequence program, the functioning of the D/A converter module cannot be guaranteed.

*2: For the Q62DAN and Q62DA, the use of Y3 to Y8 is prohibited.
For the Q64DAN and Q64DA, the use of Y5 to Y8 is prohibited.
(2) Buffer memory assignment (Q62DAN)
This section explains the assignment of the Q62DAN buffer memory.

**POINT**
Do not write data to the system areas or areas to which writing data from a sequence program is disabled. Doing so may cause malfunction.

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Default*1</th>
<th>Read/write*2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0H</td>
<td>D/A conversion enable/disable</td>
<td>3H</td>
<td>R/W</td>
</tr>
<tr>
<td>1H</td>
<td>CH1 Digital value</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>2H</td>
<td>CH2 Digital value</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>3H</td>
<td>System area</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AH</td>
<td>CH1 Set value check code</td>
<td>0</td>
<td>R</td>
</tr>
<tr>
<td>CH</td>
<td>CH2 Set value check code</td>
<td>0</td>
<td>R</td>
</tr>
<tr>
<td>DH</td>
<td>System area</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12H</td>
<td>Error code</td>
<td>0</td>
<td>R</td>
</tr>
<tr>
<td>13H</td>
<td>Setting range (CH1 to CH2)</td>
<td>0H</td>
<td>R</td>
</tr>
<tr>
<td>15H</td>
<td>System area</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>16H</td>
<td>Offset/gain setting mode Offset specification</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>17H</td>
<td>Offset/gain setting mode Gain specification</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>18H</td>
<td>Offset/gain adjustment value specification</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>19H</td>
<td>System area</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9DH</td>
<td>Mode switching setting</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>9FH</td>
<td>0</td>
<td>R/W</td>
<td></td>
</tr>
<tr>
<td>A0H</td>
<td>System area</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C7H</td>
<td>Pass data classification setting*3</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>CBH</td>
<td>System area</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CAH</td>
<td>CH1 Industrial shipment settings offset value*3</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>CBH</td>
<td>CH1 Industrial shipment settings gain value*3</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>CCH</td>
<td>CH2 Industrial shipment settings offset value*3</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>CCH</td>
<td>CH2 Industrial shipment settings gain value*3</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>CEH</td>
<td>CH1 User range settings offset value*3</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>CFH</td>
<td>CH1 User range settings gain value*3</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>D0H</td>
<td>CH2 User range settings offset value*3</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>D1H</td>
<td>CH2 User range settings gain value*3</td>
<td>0</td>
<td>R/W</td>
</tr>
</tbody>
</table>

*1: This is the initial value set after the power is turned on or the programmable controller CPU is reset.

*2: Indicates whether reading from and writing to a sequence program are enabled
R: Read enabled
W: Write enabled

*3: Areas used to restore the user range settings offset/gain values when online module change is made.
7.6.4 Adding or setting intelligent function module data

1) Click [Project] → [Intelligent Function Module] → [New Module].

2) The New Module dialog box is displayed.

3) Set the A/D converter module setting as follows.
   Module Type : Analog Module
   Module Name : Q62DAN
   Mounted Slot No. : 4
   (Specify start XY address: 0090)

4) Click the OK button.

5) The specified intelligent function module data are added to the Project window.

(To the next page)
6) Double-click Switch Setting.

7) The Switch Setting screen is displayed. Set Output range for CH1 to "0 to 5V".

8) Click the **OK** button.

9) Double-click Parameter.

10) The Parameter screen is displayed. Set "D/A conversion enable/disable setting" for CH1 to "0:Enable". (Only CH1 is used.)
11) Double-click Auto_Resh.

12) The Auto_Resh screen is displayed. Set Digital value for CH1 to “D30”.

13) Click [Project] → [Intelligent Function Module] → [Intelligent Function Module Parameter List].

(To the next page)
14) Check that "Setting Exist" is checked in Initialization (Count) and Auto Refresh (Count) for Q62DAN in the Intelligent Function Module Parameter List dialog box.

15) Click the **Close** button.
7.6.5 Exercise with the demonstration machine

(1) Sequence program
The sequence program converts values of the digital switches to analog signals.
Set the start XY to 90 and the digital value for CH1 to D30 for Q62DAN as explained before.

<table>
<thead>
<tr>
<th>Project name</th>
<th>Q62DAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

X90: Module READY signal
At power-on or reset of a programmable controller CPU, this signal turns on if D/A conversion is ready to be executed. D/A conversion is executed once this signal turned on.

Y91: CH1 Output enable/disable flag
Turning this flag on or off selects on each channel whether to output D/A converted values or offset values.
ON: D/A converted value, OFF: Offset value

(2) Operation of the demonstration machine
Stop the CPU and click on the toolbar.
The Online Data Operation dialog box is displayed. Click the **Parameter + Program** button, then click the **Execute** button to write data to the CPU. After that, activate the CPU and check the following items.

(a) Turn on X2 to enable D/A outputs of CH1.
(b) Voltage is output according to X3 to X5.
(c) The D/A OUTPUT voltmeter displays the voltage value that the D/A converter module outputs.
CHAPTER 8 SIMULATION FUNCTION

8.1 Simulation Function

The simulation function is for debugging a sequence program using the virtual programmable controller on a personal computer. The created sequence program can be immediately debugged without connecting a programmable controller CPU.

**NOTE**

Safety and handling precautions of the simulation function

1) The simulation function simulates the actual programmable controller CPU to debug a created sequence program. However, this function does not guarantee the operation of the debugged sequence program.

2) The simulation function uses the memory for simulation to input and output data to/from the I/O module and intelligent function module. Some instructions, functions, and device memories are not supported. Therefore, the operation results obtained from the virtual programmable controller may differ from those obtained from the actual programmable controller CPU.

8.2 Starting/Stopping Simulation

1) Click [Debug] → [Start/Stop Simulation].

2) The GX Simulator2 screen is displayed, and the simulation starts.

3) To stop the simulation, click [Debug] → [Start/Stop Simulation] again.
8.3 Debugging with Example Program

Use the following example for exercise.

<<Example program>>

```plaintext
0 | X0 | X1 | Y70

4 | M0 | Y71 | K9999

6 | SM412 | SM400 | C0

11 | MOV CO K4Y80

14 | END
```
8.3.1 Monitoring and testing device status

This section explains how to monitor device status, turn bit devices on/off forcibly, and change word device values.

(1) Turning bit devices on/off forcibly

In the example operation below, "X0" is forcibly turned on.

1) Click [Debug] → [Modify Value].

2) The Modify Value dialog box is displayed. Input "X0" to the "Device/Label" list box.

3) Click the ON button to forcibly turn "X0" on.

4) The result of the device being turned on is reflected on the ladder monitor screen.
(2) Changing the word device value
In the example operation below, the word device value "C0" is changed to "5".

1) Click [Debug] → [Modify Value].

2) The Modify Value dialog box is displayed. Input "C0" to the "Device/Label" list box.

3) Select the "Word[Signed]" from the "Data Type" list box.

4) Input "5" to the "Value" column.

5) After the setting is completed, click the Set button to forcibly change the current value of C0 to 5.

6) The change of the value of "C0" to "5" is reflected on the ladder monitor screen.
CHAPTER 9 MAINTENANCE

9.1 Typical Trouble

The following bar graph shows the ratio of faulty parts and causes of programmable controller errors.
[Source: Inspection made by JEMA (The Japan Electrical Manufacture's Association)]

Figure 9.1 Faulty parts on programmable controllers (multiple answers allowed)

![Bar graph showing faulty parts percentages on programmable controllers]

Figure 9.2 Causes of programmable controller faults (multiple answers allowed)

![Bar graph showing causes of faults percentages on programmable controllers]
9.2 Maintenance

To keep programmable controllers in the best operating condition, conduct the following daily inspection and periodic inspection.

(1) Daily inspection

The following table lists the items that must be inspected daily.

### Table 9.1 Daily inspection

<table>
<thead>
<tr>
<th>Item</th>
<th>Inspection item</th>
<th>Inspection contents</th>
<th>Judgment criterion</th>
<th>Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Installation of base unit</td>
<td>Check that fixing screws are not loose and the cover is not dislocated.</td>
<td>The screws and cover must be installed securely.</td>
<td>Retighten the screws.</td>
</tr>
<tr>
<td>2</td>
<td>Installation of I/O module</td>
<td>Check that the module is not dislocated and the module fixing hook is engaged securely.</td>
<td>The module fixing hook must be engaged and installed securely.</td>
<td>Securely engage the module fixing hook. Or tighten the screw.</td>
</tr>
<tr>
<td>3</td>
<td>Connection conditions</td>
<td>Check for loosening of the terminal screws.</td>
<td>Screws must not be loose.</td>
<td>Retighten the terminal screws.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Check for the distance between solderless terminals.</td>
<td>The proper distance must be provided between solderless terminals.</td>
<td>Set the proper distance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Check the connector part of the cable.</td>
<td>Connectors must not be loose.</td>
<td>Retighten the connector fixing screws.</td>
</tr>
<tr>
<td>4</td>
<td>Power supply module &quot;POWER&quot; LED</td>
<td>Check that the LED is on.</td>
<td>The LED must be on. (Error if the LED is off)</td>
<td>Refer to QCPU (Q mode) User's Manual.</td>
</tr>
<tr>
<td></td>
<td>CPU &quot;RUN&quot; LED</td>
<td>Check that the LED is on in RUN status.</td>
<td>The LED must be on. (Error if the LED is off)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU &quot;ERROR&quot; LED</td>
<td>Check that the LED is off.</td>
<td>The LED must be off. (Error if the LED is on or flashing)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU &quot;BAT.ARM&quot; LED</td>
<td>Check that the LED is off.</td>
<td>The LED must be off. (Error if the LED is on)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Module indication LED</td>
<td>Input LED Check that the LED turns on and off.</td>
<td>The LED must be on when the input power is turned on. The LED must be off when the input power is turned off. (Error if the LED does not turn on or turn off as indicated above)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output LED Check that the LED turns on and off.</td>
<td>The LED must be on when the output power is turned on. The LED must be off when the output power is turned off. (Error if the LED does not turn on or turn off as indicated above)</td>
<td></td>
</tr>
</tbody>
</table>
(2) Periodic inspection
The following table lists the items that must be inspected one or two times every half year to a year. When the equipment has been relocated or modified, or wiring layout has been changed, perform this inspection.

<table>
<thead>
<tr>
<th>Table 9.2 Periodic inspection</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item</strong></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>9</td>
</tr>
</tbody>
</table>

*1: When AnS Series Module is used in the system, the judgment criteria will be from 10 to 90% RH.

9.3 Consumable Product
Backup batteries on programmable controllers are consumable products.
9.4 Service Life of Output Relay

The output relays of the modules are consumed by the switching operation. A relay which is directly mounted on the print board of the output module is required to be replaced the output module itself after the consumption.

Figure 9.3 Life characteristics of output relay's contact (QY10, QY18A)
9.5 Spare Product

Alternative products are easily purchased through Mitsubishi service centers or local Mitsubishi representatives in Japan. Thus the alternative products can be prepared even after an accident. However, note that for foreign-related products such as exported products, alternative products must be sent beforehand. Considering the following tips at design work makes the maintenance easier.

(1) Easily replaceable type
Replacing building block-type modules is easy. Only replacing the faulty module is required.

(2) Memory type
To use standard RAMs or SRAM memory cards, backup batteries are required. The standard ROMs, Flash cards, and ATA cards do not require the battery for use, besides, these memories prevent unintentional program changes due to human-related mistakes. These memories are recommended to be employed in products for export.

(3) Reducing the number of module types
Reducing the number of module types is efficient for reducing the number of spare product types.

(4) Reserving I/O points
By not using all the I/O points on 16-, 32-, and 64-point I/O modules but reserving 10% to 20% of them, it is possible to just make changes on wiring and programs (I/O signals) instead of replacing the faulty module with a spare module when there are no spare modules.

(5) Creating a document
Since sequence programs are easily modified, the inconsistency between an operating program and documents may occur (i.e. ladder diagram, program list). Keep updating the document. To do this, using a printer is efficient.

(6) Mastering peripheral device
Mastering peripheral device such as a personal computer, GX Works2 helps the quick recovery from an accident.
Table 9.3 Spare products

<table>
<thead>
<tr>
<th>Product name</th>
<th>Quantity</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Battery</td>
<td>One or two</td>
<td>Storage lives of lithium batteries are about five years. Therefore, the stock should not be kept all the time but batteries should be purchased when required. However, keep stock of one or two for accidental situation.</td>
</tr>
<tr>
<td>2 I/O module</td>
<td>One per each module type</td>
<td>Note that I/O modules tend to be faulty during a test operation. Also note that the contacts of output modules are consumed in long-term use.</td>
</tr>
<tr>
<td>3 CPU module</td>
<td>One for each used model</td>
<td>CPU modules and memory cards are the core parts of a programmable controller, which means that an error of them result in the system down.</td>
</tr>
<tr>
<td>4 Memory card</td>
<td>One for each used model</td>
<td>Same as above. As the temperature of the power supply modules rises easily, and high ambient temperature may shorten their service lives.</td>
</tr>
<tr>
<td>5 Power supply module</td>
<td>One for each used model</td>
<td>Same as above. As the temperature of the power supply modules rises easily, and high ambient temperature may shorten their service lives.</td>
</tr>
</tbody>
</table>
9.6 Using Support Equipment

The following shows examples of support equipment in which programmable controller-used systems or devices automatically notify a detected failure or operation status to an operator or maintenance personnel during an automatic control operation.

1. **Displaying an error using a commercial lamp**

   Connect the error lamp to the output module of the programmable controller so that the lamp flashes when an error is detected.

   ![Lamp flicker diagram]

   - Lamp (Y50) flashes when an error is detected.
   - SM1 (Error detection)
   - SM412 (1-sec. clock)
   - Y50

2. **Displaying an error code on a commercial digital display**

   Connect the digital display to the output module of the programmable controller so that the error code number of the detected error is indicated on the digital display.

   ![Numerical display diagram]

   - Error code number is displayed on the digital display when an error is detected.
   - SM1 (Error detection)
   - BCD SD0 K2Y70 (Error code)
   - Error code 00
   - Error code 32

**NOTE**

The above programs cannot be executed when a stop error occurs.
3. **Displaying the contents of the detected error on the screen**

The errors details of the programmable controller can be displayed on an external CRT screen, plasma screen, and liquid crystal screen.

MELSEC-Q supports a wide variety of GOTs (Graphic Operation Terminals). In addition to the error display function, GOTs have a lot of useful functions such as the graphic monitoring, ladder monitoring, device monitoring, touch-panel switch, and printing function. (Refer to the catalogs for details.)
APPENDIX

Appendix 1  I/O Control Mode

The CPU supports two types of I/O control modes; the direct mode and refresh mode.

Appendix 1.1  Direct mode

In the direct mode, input signals are imported to a programmable controller every time they are input and treated as input information. The operation results of a program are output to the output data memory and the output modules. The following diagram shows the flow of I/O data in the direct mode.

- When the input contact instruction is executed:
  An OR operation is executed in the input information 1) from the input module and input information 2) in the data memory. Then the result is used as input information 3) at sequence program execution.
- When the output contact instruction is executed:
  Output information 4) is read from the data memory for output (Y), and a sequence program is executed.
- When the output OUT instruction is executed:
  The operation result 5) of the sequence program is output to the output module, and is stored in the data memory for output (Y).
- When the QCPU executes I/O in the direct mode, a sequence program uses DX for inputs and DY for outputs.
Appendix 1.2 Refresh mode

In the refresh mode, all changes caused in an input module are imported to the input data memory in a programmable controller CPU before every scan. The data in the data memory is used for an operation. The operation results made in a program for output (Y) are stored to the output data memory at every operation. All the data stored in the output data memory is batch-output to the output module after the execution of the END instruction. The following diagram shows the flow of I/O data in the refresh mode.

- **Input refresh**
  Input data in the input module is batch-read 1) before the execution of the step 0, and stored to the data memory for input (X).

- **Output refresh**
  Data 2) in the data memory for output (Y) is batch-output to the output module before the execution of the step 0.

- **When the input contact instruction is executed:**
  The input data is read from the data memory for input (X) 3), and a sequence program is executed.

- **When the output contact instruction is executed:**
  The output data 4) is read from the data memory for output (Y), and a sequence program is executed.

- **When the output OUT instruction is executed:**
  The operation result of the sequence program 5) is stored in the data memory for output (Y).
## Appendix 1.3 Comparisons between the direct mode and refresh mode

In the example ladder given below, turning on input X0 turns on output Y70.

<table>
<thead>
<tr>
<th>Item</th>
<th>Direct mode</th>
<th>Refresh mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Ladder example</td>
<td><img src="image1" alt="Direct mode diagram" /></td>
<td><img src="image2" alt="Refresh mode diagram" /></td>
</tr>
</tbody>
</table>

2. Response lag from when input is changed to when output is changed accordingly

<table>
<thead>
<tr>
<th></th>
<th>Program execution</th>
<th>Program execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0</td>
<td>Input instruction (LD X0)</td>
<td>Input instruction (LD X0)</td>
</tr>
<tr>
<td>Y70</td>
<td>Output instruction (OUT Y70)</td>
<td>Output instruction (OUT Y70)</td>
</tr>
</tbody>
</table>

- The delay time ranges from 0 (only execution time of the instruction) to 1 scan.
- The delay time is 0 to 1 scan.

3. Execution time of the I/O instruction

- The direct mode needs longer time than the refresh mode since a programmable controller accesses I/O modules.
- Generally, only short time is needed since a programmable controller accesses data memory.

4. Scan time

- The scan time is longer for the execution time of the I/O instructions.
- The actual scan time is the program execution time.
- The scan time is shorter for the execution time of the I/O instructions.
- The actual scan time is the total time of a program execution, input transfer, and output transfer.
Appendix 2  Special Relay

The special relay (SM) is an internal relay whose application is fixed in the programmable controller. For this reason, the special register cannot be used in the same way as other internal registers are used in sequence programs. However, the bit of the special relay can be turned on or off as needed to control the CPU module.

The following shows how to read the items in the list.
For details of special relays, refer to QCPU User's Manual Hardware Design, Maintenance and Inspection.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Indicates the special relay number.</td>
</tr>
<tr>
<td>Name</td>
<td>Indicates the special relay name.</td>
</tr>
<tr>
<td>Meaning</td>
<td>Indicates the contents of the special relay.</td>
</tr>
<tr>
<td>Explanation</td>
<td>Explains the contents of the special relay in detail.</td>
</tr>
</tbody>
</table>

**Set by**
*Indicates the setting side and setting timing of the special register.*

<table>
<thead>
<tr>
<th>(When set)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Set by&gt;</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>Set by the system</td>
</tr>
<tr>
<td>U</td>
<td>Set by user (in sequence program or test operation at a peripheral device)</td>
</tr>
<tr>
<td>S/U</td>
<td>Set by both system and user</td>
</tr>
<tr>
<td>&lt;When set&gt;</td>
<td>indicated only if setting is done by system.</td>
</tr>
<tr>
<td>Every END processing</td>
<td>Set during every END processing</td>
</tr>
<tr>
<td>Initial</td>
<td>Set during initial processing (after power-on or status change from STOP to RUN)</td>
</tr>
<tr>
<td>Status change</td>
<td>Set when the operating status is changed</td>
</tr>
<tr>
<td>Error</td>
<td>Set if an error occurs</td>
</tr>
<tr>
<td>Instruction execution</td>
<td>Set when an instruction is executed</td>
</tr>
<tr>
<td>Request</td>
<td>Set when requested by a user (using the special relay)</td>
</tr>
<tr>
<td>When system is switched</td>
<td>Set when the system is switched (between the control system and the standby system)</td>
</tr>
</tbody>
</table>

**Corresponding ACPU**
*Indicates a special relay (M9□□□) supported by the ACPU. (*M9□□□ format change* indicates the one whose application has been changed. Incompatible with the Q00J/Q00/Q01, and QnPRH.)*
*“New” indicates the one added for the Q-series CPU.*

<table>
<thead>
<tr>
<th>Corresponding CPU</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Indicates the CPU module supporting the special relay.</td>
<td></td>
</tr>
<tr>
<td>QCPU : All the Q-series CPU modules</td>
<td></td>
</tr>
<tr>
<td>Q00/Q00/Q01 : Basic model QCPU</td>
<td></td>
</tr>
<tr>
<td>Qn(H) : High Performance model QCPU</td>
<td></td>
</tr>
<tr>
<td>QnPH : Process CPU</td>
<td></td>
</tr>
<tr>
<td>QnP RH : Redundant CPU</td>
<td></td>
</tr>
<tr>
<td>QnU : Universal model QCPU</td>
<td></td>
</tr>
<tr>
<td>CPU module name</td>
<td>Only the specified CPU model (Example: Q02U)</td>
</tr>
</tbody>
</table>

For details on the following items, refer to these manuals:
*For network related items → Manuals for each network module*
*For SFC programs → MELSEC-Q/L/QnA Programming Manual (SFC)*

**POINT**
Do not change the values of special relays set by the system using a program or by test operation.
Doing so may result in a system down or communication failure.
Appendix 3  Special Register

The special register (SD) is an internal register whose application is fixed in the programmable controller. For this reason, the special register cannot be used in the same way as other internal registers are used in sequence programs. However, data can be written to the special register to control the CPU module as needed. Data is stored in binary format if not specified.

The following shows how to read the items in the list.
For details of special registers, refer to QCPU User's Manual Hardware Design, Maintenance and Inspection.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>• Indicates the special register number.</td>
</tr>
<tr>
<td>Name</td>
<td>• Indicates the special register name.</td>
</tr>
<tr>
<td>Meaning</td>
<td>• Indicates the contents of the special register.</td>
</tr>
<tr>
<td>Explanation</td>
<td>• Indicates the detailed contents of the special register.</td>
</tr>
<tr>
<td>Set by (When set)</td>
<td>• Indicates the setting side and setting timing of the special register.</td>
</tr>
<tr>
<td></td>
<td>&lt;Set by&gt;</td>
</tr>
<tr>
<td></td>
<td>S : Set by the system</td>
</tr>
<tr>
<td></td>
<td>U : Set by user (in sequence program or test operation at a peripheral</td>
</tr>
<tr>
<td></td>
<td>device)</td>
</tr>
<tr>
<td></td>
<td>S/U : Set by both system and user</td>
</tr>
<tr>
<td></td>
<td>&lt;When set&gt; indicated only if setting is done by system.</td>
</tr>
<tr>
<td></td>
<td>Every END processing : Set during every END processing</td>
</tr>
<tr>
<td></td>
<td>Initial : Set during initial processing (after power-on or status change</td>
</tr>
<tr>
<td></td>
<td>from STOP to RUN)</td>
</tr>
<tr>
<td></td>
<td>Status change : Set when the operating status is changed</td>
</tr>
<tr>
<td></td>
<td>Error : Set if an error occurs</td>
</tr>
<tr>
<td></td>
<td>Instruction execution : Set when an instruction is executed</td>
</tr>
<tr>
<td></td>
<td>Request : Set only when there is request from a user (through SM, etc.)</td>
</tr>
<tr>
<td></td>
<td>When system is switched : Set when the system is switched</td>
</tr>
<tr>
<td></td>
<td>(between the control system and the standby system)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corresponding ACPU D9□□□</th>
<th>• Indicates special register (D9□□□) supported by the ACPU.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;(D9□□□ format change&quot; indicates the one whose application has been</td>
</tr>
<tr>
<td></td>
<td>changed. Incompatible with the Q00J/Q00/Q01, and QnPRH.</td>
</tr>
<tr>
<td></td>
<td>&quot;New&quot; indicates the one added for the Q-series CPU.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corresponding CPU</th>
<th>Indicates the CPU module supporting the special relay.</th>
</tr>
</thead>
<tbody>
<tr>
<td>QCPU</td>
<td>All the Q-series CPU modules</td>
</tr>
<tr>
<td>Q00J/Q00/Q01</td>
<td>Basic model QCPU</td>
</tr>
<tr>
<td>Qn(H)</td>
<td>High Performance model QCPU</td>
</tr>
<tr>
<td>QnPH</td>
<td>Process CPU</td>
</tr>
<tr>
<td>QnPRH</td>
<td>Redundant CPU</td>
</tr>
<tr>
<td>QnU</td>
<td>Universal model QCPU</td>
</tr>
<tr>
<td>CPU module name</td>
<td>Only the specified CPU model (Example: Q02U)</td>
</tr>
</tbody>
</table>

For details on the following items, refer to these manuals:
• For network related items → Manuals for each network module
• For SFC programs → MELSEC-Q/L/QnA Programming Manual (SFC)

POINT

Do not change the values of special registers set by the system using a program or by test operation. Doing so may result in a system down or communication failure.
Appendix 4  Application Program Example

Appendix 4.1  Flip-flop ladder

(1) Y70 turns on when X0 is turned on, and turns off when X1 is turned on.

(2) When X2 is turned on, Y71 turns off if Y70 is on, and turns on if Y70 is off. This flip-flop operation is repeated.

<table>
<thead>
<tr>
<th>Project name</th>
<th>QA-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>
(3) The flip-flop operation starts when X2 is turned on. In this operation, Y70 turns on if the timer T0 is on, and Y71 turns on if the timer T1 is on. (Cycle: 10sec.)

<table>
<thead>
<tr>
<th>Project name</th>
<th>QA-17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

```
<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X2</td>
<td></td>
<td>T1</td>
<td></td>
<td></td>
<td></td>
<td>K50</td>
</tr>
<tr>
<td>7</td>
<td>T1</td>
<td></td>
<td>T0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>M0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>T0</td>
<td></td>
<td>T1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>T1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>M1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contact T0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contact T1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y71</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Appendix 4.2  One shot ladder

(1) Output starts and continues for a certain time after the input X1 is turned on.
(Time for the input being on must be longer than the set time limit.)

![Diagram 1](image1)

X1
Normally closed contact T15
Y75
Set time limit
7sec.

(2) When the input X0 is turned on momentarily, Y76 turns on for a certain time.

![Diagram 2](image2)

(3) Output starts and continues for a certain time when the input X0 is switched from on to off.

![Diagram 3](image3)

X0
Y76
Set time limit
10sec.
Pulse duration
Appendix 4.3  Long-time timer

(1) Necessary time is obtained by connecting timers in serial.

Turns on after time limit elapses

(2) Necessary time is obtained by using timers and counters.

Time limit of timer × Set value of counter = Long-time timer (note that accuracy of timers are accumulated.)

(Note) Sufficient time is obtained with the counter C7 which counts the number of time-outs of the timer T14. M56 resets T14 after time-out. With C7, the output Y73 is self-energized while count up is in progress. With Y73, T14 is reset and the following time count is stopped.
Appendix 4.4  Off delay timer

MELSEC-Q does not provide off delay timers. Configure an off delay timer as follows.

(1) The timer T6 starts operating when X5 is turned off.

(2) Turning on X5 momentarily sets the operation ready.

The timer T8 starts operating when X6 is momentarily turned on.

(Note) The above ladder operates as an off delay ladder by momentarily turning on inputs X5 and X6.
M45 is equivalent to a momentary contact of T8.
Appendix 4.5  On delay timer (momentary input)

An on delay timer of a programmable controller operates easily with a continuous input. A relay M must be used with a momentary input.

<table>
<thead>
<tr>
<th>Project name</th>
<th>QA-19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

The above ladder operates as an on delay ladder by momentarily turning on inputs X1 and X2.
Appendix 4.6  ON-OFF repeat ladder

In an ON-OFF repeat ladder, Y70 turns on when X1 is turned on, and turns off when X1 is turned on again.

Appendix 4.7  Preventing chattering input

The timer is set so that it starts output when the input keeps being on for 0.2sec.

M1 turns on when X0 keeps being on for 0.2sec. or longer. Therefore, use M1 instead of X0 when creating a program.
Appendix 4.8  Ladders with a common line

The following ladder cannot be operated as it is. To make such ladders controllable, use master control instructions (MC, MCR) in the program.

![Relay ladder diagram]

### Sequence program with master control instructions

<table>
<thead>
<tr>
<th>0</th>
<th>X0</th>
<th>X1</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>X2</td>
<td>M10</td>
</tr>
<tr>
<td>6</td>
<td>X6</td>
<td>X7</td>
</tr>
<tr>
<td>9</td>
<td>X1</td>
<td>X0</td>
</tr>
<tr>
<td>10</td>
<td>X4</td>
<td>X3</td>
</tr>
<tr>
<td>14</td>
<td>X7</td>
<td>Y79</td>
</tr>
<tr>
<td>17</td>
<td>M10</td>
<td>Y71</td>
</tr>
<tr>
<td>20</td>
<td>M20</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>M11</td>
<td>X2</td>
</tr>
<tr>
<td>25</td>
<td>M21</td>
<td></td>
</tr>
</tbody>
</table>

Note) In GX Works2, the on/off status of the master control is displayed in the title tag on the monitor screen.
Appendix 4.9  Time control program

The time value is set in the two digits of a digital switch. The currently elapsed time is displayed on Y40 to Y47 while the outputs Y70 to Y72 turn on after the set time limit has elapsed. This operation is repeated.

<table>
<thead>
<tr>
<th>Digital switch for setting time</th>
<th>Display for current time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push button for reading time</td>
<td>X3</td>
</tr>
<tr>
<td>Switch for timer</td>
<td>X4</td>
</tr>
<tr>
<td>Switch for operation</td>
<td>X5</td>
</tr>
<tr>
<td>X20 to 27</td>
<td>Y40 to 47</td>
</tr>
<tr>
<td>59 0.1sec. units</td>
<td>26 0.1sec. units</td>
</tr>
<tr>
<td>Programmable controller</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Y70 Turns on when current value is less than 2sec.</td>
</tr>
<tr>
<td></td>
<td>Y71 Turns on when current value is just 3sec.</td>
</tr>
<tr>
<td></td>
<td>Y72 Turns on when current value is 4.1sec. or more</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project name</th>
<th>QA-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X3</th>
<th>PLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>M5</td>
<td></td>
</tr>
<tr>
<td>X4</td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td></td>
</tr>
<tr>
<td>&lt;&gt; K0 D1</td>
<td></td>
</tr>
<tr>
<td>X5</td>
<td></td>
</tr>
<tr>
<td>[ &gt; K20 T3</td>
<td></td>
</tr>
<tr>
<td>[ = K30 T3</td>
<td></td>
</tr>
<tr>
<td>[ &lt; K40 T3</td>
<td></td>
</tr>
<tr>
<td>BCD T3</td>
<td></td>
</tr>
<tr>
<td>K2Y40</td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td></td>
</tr>
<tr>
<td>Y70</td>
<td></td>
</tr>
<tr>
<td>Y71</td>
<td></td>
</tr>
<tr>
<td>Y72</td>
<td></td>
</tr>
</tbody>
</table>

Reads set time 2 digits in 0.1sec. units
Starts timer
Repeats flicker
Outputs time value to exterior
Turns on when T3 is from 0.1 to 1.9 sec.
Turns on when T3 is 3.0 sec.
Turns on when T3 is 4.1 sec. or more.
Appendix 4.10  Clock ladder

The clock data such as hour, minute, and second is output to a digital display.

<table>
<thead>
<tr>
<th>Project name</th>
<th>QA-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T1</th>
<th>T0</th>
<th>K5</th>
</tr>
</thead>
<tbody>
<tr>
<td>C11</td>
<td>RST</td>
<td>C11</td>
</tr>
<tr>
<td>C12</td>
<td>RST</td>
<td>C12</td>
</tr>
<tr>
<td>C13</td>
<td>RST</td>
<td>C13</td>
</tr>
<tr>
<td>SM400</td>
<td>BCD</td>
<td>C11</td>
</tr>
<tr>
<td></td>
<td>BCD</td>
<td>C12</td>
</tr>
<tr>
<td></td>
<td>BCD</td>
<td>C13</td>
</tr>
</tbody>
</table>

LED of output module

K2Y40

K2Y48

K2Y50

47 seconds

64 hours

18 minutes
Appendix 4.10.1 Clock function (supplement)

The following ladder displays the time setting set in GX Works2 to the Q demonstration machine.

<table>
<thead>
<tr>
<th>Project name</th>
<th>QEX13</th>
</tr>
</thead>
</table>

* Clock data setting program

```
0
X7
Clock data setting command

<Setting of day and hour>
[MOVP K4X30 SD211] Day/hour status

<Setting of minute and second>
[MOVP K4X20 SD212] Minute/second status

<Clock data setting request ON>
[PLS SM210] Clock data setting request flag
```

* Clock data reading program

```
55
X7
Clock data setting command

<Clock data reading request ON>
(SM213 Clock data reading request flag

<Year (last 2 digits) and month>
[MOV SD210 K4Y60] Year (last 2 digits)/month status

<Day and hour>
[MOV SD211 K4Y50] Day/hour status

<Minute and second>
[MOV SD212 K4Y40] Minute/second status
```
<table>
<thead>
<tr>
<th>Day</th>
<th>Hour</th>
<th>Code</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday</td>
<td>118</td>
<td>H2000 SD213</td>
<td>(Y76)</td>
</tr>
<tr>
<td>Monday</td>
<td>133</td>
<td>H2001 SD213</td>
<td>(Y75)</td>
</tr>
<tr>
<td>Tuesday</td>
<td>148</td>
<td>H2002 SD213</td>
<td>(Y74)</td>
</tr>
<tr>
<td>Wednesday</td>
<td>163</td>
<td>H2003 SD213</td>
<td>(Y73)</td>
</tr>
<tr>
<td>Thursday</td>
<td>179</td>
<td>H2004 SD213</td>
<td>(Y72)</td>
</tr>
<tr>
<td>Friday</td>
<td>195</td>
<td>H2005 SD213</td>
<td>(Y71)</td>
</tr>
<tr>
<td>Saturday</td>
<td>210</td>
<td>H2006 SD213</td>
<td>(Y70)</td>
</tr>
</tbody>
</table>

[END]
Appendix 4.11  Starting \( \_ \) - \( \triangle \) operation of electrical machinery

Turning on the start switch starts the \( \_ \) operation. After the \( \_ \) operation time has elapsed, the \( \triangle \) operation mode is activated through an arc interlock state.

<table>
<thead>
<tr>
<th>Project name</th>
<th>QA-20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

![Diagram of electrical operation]
Appendix 4.12  Displaying elapsed time and outputting before time limit

The following ladder outputs the time elapsed in the timer on the LED display, and indicates that the set time limit has been reached. This system can also be applied to counters.

![Ladder diagram]
Appendix 4.13  Retentive timer

The input X2 switches between on and off continuously. The on-time of X2 is accumulated and Y72 turns on according to this accumulated value n.

(1) For a ladder that accumulates a value without a retentive timer

<table>
<thead>
<tr>
<th>Project name</th>
<th>QA-21</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

(2) When retentive timers are assigned in the device setting of the PLC parameter

Retentive timer (ST): 224 points (ST0 to ST223)
Appendix 4.14  Switching timer set value externally

(1) With an external switch, a value to be set in one timer can be selected from three patterns; 1sec., 10sec., and 100sec. A timer is activated and reset with a push button switch.

```
<table>
<thead>
<tr>
<th>Project name</th>
<th>QA-22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>
```

```
0   X0  MOV K10   D0
3   X1  MOV K100  D0
6   X2
9   X3
11  X4  SET M0
12  M0
19  T8  Y70
19  T8  Y71
```

- Set value 1sec.
- Set value 10sec.
- Set value 100sec.
- Starts timer
- Stops timer
- Turns on while timer is in operation
- Turns on by time-out
Appendix 4.15  Setting counters externally

With an external digital switch having 4 digits, a counter can be set remotely and their current values are displayed in 4 digits. In addition to every count-up, the timer outputs data when it reaches a value 100 short of the set value and a value 50 short of the set value.

Note that a setting error is indicated if the set value of the counter is less than 100.

![Diagram of setting counters externally]
<table>
<thead>
<tr>
<th>Setting</th>
<th>Reads set value</th>
<th>Outputs error when set value is 100 or less</th>
<th>Set value -100 (100 short of set value)</th>
<th>Set value -50 (50 short of set value)</th>
<th>ON during operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0</td>
<td>M0</td>
<td>BIN K4X20 D0</td>
<td>MOV D0 D1</td>
<td>MOV D0 D2</td>
<td>MOV K50 D2</td>
</tr>
<tr>
<td>Y70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y71</td>
</tr>
</tbody>
</table>

- Counter that turns on at 100 before the set value
- Counter that turns on at 50 before the set value
- Counter is reset by turning on X1
- Displays counted values to exterior
- Turns on at 100 before the set value
- Turns on at 50 before the set value
- Turns on by count up

Note) In GX Works2, the on/off status of the master control is displayed in the title tag on the monitor screen.
Appendix 4.16  Measuring operation time

Setting an operation time to a control target is useful for judging the timing of a component replacement and lubrication. The timer ST and data register D must have a backup power source so that they can continue operating at a power failure. With the contents of D31 (in one hour units) displayed externally, the program can work as an operation timer.

<table>
<thead>
<tr>
<th>Project name</th>
<th>QA-23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

- | X2 < ST250 < ST250 6-minute timer
- | = K10 D30 MOV K0 D30 1-hour timer
- | = K1 D31
- | SM400 (always ON)
- | <= K1000 D31

The management time is set to 100 hours.

Appendix 4.17  Measuring cycle time

Measuring the operation time of a control target (from its start to end) allows displaying the cycle time-out and managing a control time lag. The following ladder in which the <, >, and = instructions are used to determine the state of T200 indicates a cycle time-out and measures a time lag with the counter.

<table>
<thead>
<tr>
<th>Project name</th>
<th>QA-24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

- | X0 M56 5 In cycle
- | M56 T200 4 Measures cycle time
- | < K400 T200 10 Cycle time run out
- | < K300 T200 14 Number of cycle times of 3.01 to 4.00 sec.
- | <= K400 T200 24 Clears time out display and accumulated counts
The following explains how to obtain absolute values of negative values -32768 or smaller (to -2147483648, 32 bit data).

(Example)
Every time X1 is turned on, 999 is subtracted from a set value and the result is displayed. When the result value is negative, the output Y70 turns on, and the absolute value of the result is displayed.
Appendix 4.19  Program showing divided value of 4-digit BIN value to 4 places of decimals

(1) Example 1
The program displays the operation result using a dividend and a divisor which are individually specified in two 4-digit digital switches on two 4-digit displays (integral part and decimal part).

![Diagram of the program flow and digital display](image)

<table>
<thead>
<tr>
<th>Digital switch</th>
<th>X3F to X30</th>
<th>X2F to X20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital display</td>
<td>QCPU</td>
<td>QX 42</td>
</tr>
<tr>
<td></td>
<td>Y5F to Y50</td>
<td>Y4F to Y40</td>
</tr>
</tbody>
</table>

- **Dividend**: Digital switch X30 to X3F → D0
- **Divisor**: Digital switch X20 to X2F → D1

\[(D0) / (D1) = (D2) \ldots (D3)\]

**Quotient** Remainder

4 × (Z1) → (D10) \ HC-(D10) → (Z0)
1st time 4 × 0 → 0 \ HC-K0 → HC
2nd time 4 × 1 → 4 \ HC-K4 → H8
3rd time 4 × 2 → 8 \ HC-K8 → H4
4th time 4 × 3 → 12 \ HC-K12 → H0
(D3) × 10 → (D3)
(D3) / (D1) = (D2) \ldots (D3)

NO

INC Z1

YES

END

![Diagram showing the process of division and remainder calculation](image)
Sequence program of example 1

The FOR-NEXT instruction is executed to divide each decimal place individually and 4 decimal places are displayed in K4Y40.

<table>
<thead>
<tr>
<th>Project name</th>
<th>QA-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>X0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BINP K4X30 D0</td>
</tr>
<tr>
<td></td>
<td>BINP K4X20 D1</td>
</tr>
<tr>
<td></td>
<td>/ D0 D1 D2</td>
</tr>
<tr>
<td></td>
<td>BCDP D2 K4Y50 D3</td>
</tr>
<tr>
<td></td>
<td>DMOV PK0 Z0</td>
</tr>
<tr>
<td></td>
<td>MOV PK0 D10</td>
</tr>
<tr>
<td></td>
<td>PLS M0</td>
</tr>
<tr>
<td></td>
<td>FOR K4</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>22</th>
<th>M0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>* K4 Z1 D10</td>
</tr>
<tr>
<td></td>
<td>- H0C D10 Z0</td>
</tr>
<tr>
<td></td>
<td>* D3 K10 D3</td>
</tr>
<tr>
<td></td>
<td>/ D3 D1 D2</td>
</tr>
<tr>
<td></td>
<td>BCD D2 K1Y40D3</td>
</tr>
<tr>
<td></td>
<td>INC Z1</td>
</tr>
<tr>
<td></td>
<td>NEXT</td>
</tr>
</tbody>
</table>

Executing the **INC Z1** instruction adds one to Z1.
(2) Example 2
In example 2, D0 is divided by D1 to obtain D5 in 4 decimal places. The dividend D0 is multiplied with 10000. The result of the dividing calculation using this multiplied value is converted to a BCD value and output to an external digital display.

K4Y50
D6, integral number in 4 digits

K4Y60
D7, remainder of a decimal number

K4Y40
D5, decimal number in 4 digits

<table>
<thead>
<tr>
<th>Project name</th>
<th>QA-6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program name</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

```
X0
```

```
[0]  BINP K4X30 D0
[1]  BINP K4X20 D1
[2]  MOV K0 D2
[3]  MOV D0 K10000 D3
[4]  D/P D3 D1 D5
[5]  DBCDP D5 D5
[6]  DBCDP D7 D7
[7]  MOV D6 K4Y50
[8]  MOV D5 K4Y40
[9]  MOV D7 K4Y60
```

Integral part

Decimal part

Decimal number remainder
Appendix 4.20  Carriage line control

The following is an example of a sequence control using a carriage to convey works (materials).
Series of operations performed in one cycle is as follows; A work is set on the carriage, the carriage moves forward, the carriage stops at the forward limit, the arm pushes the work to the other conveyor side, and the carriage moves back to the backward limit.

```
<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start button (X0)</td>
<td>Operation indicator (Y70)</td>
</tr>
<tr>
<td>Switch (LS work present) (X1)</td>
<td>Carriage moves forward (Y71)</td>
</tr>
<tr>
<td>Switch (LS forward limit) (X2)</td>
<td>Carriage moves back (Y72)</td>
</tr>
<tr>
<td>Switch (LS backward limit) (X3)</td>
<td>Push (Y73)</td>
</tr>
<tr>
<td>Switch (LS open complete) (X4)</td>
<td>Push back (Y74)</td>
</tr>
</tbody>
</table>
```

Diagram: Carriage line control setup with inputs and outputs.
Project name | QA-10
---|---
Program name | MAIN

Operation indicator
- Carriage moves forward.
- Push
- Carriage moves back.
- Completion flag

Timing chart
- Start button X0
- Switch (LS work present) X1
- Switch (LS forward limit) X2
- Switch (LS backward limit) X3
- Switch (LS open complete) X4
- Operation indicator Y70
- Carriage moves forward Y71
- Carriage moves back Y72
- Push Y73
- Push back Y74

3sec
Appendix 4.21  Compressor sequential operation using ring counters

This system provides pressure control using three compressors. A pressure shortage is detected by the three pressure switches. The number of compressors operating simultaneously depends on the degree of shortage. To equal the number of usages of each compressor, compressors are activated according to the set order.

![Diagram of Compressor System Control](image-url)
Operation explanation

(1) The pressure switches (X2, X3, and X4) are initially off. In this state, turning on the start switch (X0) activates the three compressors all together, and when sufficient pressure is obtained (X2, X3, and X4 turn on), the three compressors stop. This is the basic operation of this system.

If all the compressors are at stop with sufficient pressure or the pressure shortage "Minor" is detected (X4 turns off), one compressor is activated to supply pressure until sufficient pressure is obtained.

The compressor activated at this time activates in order from A to C each time compressors are reactivated in reaction to pressure shortage.

Note that the stop switch (X1) is available for stopping compressors at any time.

(2) If one compressor does not supply sufficient pressure and the pressure shortage level goes up to "Medium" (X3 turns off), the second compressor is activated to support the first compressor. This second compressor will be compressor C if compressor A has been in operation, A if B has been in operation, and B if C has been in operation.

(3) If two compressors do not supply sufficient pressure and pressure shortage level goes up to "Major" (X2 turns off), the last compressor is also activated.

When only one compressor is in operation and pressure shortage level goes from "Minor" to "Major" directly, the rest two compressors are activated simultaneously.

(4) When two or three compressors are in operation, they continue operating together until sufficient pressure is obtained. Then they stop together when sufficient pressure is obtained (X4 turns on).

![Timing chart]
During operation

Indicates pressure status

Pressure shortage "Minor"

Pressure shortage "Medium"

Pressure shortage "Major"

Pressure shortage "Minor" is indicated when the pressure switch X4 turns off.

Pressure shortage "Medium" is indicated when the pressure switch X3 (Medium) turns off.

Pressure shortage "Major" is indicated when the pressure switch X2 (Minor) turns off.

Turns on M9 at startup

Shifts by pressure shortage "Minor"

Shifts by pressure shortage "Medium"

Shifts by pressure shortage "Major"

Resets when X1 (stop) turns on

Shift register

Returns shift to M10

Compressor A

Compressor B

Compressor C
After the basic operation, one compressor is activated in reaction to pressure shortage detected. To use the three compressors equally, they are activated according to the set order. This control is enabled by the 3-stage ring counter (ring-shaped shift registers) M10 to M12. A shift signal is generated when pressure shortage is detected (X04 switches from on to off).

![Diagram of compressor control system]
Appendix 4.22  Application example of positioning control

The following is an example of a positioning system with a pulse generator that outputs pulses per motor, brake, and unit of distance.

In this system, a command value is set with the digital switch, and this set command value is compared with the current value at start-up to determine in which direction, forward or reverse, the motor rotates. The current value in the register D16 is subtracted by 1 in forward direction, and incremented by 1 in reverse direction. Positioning is completed when the command value matches the current value. The current value is converted to a BCD value so that current position is represented in 4-digit decimal numbers.
Appendix 4.23  Application example using index Z

(1) The number of manufactured products is counted every day in one month cycle,  
and the resulting number is stored to the corresponding register of the date (D1  
to D31).

(2) The planned number of products to be manufactured is inputted with the  
external digital switch. Production stops when this number is accomplished.

(3) The date is also specified with the external digital switch.

(4) The accumulated number of products manufactured in the current month as  
well as the number of manufactured products on the current day is displayed to  
exterior.

The number of products manufactured on the current day is counted by C5.  
The accumulated number of products manufactured is counted by C6.  
The date is entered in the index Z to indirectly specify the data register  
corresponding to the date using D0Z0.  
When Z0 is 30, D0Z0 becomes 0 + 30, specifying D30.

Manufacturing results of each day ranging from 1 to 31 are stored in D1 to D31,  
which are available as production data.
Digital switch
 Writes 32760 to D35 and counts products manufactured when X20 to ZF are 0

Inputs production command

Inputs date

Y70 flashes to indicate error when date exceeding 31 is set

Specifies date indirectly

Stores number of products manufactured to data register

Displays manufacture date to exterior

Displays number of products manufactured on current day

Displays number of products manufactured in one month

Clears number of products manufactured on day anytime, if necessary

Clears all at end of month

Simultaneously transfers data 0 to D0 to D31.

Simultaneously transfers data 0 to K4Y40, K4Y50, and K4Y60.
Appendix 4.24  Application example of FIFO instruction

Manual coating work and its working time can be stored and duplicated by machinery later.

MELSEC-Q

Left (Y71) (Y70) Right → Conveyor system

Position is detected by sensors of X20 to 25

Magnified view of teaching panel

Coating bath

1) 2) 3) 4) 5) 6) Step

Cleaning (Y72) 1) 2) 3) 4) 5) 6)

Teaching panel

Current value of T1

D0 135  Write using FIFWP

Coating bath pattern

Pointer D10 6

Step 1) 11 1

2) 12 2

3) 13 8

4) 14 4

5) 15 16

6) 16 32

K2X20 32  Write using FIFWP

FIFO table

Write using FIFWP

Read using FIFRP

Reading

FIFO table

Read using FIFRP

D0 (timer constant of T0)

K2Y74 1

Write using FIFWP

Current value of T1

D0 135  Write using FIFWP

FIFO table

Write using FIFWP

Read using FIFRP

Data is backed up when X03 is turned on

Backed up data is read when X05 is turned on

Writing using FIFWP
### Operation pattern from manual to automatic operation

<table>
<thead>
<tr>
<th>Teaching panel</th>
<th>Cleaning machine</th>
<th>Coating bath</th>
</tr>
</thead>
<tbody>
<tr>
<td>X00 = Manual right moving button</td>
<td>Y70 = Conveyor, Moving right</td>
<td>X20 = Coating bath-1 (K2X20 = K1)</td>
</tr>
<tr>
<td>X01 = Manual left moving button</td>
<td>Y71 = Conveyor, Moving left</td>
<td>X21 = Coating bath-2 (K2X20 = K2)</td>
</tr>
<tr>
<td>X02 = Manual cleaning button</td>
<td>Y72 = Conveyor, Cleaning</td>
<td>X22 = Coating bath-3 (K2X20 = K4)</td>
</tr>
<tr>
<td>X03 = Recording data button</td>
<td></td>
<td>X23 = Coating bath-4 (K2X20 = K8)</td>
</tr>
<tr>
<td>X05 = Reading data button</td>
<td></td>
<td>X24 = Coating bath-5 (K2X20 = K16)</td>
</tr>
<tr>
<td>X06 = Automatic operation button</td>
<td></td>
<td>X25 = Coating bath-6 (K2X20 = K32)</td>
</tr>
<tr>
<td>X07 = Operation stop button</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y73 = Automatic operation indication LED</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Step-by-Step Explanation

1. **Start moving to right (X00 = ON)**
   - Move conveyor to the right (Y70 = ON)

2. **Stop moving to right (X00 = OFF)**
   - Stop conveyor (Y70 = OFF)

3. **Start cleaning (X02 = ON)**
   - Start cleaning (Y72 = ON)

4. **Finish cleaning (X02 = OFF)**
   - Stop cleaning (Y72 = OFF)

5. **Start moving to right (X00 = ON)**
   - Move conveyor to the right (Y70 = ON)

6. **Stop moving to right (X00 = OFF)**
   - Stop conveyor (Y70 = OFF)

7. **Start cleaning (X02 = ON)**
   - Start cleaning (Y72 = ON)

8. **Finish cleaning (X02 = OFF)**
   - Stop cleaning (Y72 = OFF)

9. **Start automatic operation (X06 = ON) & Automatic operation indication LED (Y73 = ON)**
   - Starts automatic operation

The same operation is repeated from (A).
### Project name QA-9

### Program name MAIN

<table>
<thead>
<tr>
<th>SM403</th>
<th>MOV K0 D0 K50</th>
</tr>
</thead>
<tbody>
<tr>
<td>X6</td>
<td></td>
</tr>
<tr>
<td>Y73</td>
<td></td>
</tr>
</tbody>
</table>

- Resets data to 0 only once at RUN
- Outputs to automatic operation indication LED
  (Automatic operation mode is also indicated.)

<table>
<thead>
<tr>
<th>Y73</th>
<th>T0</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>K0 D10</td>
</tr>
</tbody>
</table>

- FIFO reading pulse in automatic operation
- Finishes automatic operation
  if data is not present

<table>
<thead>
<tr>
<th>M2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>K2Y74</td>
<td>K2X20</td>
</tr>
<tr>
<td>M5</td>
<td>M3</td>
</tr>
<tr>
<td>M4</td>
<td></td>
</tr>
</tbody>
</table>

- Reads position data of coating bath
  if data is present

<table>
<thead>
<tr>
<th>M3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Y73</td>
<td></td>
</tr>
<tr>
<td>M1</td>
<td>M2</td>
</tr>
<tr>
<td>T2</td>
<td>Y73</td>
</tr>
</tbody>
</table>

- Reads cleaning time
- Reading completed flag

<table>
<thead>
<tr>
<th>M4</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>K2Y74</td>
<td>K2X20</td>
</tr>
<tr>
<td>M5</td>
<td>M4</td>
</tr>
</tbody>
</table>

- Moves conveyor to right
  since current position is on left
- Moves conveyor to left
  since current position is on right

<table>
<thead>
<tr>
<th>M5</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M3</td>
<td></td>
</tr>
<tr>
<td>Y73</td>
<td></td>
</tr>
<tr>
<td>X2</td>
<td>Y73</td>
</tr>
<tr>
<td>T2</td>
<td>D10</td>
</tr>
<tr>
<td>T1</td>
<td></td>
</tr>
</tbody>
</table>

- Completes the movement and starts cleaning (preventing chattering)
- Auto-cleaning timer

<table>
<thead>
<tr>
<th>Y70</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X0</td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>Y73</td>
</tr>
<tr>
<td>T2</td>
<td></td>
</tr>
<tr>
<td>T0</td>
<td></td>
</tr>
</tbody>
</table>

- Moves conveyor to right
- Moves conveyor to left
- In auto-cleaning
- Cleaning from conveyor

<table>
<thead>
<tr>
<th>M6</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M5</td>
<td></td>
</tr>
<tr>
<td>K2X20</td>
<td>K0</td>
</tr>
<tr>
<td>X2</td>
<td>Y73</td>
</tr>
<tr>
<td>SM403</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M6</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M3</td>
<td></td>
</tr>
<tr>
<td>Y73</td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M6</td>
<td>K3200</td>
</tr>
</tbody>
</table>

- Measures manual cleaning time
- Records manual cleaning time
- Auto-cleaning end pulse

<table>
<thead>
<tr>
<th>M7</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>K6</td>
<td>D10</td>
</tr>
<tr>
<td>M1</td>
<td></td>
</tr>
</tbody>
</table>

- Records position of coating bath
-Records cleaning time

<table>
<thead>
<tr>
<th>X3</th>
<th>Y73</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMOV</td>
<td>D10 D30 K20</td>
</tr>
</tbody>
</table>

- Saves recorded data

<table>
<thead>
<tr>
<th>X5</th>
<th>Y73</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMOV</td>
<td>D30 D10 K20</td>
</tr>
</tbody>
</table>

- Reads saved data
- Displays number of recorded data

<table>
<thead>
<tr>
<th>SM400</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BCD</td>
<td>D10 K1Y60</td>
</tr>
</tbody>
</table>

- End
Appendix 4.25  Application example of data shift

Works are conveyed along with their code numbers, and the data register of the processing machinery is analyzed to machine the work according to its code number.

A code number is stored in the data register, and M corresponding to the stored number is activated to machine the work.

A code number shifts when X2 is turned on.

A code number is input by K1X20.
During operation

Imports code number

Shifts code number

Machinery A

Machinery B

Machinery C
Appendix 4.26  Example of operation program calculating square root of data

The data stored in D5 is calculated to its square root and the result is stored in D6 and D7.

\[
\sqrt{D5} = D6 \cdot D7
\]

A value in 5th decimal digit is rounded off. Therefore, a value in 4th decimal place has error of ±1.

**REMARK**
QCPUs provide square root operation instructions for data in a real number (floating point) format.
Appendix 4.27  Example of operation program calculating n-th power of data

A value stored in D10 is calculated to its n-th power ("n" is a value stored in D14) and the result is stored in D10.

```
0 D10 D10 EFMOVP K0 K10 Clear data
0 D10 K4X30 D10 Sets data
0 D10 D10 MOV K2X20 D14
0 D10 D10
18 D10 D10 SCJ P0

P0 D10 D10 D15 FOR D14

28 D10 K10000 D16 BCD-outputs value

29 D10 D10 DBCD D16 K6Y50 in 10 digits to exterior

29 D10 D10 DBCD D18 K4Y40
```

NOTE

An operation error occurs if a value in D10 exceeds 2147483647.
Appendix 4.28  Program using digital switch to import data

When a set value of the digital switch is always input and stored to D10 of the programmable controller

In the above program, changing a value of the digital switch while the programmable controller is in RUN may cause codes other than 0 to 9 depending on the timing of the change, which may cause an operation error of the CPU. To avoid this, write a program as follows.

(Example 1) For 4 digits of X20 to X2F

(Example 2) For 8 digits of X20 to X3F
Appendix 4.29  Displaying number of faults and fault numbers using fault detection program

The following program sequentially displays the number of turned-on bit devices (such as X, M, and F) among many bit devices being used continuously, together with their device numbers.

[Application example]
When M or F is used as an output device of a fault detection program, use the following program to obtain a certain fault number from the faults.

[Sequence program flow]

(Operating procedure)

1) Searching for faulty (ON) devices
2) Displaying the number of faulty devices

Device F is used in the program example.

Display B
(Y50 to Y5F)
Display A
(Y40 to Y4F)
Display C
(Y60 to Y6F)

Condition of program
The total number of faulty circuits is set to 50.
Project name: QA-31
Program name: MAIN

Faulty circuit searches for ON devices

Specifications start number of faulty circuit (F1 to 0)
(1) Searching for ON devices

\[
\text{DSUMP K8F1 D0}
\]
\[
\text{DSUMP K8F33 D0}
\]

The total number of bits with 1 is stored in BIN.
(in this example, 16)

Turning on X2 stores the number of turned-on bits among F1 to F64 to D10 and display it.
(2) Searching for ON devices shifting 32-bit data to right

- **DMOV K8F1 D0**
- **DMOV K8F33 D0**

**DROR D0 K1**

(a) Turning on X0 sets the above shift data (D0 and D1). After that, the data is shifted right by 1 bit at each scan until a turned-on bit is detected. When a turned-on bit is detected, shifting stops in that scan (SM700 turns on), and the accumulated number of shifts (equivalent to a device number) is displayed.

(b) Each time X1 is turned on, the next turned-on bit is detected and the detected device number is displayed. At the same time, 1 is subtracted from the number of turned-on bits which have been obtained in advance to display the remaining number of turned-on bits.
### Appendix 5  Memory and File to be Handled by CPU Module

- Data to be stored in memories

The following table lists the data and drive numbers which can be stored in the program memory, standard RAM, standard ROM, and memory card.

<table>
<thead>
<tr>
<th>Item</th>
<th>CPU module built-in memory</th>
<th>Memory card (RAM)</th>
<th>Memory card (ROM)</th>
<th>File name and extension</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Program memory</td>
<td>Standard RAM</td>
<td>Standard ROM</td>
<td>SRAM card</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Drive 0¹</td>
<td>Drive 3¹</td>
<td>Drive 4¹</td>
<td>Drive 1¹</td>
<td>Drive 2¹</td>
</tr>
<tr>
<td>Parameter</td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td></td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Program</td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Device comment</td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Device initial value</td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Device data</td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>File register</td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Local device</td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Sampling trace file</td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Error history data</td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Device data storage file</td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Module error collection file</td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Backup data</td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>Programmable controller user data</td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
<tr>
<td>User setting system area</td>
<td>☐</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
<td>☒</td>
</tr>
</tbody>
</table>

*1: A drive number is used to specify a memory to be written/read by the external device using a sequence program or MC protocol. Since the memory name is used to specify the target memory in GX Works2, the drive number needs not to be considered.

*2: Store the intelligent function module parameters in the same drive with the parameters.

*3: When they are stored in different drives, the intelligent function module parameters do not become valid.

*4: To execute a program stored in the standard ROM cannot be executed.

*5: Store the program to the program memory before execution.

*6: To execute a program stored in the memory card, make the setting in the Boot File tab of the Q Parameter Setting window.

*7: The device comments cannot be read by instructions in a sequence program.

*8: Reading from a sequence program requires several scans.

*9: Only each one of file register, one local device, and sampling trace file can be stored in the standard RAM.


*11: A sequence program allows reading only. No data can be written from the sequence program.

*10: Data can be written or read with the following instructions.

- SP.FREAD (batch-reads data from the specified file in the memory card.)
- SP.FWRITE (batch-writes data to the specified file in the memory card.)

*11: Set an area used by the system.
The following tables list the memory capacities and necessity of formatting of each memory.

<table>
<thead>
<tr>
<th>Program memory</th>
<th>Q00U CPU</th>
<th>Q00UCPU</th>
<th>Q01UCPU</th>
<th>Q02UCPU</th>
<th>Q03UCPU</th>
<th>Q04UCPU</th>
<th>Formatting</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10K steps (40K byte)</td>
<td>15K steps (60K byte)</td>
<td>20K steps (80K byte)</td>
<td>30K steps (120K byte)</td>
<td>*1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard ROM</td>
<td>256K byte</td>
<td>312K byte</td>
<td>1024K byte</td>
<td>Unnecessary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard RAM</td>
<td>-</td>
<td>128K byte</td>
<td>192K byte</td>
<td>*1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory card</th>
<th>Q04UD/EJH CPU</th>
<th>Q06UD/EJH CPU</th>
<th>Q010UD/EJH CPU</th>
<th>Q13UD/EJH CPU</th>
<th>Q20UD/EJH CPU</th>
<th>Q26UD/EJH CPU</th>
<th>Formatting</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM card</td>
<td>Q2MEM-1MBS: 1M byte</td>
<td>Q2MEM-2MBS: 2M byte</td>
<td>Q2MEM-4MBS: 4M byte</td>
<td>Q3MEM-8MBS: 8M byte</td>
<td>Necessary (use GX Works2.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flash card</td>
<td>Q2MEM-2MBF: 2M byte</td>
<td>Q2MEM-4MBF: 4M byte</td>
<td>Unnecessary</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATA card</td>
<td>Q2MEM-8MBA: 8M byte</td>
<td>Q2MEM-16MBA: 16M byte</td>
<td>Q2MEM-32MBA: 32M byte</td>
<td>Necessary (use GX Works2.)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Program memory</th>
<th>Q04UD/EJH CPU</th>
<th>Q06UD/EJH CPU</th>
<th>Q010UD/EJH CPU</th>
<th>Q13UD/EJH CPU</th>
<th>Q20UD/EJH CPU</th>
<th>Q26UD/EJH CPU</th>
<th>Formatting</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>40K steps (160K byte)</td>
<td>60K steps (240K byte)</td>
<td>100K steps (400K byte)</td>
<td>130K steps (520K byte)</td>
<td>200K steps (800K byte)</td>
<td>260K steps (1040K byte)</td>
<td>*1</td>
</tr>
<tr>
<td>Standard ROM</td>
<td>1024K byte</td>
<td>2048K byte</td>
<td>4096K byte</td>
<td>Unnecessary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard RAM</td>
<td>-</td>
<td>768K byte</td>
<td>1024K byte</td>
<td>1280K byte</td>
<td>*1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*1: When the memory contents become indefinite in the initial status or due to the end of battery life, the memory is automatically formatted after the programmable controller is powered off and then on or is reset. Make sure to format the memory in GX Works2 before using.
Appendix 6. Comparison with GX Developer (changes)

(1) Supported CPU modules

The following table lists the CPU modules that are supported in GX Works2.

<table>
<thead>
<tr>
<th>Programmable controller series</th>
<th>Programmable controller type</th>
</tr>
</thead>
<tbody>
<tr>
<td>QCPU (Q mode)</td>
<td>High Performance model QCPU (Q02, Q02H, Q06H, Q12H, Q25H)</td>
</tr>
<tr>
<td></td>
<td>Universal model QCPU (Q00UJ, Q00U, Q01U, Q02U, Q03UD, Q03UDE, Q04UDH, Q04UDEH, Q06UDH, Q06UDEH, Q10UDH, Q10UDEH, Q13UDH, Q13UDEH, Q20UDH, Q20UDEH, Q26UDH, Q26UDEH, Q50UDEH, Q100UDEH)</td>
</tr>
</tbody>
</table>

The following table lists the CPU modules that are not supported in GX Works2. Use GX Developer for the following CPU modules.

<table>
<thead>
<tr>
<th>Programmable controller series</th>
<th>Programmable controller type</th>
</tr>
</thead>
<tbody>
<tr>
<td>QCPU (Q mode)</td>
<td>Basic model QCPU (Q00, Q00, Q01)</td>
</tr>
<tr>
<td></td>
<td>Process CPU (Q02PH, Q06PH, Q12PH, Q25PH)</td>
</tr>
<tr>
<td></td>
<td>Redundant CPU (Q12PRH, Q25PRH)</td>
</tr>
<tr>
<td></td>
<td>Remote I/O master (Q) 71LP21, Q) 71BR11</td>
</tr>
<tr>
<td>QCPU (motion)</td>
<td>All programmable controller types</td>
</tr>
<tr>
<td>QCPU (A mode)</td>
<td>All programmable controller types</td>
</tr>
<tr>
<td>QSCPU</td>
<td>All programmable controller types</td>
</tr>
<tr>
<td>QnACPU</td>
<td>All programmable controller types</td>
</tr>
<tr>
<td>ACPU</td>
<td>All programmable controller types</td>
</tr>
<tr>
<td>Motion controller (SCPU)</td>
<td>All programmable controller types</td>
</tr>
<tr>
<td>CNC (M6, M7)</td>
<td>All programmable controller types</td>
</tr>
</tbody>
</table>
## Unsupported features

The following table lists the features that are not supported in GX Works2. Use GX Developer, GX Simulator, or GX Configurator for the following features.

<table>
<thead>
<tr>
<th>Unsupported feature</th>
<th>Alternate S/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online function</td>
<td>TEL function</td>
</tr>
<tr>
<td>Debug function for ladder program</td>
<td>Monitor condition/Monitor stop condition setting function</td>
</tr>
<tr>
<td></td>
<td>Scan time measurement function</td>
</tr>
<tr>
<td></td>
<td>Skip/Parts/Step execution function</td>
</tr>
<tr>
<td>Debug function for ST program</td>
<td>Debug function</td>
</tr>
<tr>
<td></td>
<td>Breakpoint function</td>
</tr>
<tr>
<td>Intelligent function module programming function</td>
<td>Protocol FB support function</td>
</tr>
<tr>
<td>Intelligent function module debug function</td>
<td>Debug support function</td>
</tr>
<tr>
<td>Online function for positioning module</td>
<td>Trace function</td>
</tr>
<tr>
<td></td>
<td>System monitor function</td>
</tr>
<tr>
<td></td>
<td>Test mode function</td>
</tr>
<tr>
<td>Device initial value function</td>
<td>Device memory registration function</td>
</tr>
<tr>
<td></td>
<td>Printing function</td>
</tr>
<tr>
<td>Password function</td>
<td>Password registration function for data in project</td>
</tr>
<tr>
<td>Interaction with GX Explorer</td>
<td>Boot by GX Explorer</td>
</tr>
<tr>
<td>Interaction with PX Developer</td>
<td>Boot by PX Developer</td>
</tr>
<tr>
<td>Interaction with GX Converter</td>
<td>I/O function with GX Converter</td>
</tr>
<tr>
<td>MEDOC print format import</td>
<td>Import in MEDOC print format</td>
</tr>
<tr>
<td>Online function</td>
<td>Intelligent module diagnostics from system monitor</td>
</tr>
<tr>
<td>Sampling trace function</td>
<td>Sampling trace function conditionally on step number</td>
</tr>
</tbody>
</table>
(3) Supported project types

The following table lists the project types that are supported in GX Works2.

<table>
<thead>
<tr>
<th>Project type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple project (without labels)</td>
<td>This is the equivalent of the &quot;Do not use label&quot; project of GX Developer.</td>
</tr>
</tbody>
</table>
|                              | 1) When a project created in the "Do not use label" of GX Developer is read with GX Works2, the project becomes the Simple project (without labels).
|                              | 2) When a project created in the Simple project (without labels) of GX Works2 is read with GX Developer, the project becomes the "Do not use label" project. |
| Simple project (with labels)  | This is the equivalent of the "Use label" project of GX Developer.          |
|                              | 1) When a project created in "Use label" of GX Developer is read with GX Works2, the project becomes the Simple project (with labels).
|                              | 2) When a project created in the Simple project (with labels) of GX Works2 is read with GX Developer, the project becomes the "Use label" project. |
| Structured project           | In GX Works2, "structured programming" is available. The structured programming proceeds while creating POUs and combining them (registering tasks in the program file). |
|                              | 1) When a project created in "Use label" with ST of GX Developer is read with GX Works2, the project becomes "Structured Project" |
|                              | 2) The projects created in "Structured Project" of GX Works2 cannot be read with GX Developer. |

(a) Using project functions

Before using the project function in GX Works2, review the following precautions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description (differences between GX Developer and GX Works2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protect projects</td>
<td>By installing projects as &quot;monitoring only&quot;, the projects can be protected on each personal computer. By setting projects as &quot;read-only with the &quot;Security&quot; function, project-by-project protection is now available.</td>
</tr>
</tbody>
</table>
| Change project types         | Project types cannot be changed from "Do not use label" to "Use label". The following project type changes are now available. | 1) From "Simple project (without labels)" to "Simple project (with labels)"
|                              | 2) From "Simple project (with labels)" to "Structured Project"
|                              | * Project type cannot be changed directly from "Simple project (without labels)" to "Structured Project". |
| Read GX Developer format projects | Selecting [Project] → [Open Other Project] can read GX Developer format projects. |
| Read GX Configurator-QP format projects | Selecting [Project] → [Intelligent Function Module] → [Import GX Configurator-QP Data] can read GX Configurator-QP format projects. |
| Copy data in a project to different projects | It is enabled on the project copy dialog box. | Copy and paste is now available in the Project window. |
(4) Programming languages supported by each project type

The following table lists the programming languages that are supported by each project type of GX Works2.

<table>
<thead>
<tr>
<th>Project type</th>
<th>Supported programming language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple project (without labels)</td>
<td>Ladder, SFC (MELSAP3)</td>
</tr>
<tr>
<td>Simple project (with labels)</td>
<td>Ladder, SFC (MELSAP3)</td>
</tr>
<tr>
<td></td>
<td>* Supported program element: label, structure, function block</td>
</tr>
<tr>
<td>Structured project</td>
<td>Ladder, SFC (MELSAP3), structured ladder, ST</td>
</tr>
<tr>
<td></td>
<td>* Supported program element: label, structure, function block, function block, library</td>
</tr>
</tbody>
</table>

The following programming languages are not supported in GX Works2. Use GX Developer for the following programming languages.

<table>
<thead>
<tr>
<th>Project type</th>
<th>Supported programming language</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
<td>1) When GX Works2 reads out a program created with lists in GX Developer, it can be displayed or edited in list.</td>
</tr>
<tr>
<td></td>
<td>2) When GX Developer reads out a program created with ladder in GX Works2, it can be displayed or edited in list.</td>
</tr>
<tr>
<td>MELSAP-L</td>
<td>1) When GX Works2 reads out a program created with MELSAP-L in GX Developer, it can be displayed or edited in ladder.</td>
</tr>
<tr>
<td></td>
<td>2) When GX Developer reads out a program created with SFC (MELSAP3) in GX Works2, it can be displayed or edited in MELSAP-L.</td>
</tr>
</tbody>
</table>

(a) Using ladder language

Before using the ladder language in GX Works2, review the following precautions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description (differences between GX Developer and GX Works2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program giving devices an alias</td>
<td>It is enabled by the &quot;Alias&quot; function. Use &quot;Label&quot;.</td>
</tr>
<tr>
<td>Segment a part of program into POUs (macros)</td>
<td>It is enabled by the &quot;Macro definition/import&quot; function. Use &quot;Function Block&quot;.</td>
</tr>
<tr>
<td>Find/Replace instructions/devices/labels</td>
<td>Find is enabled by directly typing an instruction/device/label in &quot;Read mode&quot;. Pressing the Space key on the ladder editor allows the simple find.</td>
</tr>
<tr>
<td>Check use status of device/label</td>
<td>It is enabled by the &quot;Cross Reference List&quot; function and &quot;List of Used Devices&quot; functions. Select [Find/Replace] → [Cross Reference], or [Find/Replace] → [Device List].</td>
</tr>
<tr>
<td>Merge the programs</td>
<td>It is enabled by the &quot;Merge Data&quot; function. Use copy and paste on the label editor.</td>
</tr>
<tr>
<td>Verify</td>
<td>No corresponding function. The Verify Result window clearly shows the following: &quot;unmatched area of the programs&quot;, &quot;only verification source contains the program&quot; and &quot;only verification destination contains the program&quot;.</td>
</tr>
</tbody>
</table>
(b) Using SFC (MELSAP3) language
Before using the SFC (MELSAP3) language in GX Works2, review the following precautions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description (differences between GX Developer and GX Works2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GX Developer</td>
</tr>
<tr>
<td>Change block number</td>
<td>It is enabled by the &quot;copy and paste&quot; function in block list.</td>
</tr>
<tr>
<td>Auto scroll</td>
<td>A new block diagram can be opened by block start.</td>
</tr>
<tr>
<td>Open a start source block by block start</td>
<td>No corresponding function</td>
</tr>
<tr>
<td>Open operation/transition condition programs</td>
<td>Moving a cursor on the SFC diagram can display zoom (operation output/transition condition).</td>
</tr>
</tbody>
</table>

(c) Using labels
Before using labels in GX Works2, review the following precautions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description (differences between GX Developer and GX Works2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GX Developer</td>
</tr>
<tr>
<td>Check devices automatically assigned to labels</td>
<td>It is enabled by the &quot;Show assigned device&quot; function of label editor.</td>
</tr>
<tr>
<td>Import/Export device comments to labels</td>
<td>It is enabled by the &quot;device comment import&quot; function and &quot;device comment export&quot; functions.</td>
</tr>
<tr>
<td>Use pointer-type labels</td>
<td>Local pointers are assigned.</td>
</tr>
<tr>
<td>Unusable reserved words for label name</td>
<td>The definition of reserved words is different between GX Developer and GX Works2.</td>
</tr>
</tbody>
</table>
### (d) Using function blocks
Before using function blocks in GX Works2, review the following precautions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description (differences between GX Developer and GX Works2)</th>
</tr>
</thead>
</table>
| Use function blocks created with ladder | Function blocks created with ladder can be used for ladder program, ST program, and SFC program operation outputs.  
  * When using function blocks created with ladder for ST programs, select [Tool] → [Options] → [Compile] → [Basic Setting] → “Enable function block call ‘from ladder to Structured Ladder/FBD’ and ‘from Structured Ladder/FBD or ST to ladder’”. |
| Use function blocks created with structured ladder | Function blocks created with structured ladder can be used for ladder programs, structured ladder programs and ST programs. |
| Use function blocks created with ST | Function blocks created with ST can be used for ladder programs, structured ladder programs, and ST programs.  
  * When using function blocks created with ST for ladder programs, select [Tool] → [Options] → [Compile] → [Basic Setting] → “Enable function block call ‘from ladder to Structured Ladder/FBD’ and ‘from Structured Ladder/FBD or ST to ladder’”. |
| When the option “Enable function block call ‘from ladder to Structured Ladder/FBD’ and ‘from Structured Ladder/FBD or ST to ladder’” is set | When the VAR_IN_OUT input variable and output variable have different label/device, the input variable value is always equal to the output variable value. |

### (5) Using device comments
Before using device comments in GX Works2, review the following precautions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description (differences between GX Developer and GX Works2)</th>
</tr>
</thead>
</table>
| Delete comments of unused devices | It is enabled by the "Delete unused comments" function.  
  After checking the unused device by selecting [Find/Replace] → [Device List], delete the device comment directly. |
| Sample comment                    | Sample comments of the special relay/special register are provided in project format.  
  Comments of the special relay/special register and intelligent function module can be imported by the "Import from Sample Comment" function on the device comment editor. |

### (6) Using device memory
Before using the device memory in GX Works2, review the following precautions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| Device memory display             | Multiple device ranges can be displayed in a window.  
  * By selecting “All Range” when devices are input, all the device ranges can be displayed in a window in the same way as that of GX Developer. |
| Copy and paste device memory data to Excel | To copy and paste device memory data to Excel, select [Tool] → [Read from Excel File]/[Write to Excel File]. |
(7) Using device initial values
Before using device initial values in GX Works2, review the following precautions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description (differences between GX Developer and GX Works2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>GX Developer</strong></td>
</tr>
<tr>
<td>Maximum amount of device initial value data to be created</td>
<td>Only one set of data can be created.</td>
</tr>
<tr>
<td>Restriction of device number</td>
<td>The device number must be within the maximum points of each programmable controller of devices.</td>
</tr>
<tr>
<td>Write to PLC/read from PLC IC memory card write/read</td>
<td>Only 1 data can be read and written.</td>
</tr>
</tbody>
</table>

(8) Using online function
Before using the online function in GX Works2, review the following precautions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description (differences between GX Developer and GX Works2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>GX Developer</strong></td>
</tr>
<tr>
<td>Connection destination setting</td>
<td>A project can contain only one set of &quot;connection destination&quot; information.</td>
</tr>
<tr>
<td>Write/Read data to/from intelligent function modules</td>
<td>Data can be written or read to/from CPU modules and intelligent function modules simultaneously.</td>
</tr>
<tr>
<td>Write data to the Flash ROM of the CPU module</td>
<td>It is enabled by the &quot;PLC write (Flash ROM)&quot; function.</td>
</tr>
<tr>
<td>Remote operation window/PLC diagnostics window/System monitor window</td>
<td>Selecting [Online] → [Remote Operation] and [Diagnostics] → [System Monitor]/[PLC Diagnostics] can display the module image and the programmable controller CPU operation status is now easy to see. The remote operation, memory operation, and clock setup can be started from the PLC Diagnostics window.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read from PLC</td>
<td>Symbolic information in GX Developer format does not include SFC programs. Read symbolic information on &quot;Simple project (without labels)&quot;. If symbolic information of GX Developer or GX IEC Developer is read out, the project becomes uncompiled.</td>
</tr>
</tbody>
</table>
(9) Using monitor/debug function
Before using the monitor/debug function in GX Works2, review the following precautions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry device monitor</td>
<td>The &quot;entry device monitor&quot; function is now a docking window as a &quot;watch&quot; function so that it can be displayed without overlapping with the program editor. Device/label is now enabled to be entered by dragging and dropping from the program editor and the on/off status of bit devices and current values of word devices can be modified on the monitor window.</td>
</tr>
<tr>
<td>Device batch monitoring</td>
<td>The &quot;device batch monitoring&quot; and &quot;buffer memory batch monitoring&quot; functions are now integrated to realize the same operability.</td>
</tr>
<tr>
<td>Buffer batch monitoring</td>
<td>The on/off status of bit devices and current values of word devices can be modified on the monitor window.</td>
</tr>
<tr>
<td>Monitor and test intelligent function modules</td>
<td>To use the monitoring or test function to FL-net (OPC N-2) interface unit and AS-i master unit, execute the &quot;watch&quot; and &quot;Device/Buffer memory batch monitor&quot; function.</td>
</tr>
</tbody>
</table>

(10) Using printing function
Before using the printing function in GX Works2, review the following precautions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additional information print such as statement and device comment</td>
<td>The displayed image is printed or previewed.</td>
</tr>
<tr>
<td></td>
<td>To print additional information such as a statement and device comment, put the target information on the screen and then select [Project] → [Print Window]/[Print Window Preview].</td>
</tr>
</tbody>
</table>

(11) Copying saved project data
Before copying project data saved in GX Works2, review the following precautions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description (differences between GX Developer and GX Works2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy saved project data</td>
<td>Saved project data can be copied by copying files under the project name folder.</td>
</tr>
<tr>
<td></td>
<td>Copy all the workspace name folders and &quot;workspacelist.xml&quot; created in the same hierarchy as the workspace name folders.</td>
</tr>
</tbody>
</table>

(12) Compatibility with GX Developer
For the compatibility between GX Developer and GX Works2, review the following precautions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open projects in other formats</td>
<td>Before opening a GX Developer &quot;Use label&quot; project of which a program and function block have the same name, change the data name in GX Developer. Function names of ST language are different between GX Developer and GX Works2. Compile the program and correct errors.</td>
</tr>
<tr>
<td>Export projects to GX Developer format file</td>
<td>Applicable projects are the following; 1) Simple project (without labels) 2) Compiled Simple project (with labels) Projects using labels in SFC language are executed. The project can be saved in GX Developer format when none of the following is applied. 1) No device is set. 2) The length of the label name exceeds 16 characters. 3) Label name contains a device name or reserved word. 4) An invalid character is used. 5) Data type which is not supported by GX Developer is used. 6) A value which is not constant is used in the constant. Data registered to the global label is set as &quot;Auto External&quot; for all the local labels.</td>
</tr>
</tbody>
</table>
(13) Compatibility with GX IEC Developer
For the compatibility between GX IEC Developer and GX Works2, review the following precautions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Shortcut key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open projects in other formats</td>
<td>Function names of ST language are different between GX IEC Developer and GX Works2. Compile the program and correct errors.</td>
<td></td>
</tr>
<tr>
<td>User library</td>
<td>Before using GX IEC Developer user libraries which a password is set to, cancel the password in GX IEC Developer.</td>
<td></td>
</tr>
</tbody>
</table>

(14) Key operation
This section explains the differences of the key operation between GX Developer and GX Works2.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Shortcut key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read mode</td>
<td>Activates the read mode.</td>
<td>Shift + F2 (*1)</td>
</tr>
<tr>
<td>Write mode</td>
<td>Activates the write mode.</td>
<td>F2 (*1)</td>
</tr>
<tr>
<td>Cross reference</td>
<td>Displays the cross reference.</td>
<td></td>
</tr>
<tr>
<td>Device List</td>
<td>Displays the device list.</td>
<td></td>
</tr>
<tr>
<td>Convert (all programs being edited)</td>
<td>Converts all programs being edited.</td>
<td>Ctrl + Alt + F4</td>
</tr>
<tr>
<td>Project data list</td>
<td>Switches display/non-display of the project data list.</td>
<td>Alt + O</td>
</tr>
<tr>
<td>Switch between the project data list and window</td>
<td>Switches between the project data list and each window.</td>
<td>Alt + 7</td>
</tr>
<tr>
<td>Switch between ladder and list</td>
<td>Switches between the ladder window and list window.</td>
<td>Alt + F1</td>
</tr>
<tr>
<td>Monitor (all the windows)</td>
<td>Monitors ladders of all the opened programs.</td>
<td>Ctrl + F3</td>
</tr>
<tr>
<td>Monitor (write mode)</td>
<td>Activates the write mode during ladder monitoring.</td>
<td>Shift + F3 (*2)</td>
</tr>
<tr>
<td>Stop monitor (all the windows)</td>
<td>Stops the ladder monitoring for all the opened programs.</td>
<td>Ctrl + Alt + F3</td>
</tr>
<tr>
<td>Device test</td>
<td>Turns on or off the device forcibly or modifies the current value.</td>
<td>Alt + 1</td>
</tr>
<tr>
<td>Skip execution</td>
<td>Executes selected sequence programs in skip execution.</td>
<td>Alt + 2</td>
</tr>
<tr>
<td>Partial execution</td>
<td>Executes sequence programs partially.</td>
<td>Alt + 3</td>
</tr>
<tr>
<td>Step execution</td>
<td>Executes the programmable controller CPU in step execution.</td>
<td>Alt + 4</td>
</tr>
<tr>
<td>Remote operation</td>
<td>Executes remote operations.</td>
<td>Alt + 6</td>
</tr>
</tbody>
</table>

*1: In GX Works2, switching the ladder editor to the read mode/write mode is unnecessary. The ladder can be edited any time.

*2: In GX Works2, switching the ladder editor to the monitor (write mode) during the ladder monitoring is unnecessary. Even during the ladder monitoring, the ladder can be edited and written to the programmable controller in the RUN status.
Appendix 7  Customizing Shortcut Keys

Shortcut keys of each function can be customized. Customized shortcut keys can be registered as a template and utilized.

Screen display
Select [Tool] → [Key Customize].

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortcut Key</td>
<td>-</td>
</tr>
<tr>
<td>Category</td>
<td>Select a category from the group list categorized by window.</td>
</tr>
<tr>
<td>Command</td>
<td>Select a function name whose shortcut key is to be changed.</td>
</tr>
<tr>
<td>Current Key</td>
<td>Displays the shortcut key assigned to the selected command.</td>
</tr>
<tr>
<td>Press the keys to assign</td>
<td>Specify a new shortcut key to be assigned. Pressing a key(s) on the keyboard assigns the key(s). Example) Ctrl + S</td>
</tr>
<tr>
<td>Current</td>
<td>Displays the menu name to which the entered shortcut key is assigned. When the key is already assigned to another function, the function name is displayed.</td>
</tr>
<tr>
<td>Template</td>
<td>Select a template of shortcut keys from the list box.</td>
</tr>
<tr>
<td></td>
<td>• Default Setting</td>
</tr>
<tr>
<td></td>
<td>The default setting is set.</td>
</tr>
<tr>
<td></td>
<td>• GPPA Format Setting</td>
</tr>
<tr>
<td></td>
<td>The shortcut key setting at ladder programming is changed to the same setting as that at GPPA.</td>
</tr>
</tbody>
</table>
Screen button

Assign
Assigns the shortcut key. The assigned shortcut key is displayed in "Current Key".

Delete
Deletes the shortcut key selected in "Current Key".

Register Current Setting as Template...
The Enter Template Name screen is displayed.
Register the assigned shortcut keys as a template with a name.
The registered template is displayed in "Template".

Apply
The selected template of shortcut keys is applied.

Delete
Deletes a template selected in "Template".

Import...
Imports a pre-saved template file (*.gks) and adds it to "Template".

Export...
Saves a template selected in "Template" as a template file (*.gks).
In the Universal model QCPU (excludes Q00UJ CPU), expanding the index register to 32 bits enables the indexing for all the file register areas.

A method for specifying index registers for 32-bit indexing can be selected from following two methods.

- Specifying the index range used for 32-bit indexing
- Specifying the 32-bit indexing using "ZZ" specification

1) When specifying the index range used for 32-bit indexing

(a) Each index register can be set between -2147483648 and 2147483647. The following shows an example of indexing.

```
X0 DMOV K40000 Z0
Stores 40000 in Z0.
```

```
X0 MOV ZR10Z0 D0
Stores the data of ZR10Z0 = ZR{10+40000} = ZR40010 in D0.
```

(b) Specification method

For indexing with a 32-bit index register, specify the start number of an index register to be used on the Device tab of the PLC parameter setting screen in GX Works2.

**POINT**

When the start number of the index register used is changed on the Device tab of the PLC parameter setting screen, do not change the parameters only or do not write only the parameters into the programmable controller. Be sure to write the parameters into the programmable controller with the program. When the parameter is forced to be written into the programmable controller, an error of CAN'T EXE. PRG. occurs. (Error code: 2500)
(c) Device for which indexing can be used
Indexing can be used only for the devices shown below.
• ZR: Serial number access format file register
• D: Extended data register
• W: Extended link register

(d) Usable range of index registers
The following table shows the usable range of index registers for indexing with 32-bit index registers.
For indexing with 32-bit index registers, the specified index register (Zn) and the next index register of the specified register (Zn+1) are used. Be sure not to overlap index registers to be used.

<table>
<thead>
<tr>
<th>Setting value</th>
<th>Index registers to be used</th>
<th>Setting value</th>
<th>Index registers to be used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z0</td>
<td>Z0, Z1</td>
<td>Z10</td>
<td>Z10, Z11</td>
</tr>
<tr>
<td>Z1</td>
<td>Z1, Z2</td>
<td>Z11</td>
<td>Z11, Z12</td>
</tr>
<tr>
<td>Z2</td>
<td>Z2, Z3</td>
<td>Z12</td>
<td>Z12, Z13</td>
</tr>
<tr>
<td>Z3</td>
<td>Z3, Z4</td>
<td>Z13</td>
<td>Z13, Z14</td>
</tr>
<tr>
<td>Z4</td>
<td>Z4, Z5</td>
<td>Z14</td>
<td>Z14, Z15</td>
</tr>
<tr>
<td>Z5</td>
<td>Z5, Z6</td>
<td>Z15</td>
<td>Z15, Z16</td>
</tr>
<tr>
<td>Z6</td>
<td>Z6, Z7</td>
<td>Z16</td>
<td>Z16, Z17</td>
</tr>
<tr>
<td>Z7</td>
<td>Z7, Z8</td>
<td>Z17</td>
<td>Z17, Z18</td>
</tr>
<tr>
<td>Z8</td>
<td>Z8, Z9</td>
<td>Z18</td>
<td>Z18, Z19</td>
</tr>
<tr>
<td>Z9</td>
<td>Z9, Z10</td>
<td>Z19</td>
<td>Cannot be specified.</td>
</tr>
</tbody>
</table>

(e) The following shows an example of indexing and the actual process device.
(When Z0 (32-bit) is 100000 and Z2 (16-bit) is -20)

<table>
<thead>
<tr>
<th>Ladder example</th>
<th>Actual process device</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Ladder Diagram" /></td>
<td><img src="image" alt="Actual Process Diagram" /></td>
</tr>
<tr>
<td>Description</td>
<td>MOV ZR100000 D10</td>
</tr>
<tr>
<td></td>
<td>[ZR100000Z0 ... ZR(140000+100000) = ZR101000]</td>
</tr>
<tr>
<td></td>
<td>[D30Z2 ... D(30-20) = D10]</td>
</tr>
</tbody>
</table>
(2) When specifying the 32-bit indexing using “ZZ” specification

(a) One index register can specify 32-bit indexing using “ZZ” specification such as “ZR0ZZ4”.

The following shows the 32-bit indexing with "ZZ" specification.

M0 \[\text{DMOVP} \ K100000 \ Z4 \]
M0 \[\text{MOVP} \ K100 \ ZR0ZZ4 \]

Stores 100000 at Z4 and Z5.

Indexing ZR device with 32-bit index registers (Z4 and Z5)
ZR (0+100000) =ZR100000

(b) Specification method

For 32-bit indexing using “ZZ” specification, select “Use ZZ” in [Indexing Setting for ZR Device] in the Device tab in PLC parameter setting screen.

(c) Device for which indexing can be used

Indexing can be used only for the devices shown below.

- ZR: Serial number access format file register
- D: Extended data register
- W: Extended link register

(d) Usable range of index registers

The following table shows the usable range of index registers in 32-bit indexing with "ZZ" specification.

The 32-bit indexing with "ZZ" specification is specified as the format ZRmZZn.

Specifying ZRmZZn enables Zn and Zn+1 of 32-bit values to index the device number of ZRm.

<table>
<thead>
<tr>
<th>&quot;ZZ&quot; specification*</th>
<th>Index registers to be used</th>
<th>&quot;ZZ&quot; specification*</th>
<th>Index registers to be used</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐ ZZ0</td>
<td>Z0, Z1</td>
<td>☐ ZZ10</td>
<td>Z10, Z11</td>
</tr>
<tr>
<td>☐ ZZ1</td>
<td>Z1, Z2</td>
<td>☐ ZZ11</td>
<td>Z11, Z12</td>
</tr>
<tr>
<td>☐ ZZ2</td>
<td>Z2, Z3</td>
<td>☐ ZZ12</td>
<td>Z12, Z13</td>
</tr>
<tr>
<td>☐ ZZ3</td>
<td>Z3, Z4</td>
<td>☐ ZZ13</td>
<td>Z13, Z14</td>
</tr>
<tr>
<td>☐ ZZ4</td>
<td>Z4, Z5</td>
<td>☐ ZZ14</td>
<td>Z14, Z15</td>
</tr>
<tr>
<td>☐ ZZ5</td>
<td>Z5, Z6</td>
<td>☐ ZZ15</td>
<td>Z15, Z16</td>
</tr>
<tr>
<td>☐ ZZ6</td>
<td>Z6, Z7</td>
<td>☐ ZZ16</td>
<td>Z16, Z17</td>
</tr>
<tr>
<td>☐ ZZ7</td>
<td>Z7, Z8</td>
<td>☐ ZZ17</td>
<td>Z17, Z18</td>
</tr>
<tr>
<td>☐ ZZ8</td>
<td>Z8, Z9</td>
<td>☐ ZZ18</td>
<td>Z18, Z19</td>
</tr>
<tr>
<td>☐ ZZ9</td>
<td>Z9, Z10</td>
<td>☐ ZZ19</td>
<td>Cannot be specified.</td>
</tr>
</tbody>
</table>

*: ☐ indicates a device name (ZR, D, W) for indexing target
(e) The following shows an example of the 32-bit indexing with "ZZ" specification and the actual processing device.
(When Z0 (32-bit) is 100000 and Z2 (16-bit) is -20)

<table>
<thead>
<tr>
<th>Ladder example</th>
<th>Actual process device</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Ladder example diagram]</td>
<td>![Actual process device diagram]</td>
</tr>
</tbody>
</table>

(f) Available functions for "ZZ" specification
The 32-bit indexing specification with "ZZ" specification applies to the following functions of GX Works2.

<table>
<thead>
<tr>
<th>No.</th>
<th>Function name and description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Specifying devices in program instruction</td>
</tr>
<tr>
<td>2</td>
<td>Entry device monitor</td>
</tr>
<tr>
<td>3</td>
<td>Device test</td>
</tr>
<tr>
<td>4</td>
<td>Device test with conditions</td>
</tr>
<tr>
<td>5</td>
<td>Monitor condition setting</td>
</tr>
<tr>
<td>6</td>
<td>Sampling trace (Trace point (specifying devices), trace target device)</td>
</tr>
</tbody>
</table>

**POINT**
ZZn cannot be used alone as a device like "DMOV K100000 ZZ0". When setting values of index registers to specify 32-bit indexing with "ZZ" specification, set the value of Zn (Z0 to Z19). ZZn alone cannot be input to each function of GX Works2.

Appendix 9  FB

Appendix 9.1  FB

FB is an abbreviation for a Function Block that is designed to convert a ladder block, which is used repeatedly in a sequence program, into a component (FB) to be utilized in a sequence program. This not only increases the efficiency of program development but also reduces programming mistakes to improve program quality.

Figure App. 9.1 Converting a sequence program into a component

Appendix 9.1.1 Conversion into components

The following section explains the process to convert a simple program into a component.

1) Program to be converted into a component

2) Divide into input and output. In addition, replace the internal device with an internal label.

3) When changed to an FB

4) Pasting the FB to a program

Figure App. 9.2 Flow of conversion into components
Appendix 9.1.2 Advantages of using FBs

This section introduces advantages of creating programs by using FBs.

(1) Easy programming
A sequence program can be created simply by pasting FBs. This significantly reduces the program development man-hours. (FB libraries provided by Mitsubishi Electric Corporation makes programming easier.)

(2) Easy reading
Using an FB creates a simple program with only a "box" (FB), an input, and an output to create an easy-to-read sequence program.
(3) Reusing
Converting a standard program into a component allows the program to be reused any number of times.
As a result, operations such as copying a sequence program and modifying a device, which have often been required in the past, will be unnecessary.

(4) Improving quality
Converting a standard program into a component as an FB to reuse the program allows development of programs of consistent quality, without relying on the technological skill of the program developers.
When developers A and B are developing sequence programs for different devices, using the same FB for the common processing enables the developers to create consistent quality of sequence programs.
(5) Protecting assets
By setting up a block password, the created FB can be protected so that it cannot be viewed.

Appendix 9.1.3 FB Libraries

An FB library is a collection of FBs that are usable in GX Works2 (Simple project). Using an FB library enables easy setting and operation of MELSEC-Q/L modules and partner products.

Example of MELSEC-Q/L module

FBs for partner products
Vision sensor FB RFID FB Laser displacement sensor FB ...

CC-Link Ethernet
Vision sensor RFID Laser displacement sensor

Partner product family
(1) FB library lineup
FB libraries include "FBs for MELSEC-Q/L modules" and "FBs for partner products".

(2) How to obtain FB libraries
FB libraries can be obtained from Mitsubishi Electric FA site.

URL  http://www.mitsubishielectric.co.jp/fa/index.html

For the procedure to obtain the FB libraries, refer to App. 9.2.2 "Preparations prior to use of FB libraries".
Appendix 9.1.4  Development tool

GX Works2 (Simple project) ver 1.12N or later is required to develop sequence programs using FBs.

**POINT**
Depending on the FB library, supporting versions of GX Works2 may differ. For details, refer to the download page of each FB.

Appendix 9.1.5  FB specifications and precautions

The following specifications and precautions must be understood prior to using FBs.

1. An FB cannot be used in another FB.
2. Because an FB specific process is added when an FB is arranged, the number of steps increases when compared to a ladder created without an FB.
3. FBs cannot be used in an interruption program.
4. FBs whose execution does not complete within a scan cannot be used in the FOR to NEXT instruction loops or subroutine programs.
Appendix 9.2 Creating a program by using an FB library

This section explains the procedure to create a program by using an FB library.

Appendix 9.2.1 Programs to be created

This section explains how to use an FB library with an example of importing an analog value from an analog input module.

Example) Reading an analog value to D10 from the analog input module (Q64AD) when the switch (X2) is turned on.

The program can easily be created by using an FB library as follows.

The FB created by a user is also available other than the FB in the FB library. For the creation method of a new FB, refer to "MELSOFT GX Works2 FB Quick Start Guide".
Appendix 9.2.2  Preparations prior to use of FB libraries

Before using an FB library, contact your distributor to obtain it. (FB libraries will not be installed when installing GX Works2.)

The following explains operation procedures using the FB library for Q64AD as an example.

1) As the file obtained from your distributor is a zip format file, unzip "q64ad_v100a.zip".

2) Double-click "setup.exe" in "q64ad_v100a".

3) The screen for installation is displayed. Follow the instructions to complete the installation.

4) The following dialog is displayed when the installation is completed. Click the OK button to close the dialog.

This completes the preparation prior to use of FB libraries.
Appendix 9.2.3  Importing an FB library to projects

This section explains how to import an FB library for analog input module (Q64AD) to be pasted to the program into a project.

Create a new project before the following operation. (refer to section 2.3.2)

1) Click [Project] → [Library] → [Install].

2) The Install dialog box is displayed.

3) Select “Q64AD” from Library List.

4) Click the Refresh FB List button.

5) Check the library to import.

6) Click the OK button.

7) The imported FBS are displayed under FB_Pool in the Project view and displayed in the Selection window.
Appendix 9.2.4  Pasting FBs

Drag and drop FBs to be pasted to the program window from the Project view or Selection window. (Drag and drop from the Project view is possible from GX Works2 1.24A or later.)

Operating Procedure

1) Paste “M+Q64AD_ReadADVal” to the program window.

2) The Input FB Instance Name dialog box is displayed.

For details of settings, refer to the next page.
Appendix 9.2.5  Setting names of the pasted FBs

When an FB library is pasted to the program window, a dialog to input a name of the pasted FB (FB instance name) is displayed.

Instance name is a name to distinguish the FB.

A temporary name is automatically set to the instance name. To use the name as it is, close the dialog by clicking **OK**.

Make sure that the same name does not exist in the same program when changing the name.

In this section, the default is used.

**Operating Procedure**

1) Enter the FB instance name ("ReadADVal_1" in the example) and click the **OK** button.

2) The FB is pasted to the program window.

**POINT**

When entering an instance name, note the following points.

- Case-sensitive
- A number cannot be set for the first letter.
- The maximum number of characters for an instance name is 16.

An error occurs if **OK** is clicked with the following setting.

(When the first letter is a number).
Appendix 9.2.6  Creating input and output ladders

Create the input ladder section and the output ladder section of the pasted FB to complete the program. Refer to the following figure and enter the information.

Appendix 9.2.7  Performing conversion/compilation

Conversion/compilation is required to execute the completed program. The following explains how to convert/compile all programs.

1) Click [Compile] → [Rebuild All].

2) The message on the left is displayed. Click the **Yes** button.

3) The compilation result is displayed in the Output window.
Appendix 9.2.8  Writing sequence programs

For the procedure to write sequence programs, refer to section 2.7 (1) "Writing data to the CPU".

Appendix 9.2.9  Operation check

For the procedure to check the operation of the created program, refer to section 2.8 Monitoring Ladder Program Status.

Turn on the switch (X2) and confirm that the analog value is read.

Double-clicking the FB in the sequence program on the screen enables monitoring of the sequence program status in the FB.

* FB name: H064AD_ReadADVal
* Function: AD conversion data read
* Version: Ver1.01B

Turn on the switch (X10).

The current analog value is displayed.