MITSUBISHI

# Type SW0D5C-ACT-E ActiveX Communication Support Tool

# Programming Manual

MELSEC

Mitsubishi Programmable Logic Controller

# • SAFETY PRECAUTIONS •

(Always read these instructions before using this equipment.)

Before using this product, please read this manual and the relevant manuals introduced in this manual carefully and pay full attention to safety to handle the product correctly.
The instructions given in this manual are concerned with this product. For the safety instructions of the programmable controller system, please read the CPU module user's manual.
In this manual, the safety instructions are ranked as "DANGER" and "CAUTION".

| | |
|---|---|
| ⚠ DANGER | Indicates that incorrect handling may cause hazardous conditions, resulting in death or severe injury. |
| ⚠ CAUTION | Indicates that incorrect handling may cause hazardous conditions, resulting in medium or slight personal injury or physical damage. |

Note that the ⚠CAUTION level may lead to a serious consequence according to the circumstances. Always follow the instructions of both levels because they are important to personal safety.

Please save this manual to make it accessible when required and always forward it to the end user.

## [Design Instructions]

### ⚠ DANGER

- When performing data changes or status control from the peripheral device to the running PLC, configure up an interlock circuit outside the PLC system to ensure that the whole system will operate safely.
  In addition, predetermine corrective actions for the system so that you can take measures against any communication error caused by a cable connection fault or the like in online operations performed from the peripheral device to the PLC.

### ⚠ CAUTION

- Read the manual carefully before performing the online operations (especially forced output and operating status change) which will be executed with the peripheral device connected to the running CPU module.
  Not doing so can damage the machine or cause an accident due to misoperation.

REVISIONS

* The manual number is given on the bottom left of the back cover.

| Print Date | * Manual Number | Revision |
|---|---|---|
| Apr., 2000 | SH (NA)-080078-A | First edition |

Japanese Manual Version SH-080080-A

Operating Instructions

(1) About Ethernet communication

(a) When access is made to the QnACPU, AnUCPU, QCPU (A mode) or motion controller CPU via the E71, the device range is equivalent to that of the AnACPU.

(b) When making access to the PLC CPU through Ethernet communication, the functions may not be executed depending on the PLC CPU status.

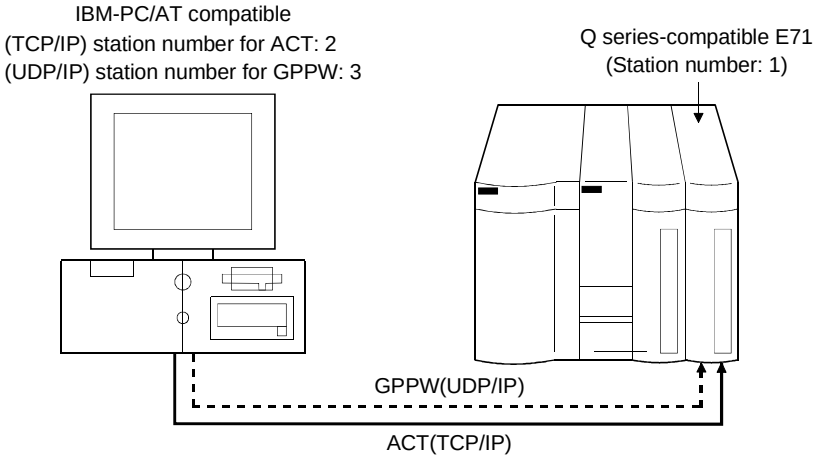
1) When the protocol is TCP/IP (target module: E71, QE71)
The functions can be executed only when the communication target PLC CPU is in the RUN mode.
An error is returned if the PLC CPU is in other than the RUN mode.

2) When the protocol is UDP/IP (target module: E71, QE71)
The functions cannot be executed until the communication target PLC CPU is RUN once.
An error is returned if the PLC CPU has not been RUN once.

(c) The communication line is broken if the CPU becomes faulty or the Ethernet module is reset during Ethernet communication (when the protocol is TCP/IP). In that case, perform line close processing (Close) and then execute reopen processing (Open).

(d) When two different communication systems (protocols) are used to make access from one IBM-PC/AT compatible to one Q series-compatible E71, two station numbers, i.e. for TCP/IP and for UDP/IP, must be set.

(Example) When ACT uses TCP/IP and GPPW uses UDP/IP




Set different station numbers as the (TCP/IP) station number for ACT and (UDP/IP) station number for GPPW. If they are set to the same station number, an error will occur on the Ethernet module side.

(2) About target existence check starting interval*1 of Ethernet module

If close processing (Close) is executed from the IBM-PC/AT compatible, the Ethernet module may not perform close processing (Close).
One of its causes is the open cable.
If open processing (Open) is executed from the IBM-PC/AT compatible with the Ethernet module not performing close processing (Close), open processing (Open) from the IBM-PC/AT compatible is not terminated normally until the Ethernet module makes a target existence check and executes close processing (Close).
If you want to terminate open processing (Open) early from the IBM-PC/AT compatible, shorten the target existence check starting interval setting of the Ethernet module. (The target existence check starting interval setting of the Ethernet module defaults to 10 minutes.)
*1: It can be set for the E71 of AJ71E71-S3 or later.

(3) About switch settings of E71 and QE71

If the 4 lower digits of the error code which occurred during Ethernet communication using E71 or QE71 is not indicated in the E71 or QE71 manual, check whether the DIP switch (SW2) of the E71 or QE71 is set as indicated below.
If the DIP switch is not set correctly, a difference has occurred in the packet format (ASCII/binary) and therefore the error code returned from the module cannot be recognized correctly.

| Communication | | SW2 Switch Setting |
|---|---|---|
| E71 | TCP/IP | ON (ASCII mode) |
| | UDP/IP | OFF (binary mode) |
| QE71(TCP/IP) | | ON (ASCII mode) |

(4) About computer link communication

(a) If the connected station CPU is the AnUCPU and the computer link module is the UC24 for computer link connection, remote operation will result in an error when access is made to the AnNCPU, AnACPU or QnACPU via the MELSECNET/10.

(b) On the UC24 and C24 computer link modules, remote "PAUSE" operation will result in an error for all connections.

(c) For the QC24, note that specifying the first I/O number of a nonexisting module and reading/writing U**\G** will not return an error for the module whose software version is "k" or earlier.

(d) In any connection form where the target station of the UC24 or C24 is the QnA, an error is returned if SetClockData or GetClockData is executed.

(5) Instructions for relaying the MELSECNET(II)

When access is made to the QnACPU, AnUCPU, QCPU (A mode) or motion controller CPU via the MELSECNET(II), the device range is equivalent to that of the AnACPU.

(6) Restrictions on use of the FXCPU

(a) When the FXCPU is used, access to the TN devices (timer present values) or CN devices (counter present values) is not permitted if the device numbers specified are split across 199 or earlier and 200 or later.

(b) As the FXCPU does not have a PAUSE switch as the PLC CPU, an error is returned if remote pause is specified in SetCpuStatus.

(c) Note that specifying the first I/O number of a nonexisting module and executing the WriteBuffer( ) method will not return an error.

(d) For the index registers (Z, V) of the FXCPU, data cannot be written to 2 or more consecutive points using WriteDeviceBlock(). (Data may be written to only one point.)

(7) CheckDeviceString

Do not use the CheckDeviceString method of each ACT control.

(8) About ActUMsg control, ActUWzd control, ActMnet2BD control and ActAFBD control

Installing ACT registers the ActUMsg control, ActUWzd control, ActMnet2BD control and ActAFBD control, but do not use them.

(9) Precautions for use of ActQJ71E71TCP, ActAJ71QE71TCP and ActAJ71E71TCP controls

(a) Provide an interval longer than the sequence scan time of the Ethernet module loaded station from when the Open method is executed until the Close method is executed.

(b) Provide an interval of at least 500ms from when the Close method is executed until the Open method is executed again.

(10) Precautions for ladder logic test communication

When running a user program, make sure that the ladder logic test function (LLT) and GPPW have started.

In addition, do not terminate the ladder logic test function (LLT) and GPPW while the user program is running.

If you do so, you will not be able to terminate the user program normally.

INTRODUCTION

Thank you for purchasing the Type SW0D5C-ACT-E ActiveX Communication Support Tool.
Read this manual and make sure you understand the functions and performance of Type SW0D5C-ACT-E ActiveX Communication Support Tool thoroughly in advance to ensure correct use.
Please make this manual available to the end user.

CONTENTS

About Manuals

The following lists the manuals for this software package.
Refer to the following table when ordering manuals.

Related Manuals

| Manual Name | Manual Number (Model Code) |
|---|---|
| Type SW0D5C-ACT-E ActiveX Communication Support Tool Operating Manual (Startup)<br>Provides procedures for installing and uninstalling SW0D5C-ACT-E and for browsing the operating manual. (Packed with the product) | IB-0800112 (13J982) |
| Type SW0D5C-ACT-E ActiveX Communication Support Tool Operating Manual<br>Gives how to perform setting and operation of each utility on SW0D5C-ACT-E. (Optionally available) | SH-080077 (13J981) |
| Type A70BDE-J71QLP23/A70BDE-J71QLP23GE/A70BDE-J71QBR13/A70BDE-J71QLR23 MELSECNET/10 Interface Board User's Manual(For SW3DNF-MNET10)<br>Describes the features, specifications, part names and setting of the MELSECNET/10 board, and the installation, uninstallation and others of the driver. (Packed with the product) | IB-0800035 (13JL93) |
| Type A80BDE-J61BT11 CC-Link System Master/Local Interface Board User's Manual (For SW3DNF-CCLINK)<br>Describes the features, specifications, part names and setting of the CC-Link master board, and the installation, uninstallation and others of the driver. (Packed with the product) | IB-0800110 (13JR14) |
| Type A80BDE-J61BT13 CC-Link Interface Board User's Manual (For SW3DNF-CCLINK)<br>Describes the features, specifications, part names and setting of the CC-Link local board, and the installation, uninstallation and others of the driver. (Packed with the product) | IB-0800036 (13JL94) |
| Type A80BDE-A2USH-S1 PLC CPU Board User's Manual (For SW0DNF-ANU-B)<br>Describes the features, specifications, part names and setting of the CPU board, and the installation, uninstallation and others of the driver. (Packed with the product) | IB-0800087 (13JR08) |

Note: Type SW0D5C-ACT-E ActiveX Communication Support Tool Operating Manual is contained in the CD-ROM together with the software package as a set.
When you want to purchase the manual alone, it is optionally available as the printed matter of the manual number (Model code) in the above table.

How to Use This Manual

"How to Use This Manual" is given purpose-by-purpose for use of ACT.
Refer to the following outlines and use this manual.

(1) To know the feature and ACT control lists (Chapter 1)
Chapter 1 gives the ACT control outline and ACT control lists.

(2) To use the ACT controls on Visual Basic or Visual C++ (Section 2.1)
Section 2.1 provides how to make settings on Visual Basic and Visual C++ to use the ACT controls.

(3) To know the programming procedure (Section 2.2)
Section 2.2 contains programming procedures.

(4) To know the device types to be specified in the functions (Section 2.3)
Section 2.3 lists the device types.

(5) To know the details of the ACT controls (Chapter 3)
Chapter 3 provides the details of the ACT controls.
Read this chapter when creating a program.

(6) To know the details of the functions (Chapter 4)
Chapter 4 gives the details of the functions.
Read this chapter when creating a program.

(7) To know how to use the sample programs (Chapter 5)
Chapter 5 provides the sample programs and how to use them.
Use them as reference when creating a program.

(8) To know the definitions of the error codes (Chapter 6)
Chapter 6 lists the error codes returned by the ACT controls and the error codes returned by the CPUs, modules and network boards.

(9) To know the accessible devices and ranges
The ACT operating manual contains the accessible devices and ranges.
Refer to the ACT operating manual.

Abbreviations and Terms in This Manual

Unless otherwise specified, the following generic terms and abbreviations are used in this manual to describe Type SW0D5C-ACT-E ActiveX Communication Support Tool.

| Generic Term/Abbreviation | Description |
|---|---|
| ACT | Abbreviation of Type SW0D5C-ACT-E ActiveX Communication Support Tool |
| Windows NT 4.0 | Abbreviation of Microsoft Windows NT Workstation 4.0 (English version) |
| Windows 95 | Abbreviation of Microsoft Windows 95 (English version) |
| Windows 98 | Abbreviation of Microsoft Windows 98 (English version) |
| Windows | Generic term of Windows 95, Windows 98 and Windows NT Workstation 4.0 |
| VB | Abbreviation of Microsoft Visual Basic 6.0 (English version) |
| VC++ | Abbreviation of Microsoft Visual C++ 6.0 (English version) |
| IBM-PC/AT compatible | Abbreviation of the IBM PC/AT or its compatible personal computer |
| GPPW | Abbreviation of Type SW□D5C-GPPW-E/SW□D5F-GPPW-E GPP function software package |
| Ladder logic test function (LLT) | Abbreviation of Type SW□D5C-LLT-E/SW□D5F-LLT-E ladder logic test tool function software package |
| MELSECNET/10 board | Abbreviation of Type A70BDE-J71QLP23/A70BDE-J71QLP23GE/A70BDE-J71QBR13/A70BDE-J71QLR23 MELSECNET/10 interface board |
| CC-Link board | Abbreviation of Type A80BDE-J61BT11 CC-Link system master/local interface board and Type A80BDE-J61BT13 CC-Link interface board |
| CPU board | Abbreviation of Type A80BDE-A2USH-S1 PLC CPU board |
| AnNCPU | Generic term of the A0J2HCPU, A1SCPU, A1SCPU-S1, A1SCPUC24-R2, A1SHCPU, A1SJCPU, A1SJHCPU, A1NCPU, A2CCPU, A2CCPUC24, A2CCPUC24-PRF, A2CJCPU, A2NCPU, A2NCPU-S1, A2SCPU, A2SCPU-S1, A2SHCPU, A2SHCPU-S1, A3NCPU and A1FXCPU |
| AnACPU | Generic term of the A2ACPU, A2ACPU-S1, A2ACPUP21/R21, A2ACPUP21-S1, A3ACPU and A3ACPUP21/R21 |
| AnUCPU | Generic term of the A2UCPU, A2UCPU-S1, A2USCPU, A2USCPU-S1, A2ASCPU, A2ASCPU-S1, A2ASCPU-S30, A2USHCPU-S1, A3UCPU and A4UCPU |
| QnACPU | Generic term of the Q2ACPU, Q2ACPU-S1, Q2ASCPU, Q2ASCPU-S1, Q2ASHCPU, Q2ASHCPU-S1, Q3ACPU, Q4ACPU and Q4ARCPU |
| ACPU | Generic term of the AnNCPU, AnACPU and AnUCPU |
| QCPU (A mode) | Generic term of the Q02CPU-A, Q02HCPU-A and Q06HCPU-A |
| QCPU (Q mode) | Generic term of the Q02CPU, Q02HCPU, Q06HCPU, Q12HCPU and Q25HCPU |
| QCPU | Generic term of the QCPU (Q mode) and QCPU (A mode) |
| FXCPU | Generic term of the FX0, FX0S, FX0N, FX1, FX1N, FX1S, FX2, FX2C, FX2N and FX2NC series |
| Motion controller CPU | Generic term of the A171SHCPU, A172SHCPU, A173UHCPU, A173UHCPU-S1, A273UHCPU and A273UHCPU-S3 |
| PLC CPU | Generic term of the QCPU, QnACPU, ACPU, FXCPU and motion controller CPU |
| C24 | Generic term of the A1SCPUC24-R2, A1SJ71C24-PRF, A1SJ71C24-R2, A1SJ71C24-R4, A2CCPUC24, A2CCPUC24-PRF, AJ71C24-S6 and AJ71C24-S8 |
| UC24 | Generic term of the AJ71UC24, A1SJ71UC24-R2, A1SJ71UC24-R4 and A1SJ71UC24-PRF |
| QC24 | Generic term of the AJ71QC24, AJ71QC24-R2, AJ71QC24-R4, A1SJ71QC24-R2 and A1SJ71QC24-R2 |
| QC24N | Generic term of the AJ71QC24N, AJ71QC24N-R2, AJ71QC24N-R4, A1SJ71QC24N and A1SJ71QC24N-R2 |
| QC24(N) | Generic term of the QC24 and QC24N |
| Q series-compatible C24 | Generic term of the QJ71C24 and QJ71C24-R2 |

| Generic Term/Abbreviation | Description |
|---|---|
| Computer link module | Generic term of the C24, UC24, QC24(N) and Q series-compatible C24 |
| E71 | Generic term of the AJ71E71, AJ71E71-S3, A1SJ71E71-B2, A1SJ71E71-B5, A1SJ71E71-B2-S3 and A1SJ71E71-B5-S3 |
| QE71 | Generic term of the AJ71QE71, AJ71QE71-B5, A1SJ71QE71-B2 and A1SJ71QE71-B5 |
| Q series-compatible E71 | Generic term of the QJ71E71 and QJ71E71-B2 |
| Ethernet module | Generic term of the E71, QE71 and Q series-compatible E71 |
| CC-Link G4 module | Abbreviation of Type AJ65BT-G4 GPP function peripheral connection module |
| Computer link communication | Abbreviation of communication made with the PLC CPU using the computer link module |
| Ethernet communication | Abbreviation of communication made with the PLC CPU using the Ethernet module |
| CPU COM communication | Abbreviation of communication made by connecting the IBM-PC/AT compatible to the RS-232C or RS-422 connector of the PLC CPU |
| CPU USB communication | Abbreviation of communication made by connecting the IBM-PC/AT compatible to the USB connector of the QCPU (Q mode) |
| MELSECNET/10 communication | Abbreviation of communication made with the PLC CPU using the MELSECNET/10 board |
| CC-Link communication | Abbreviation of communication made with the PLC CPU using the CC-Link board |
| CC-Link G4 communication | Abbreviation of communication made with the PLC CPU using the CC-Link G4 module |
| CPU board communication | Abbreviation of communication made with the PLC CPU using the CPU board |
| Ladder logic test communication | Abbreviation of communication made with the ladder logic test function (LLT) |
| Utility setting type | Abbreviation of user program creation using the communication settings utility |
| Program setting type | Abbreviation of user program creation without using the communication settings utility |
| ACT controls | Generic term of the ActiveX controls offered by ACT |

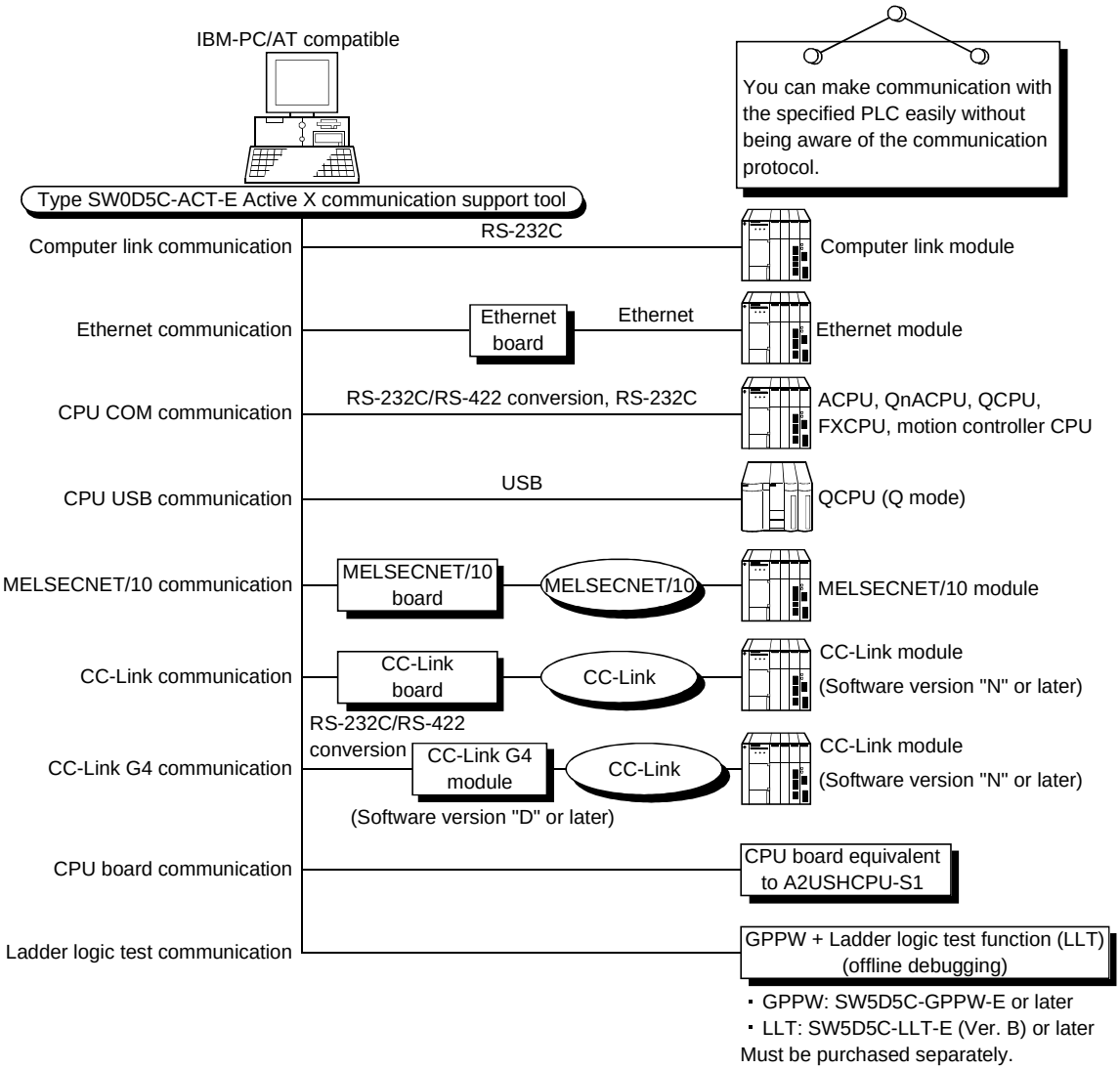
# 1 OVERVIEW



This chapter provides the function outline of the ACT controls offered by Type SW0D5C-ACT-E Active X communication support tool.

## 1.1 Outline of ACT Controls

These controls are used to create user programs for communication with a PLC CPU. This enables the user to make communication without being aware of the hardware and communication protocol on the other end.



## 1.2 ACT Control and Function Lists

The following sections give the lists of ACT controls and functions.

### 1.2.1 ACT control list

The following table lists the ACT controls included in each DLL offered by ACT.

| DLL Name | Included Control Name | Application |
|---|---|---|
| ActMulti.DLL | ActEasyIF | Used to make communication settings easily on the communication settings utility to make communication. |
| ActPcCom.DLL | ActQCPUQ | Used to make communication via the serial port of the corresponding PLC CPU. |
| | ActQCPUA | |
| | ActQnACPU | |
| | ActACPU | |
| | ActFXCPU | |
| ActComLk.DLL | ActQJ71C24 | Used to make communication via the computer link module (serial communication module). |
| | ActAJ71QC24 | |
| | ActAJ71UC24 | |
| | ActAJ71C24 | |
| ActEther.DLL | ActQJ71E71TCP | Used to make communication via the Ethernet module. |
| | ActQJ71E71UDP | |
| | ActAJ71QE71TCP | |
| | ActAJ71QE71UDP | |
| | ActAJ71E71TCP | |
| | ActAJ71E71UDP | |
| ActPcUsb.DLL | ActQCPUQUSB | Used to make communication via the USB port of the PLC CPU. |
| ActCcG4.DLL | ActCCG4QnA | Used to make communication via the CC-Link G4 module. |
| | ActCCG4A | |
| ActBoard.DLL | ActMnet10BD | Used to make communication with or via the network board. |
| | ActCCBD | |
| | ActAnUBD | |
| ActLlt.DLL | ActLLT | Used to make communication with the ladder logic test function (LLT). |

## 1.2.2 Function list

The following table lists the features of the functions and the functions available for the ACT controls.

### (1) Function list

Refer to "CHAPTER 4 FUNCTIONS" for full information on the functions.

| Function Name | Feature |
|---|---|
| Open | Opens a communication line. |
| Close | Closes a communication line. |
| ReadDeviceBlock | Batch-reads data from devices. |
| WriteDeviceBlock | Batch-writes data to devices. |
| ReadDeviceRandom | Randomly reads data from devices. |
| WriteDeviceRandom | Randomly writes data to devices. |
| SetDevice | Sets one device. |
| GetDevice | Acquires the data of one device. |
| ReadBuffer | Reads data from buffer memory. |
| WriteBuffer | Writes data to buffer memory. |
| GetClockData | Reads clock data from PLC CPU. |
| SetClockData | Writes clock data to PLC CPU. |
| GetCpuType | Reads PLC CPU type. |
| SetCpuStatus | Remote run/stop/pause of PLC CPU. |

### (2) Functions available for the ACT controls

Refer to "CHAPTER 4 FUNCTIONS" for full information on the functions available for the ACT controls.

# MEMO

# 2 ABOUT THE ACT CONTROLS

This chapter explains the settings made for use of the ACT controls, the programming procedures, the device types and the accessible ranges.



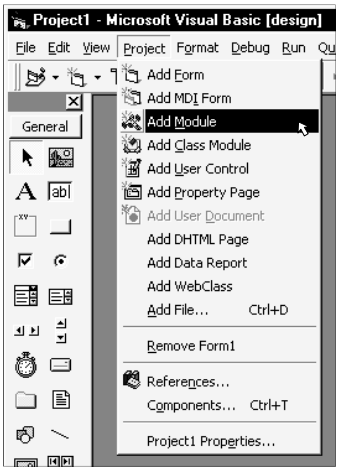
## 2.1 Settings Made for Use of the ACT Controls

This section describes the setting operation performed for use of the ACT controls.

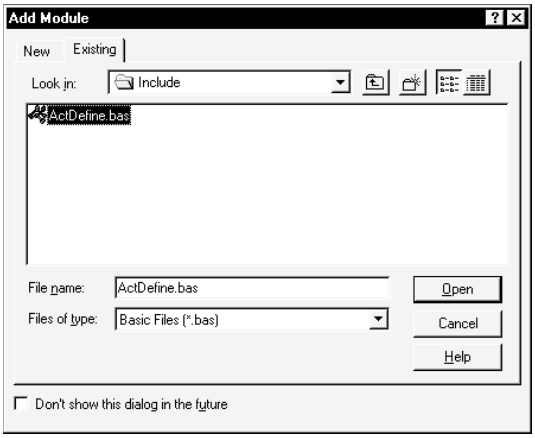
### 2.1.1 When using VB

Perform the following setting operation when using VB.

#### (1) Setting the include file

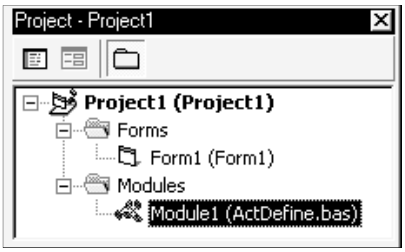



1) Start VB and choose the [Project]-[Add Module] menu.



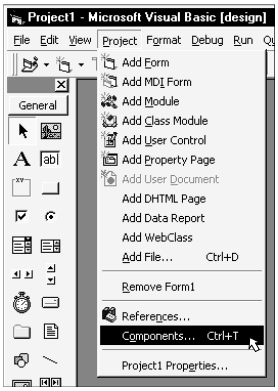

2) Choose the <<Existing>> tab and select "ActDefine.bas".

"ActDefine.bas" is stored in <User specified folder>-<Act>-<Include> at the time of installation.

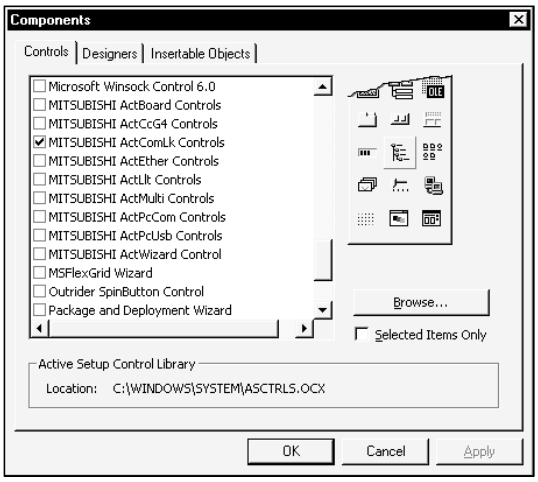



3) Registering "ActDefine.bas" adds it to Modules.

## (2) Registering the ACT controls




1) Choose the [Project]-[Components] menu.



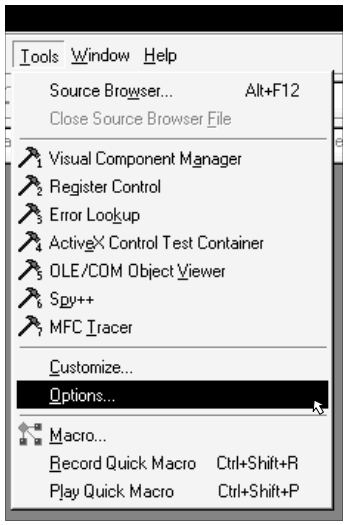

2) Select the <<Controls>> tab and choose the DLL which includes the ACT controls you want to use.

3) The ACT controls included in the selected DLL are added to the toolbox.

## 2.1.2 When using VC++

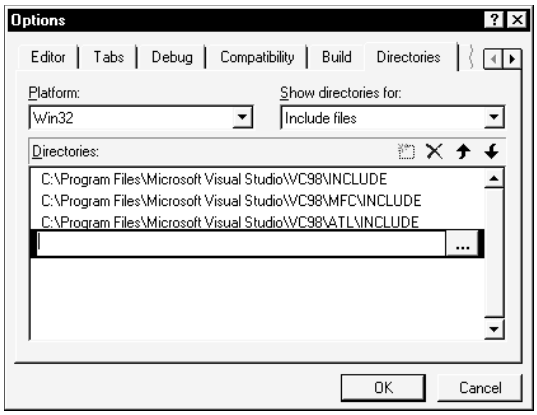
Perform the following setting operation when using VC++.

### (1) Setting the include file



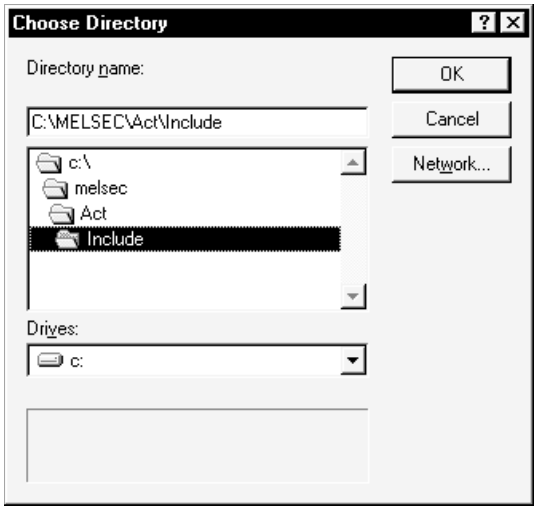

1) Start VC++ and choose the [Tools]-[Options] menu.

↓




2) Choose the <<Directories>> tab and set "Include files" in "Show directories for:".

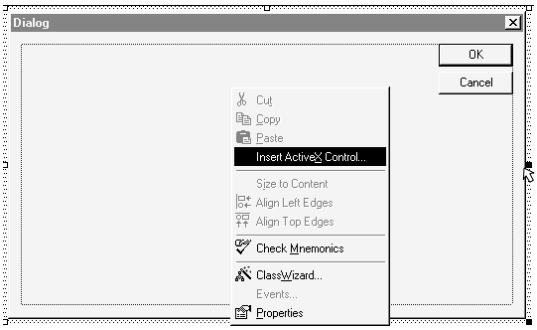
↓



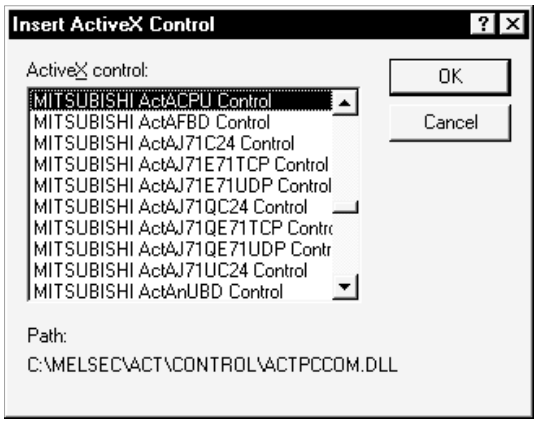

3) Double-click the item to be set, and browse the include file.

"ActDefine.H" is stored in <User specified folder>-<Act>-<Include> at the time of installation.

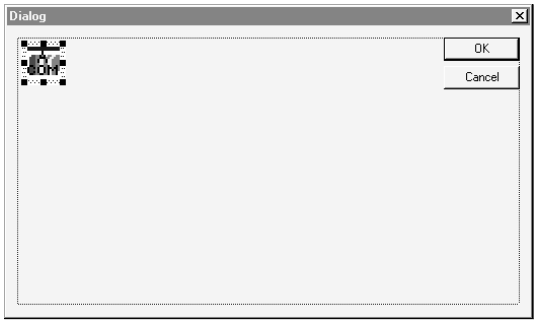
## (2) Registering the ACT control




1) Right-click the form to choose "Insert ActiveX Control".

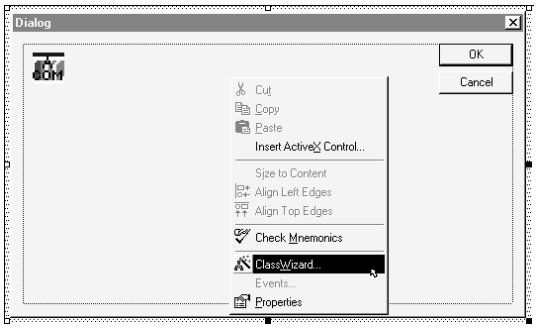



2) Select the ACT control you want to use.



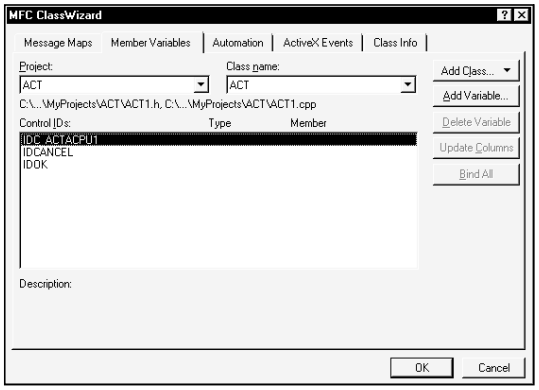

3) The selected ACT control is pasted to the form.

### (3) Adding the member variable



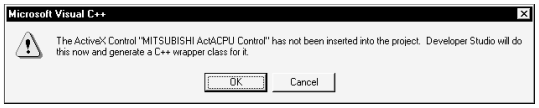

1) Click the form to choose "Class Wizard".

↓



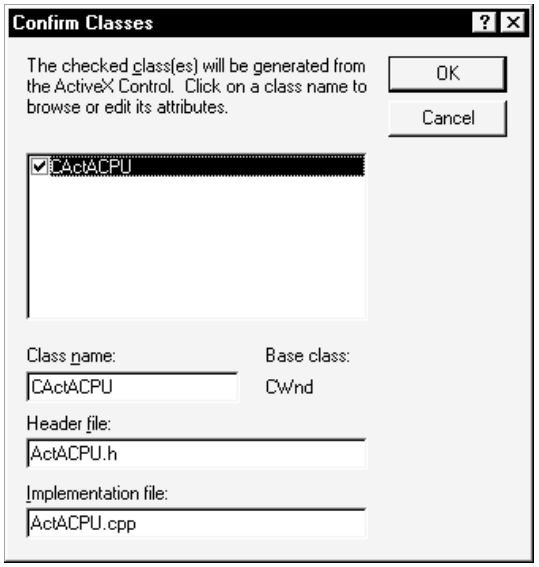

2) When the left dialog box appears, choose the <<Member Variables>> tab.
Choose the member variable adding control ID and click the Add Variable button.

↓




3) When the left screen appears, read the information and click the OK button.

↓



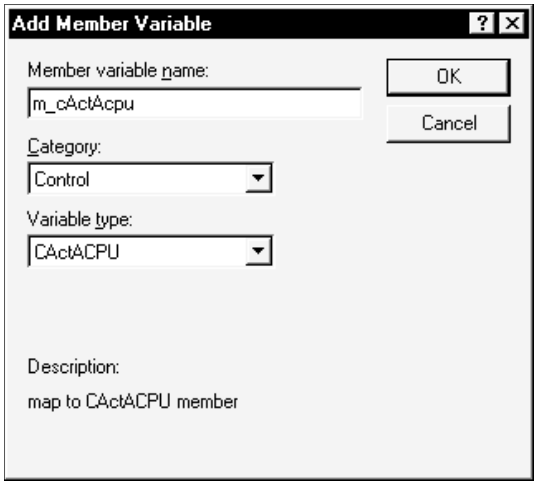

4) Check the class checkbox and click the OK button.

↓

(To the next page.)

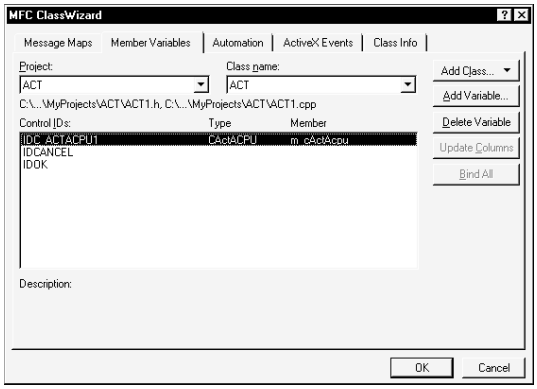
(From the previous page)

↓




5) Enter the member variable name and click the OK button.

↓



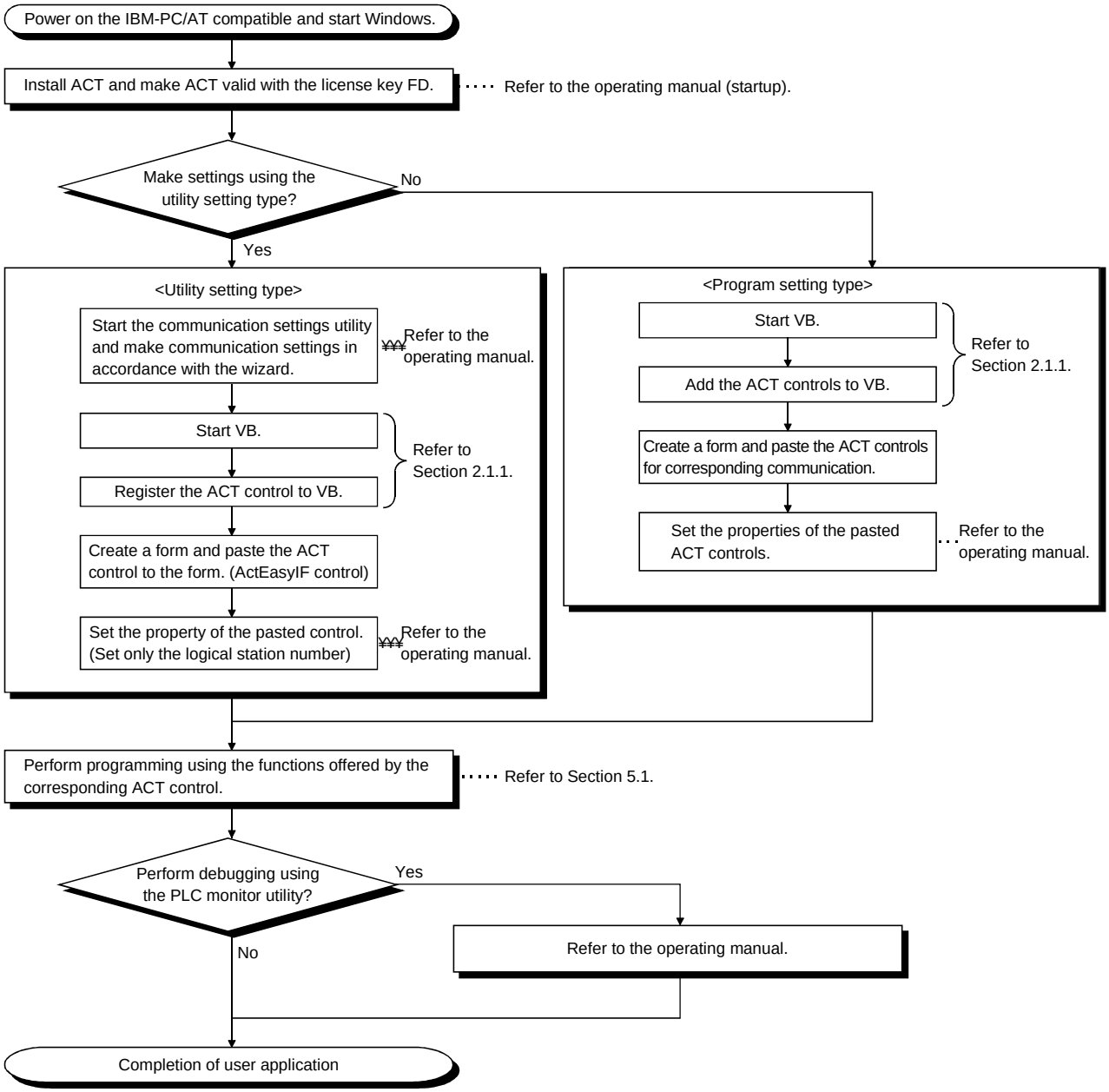

6) Make sure that the member variable has been registered.

## 2.2 Programming Procedures

This section gives the procedures of creating a user application.

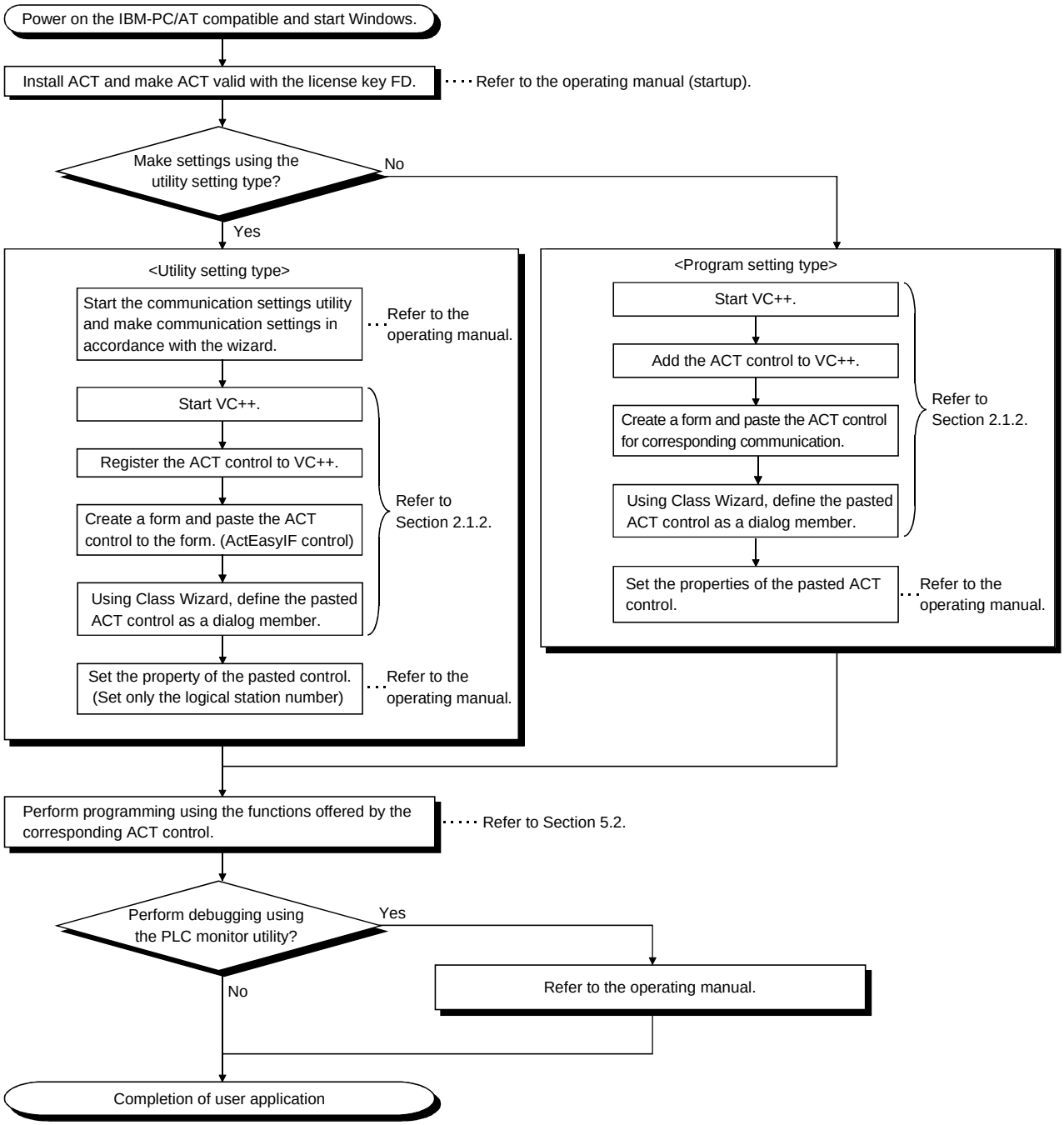
### 2.2.1 When using VB

When using VB, create a user application in the following procedure.



## 2.2.2 When using VC++

When using VC++, create a user application in the following procedure.



## 2.3 Device Types

This section explains the devices that may be specified for the functions.

POINT

(1) For the functions (ReadDeviceBlock, WriteDeviceBlock, ReadDeviceRandom, WriteDeviceRandom, SetDevice and GetDevice), specify the devices in the form of "device name + device number".
For the device numbers, note the differences between octal, decimal and hexadecimal numbers.
(2) When specifying bit devices for ReadDeviceBlock or WriteDeviceBlock, specify the device number as a multiple of 16.
(3) Only the devices indicated in this section are supported.
Do not use unsupported devices.

### (1) Common

The following device types are common to all communication paths.

| Device | | Device Name | Device No. Type | Representation | Remarks |
|---|---|---|---|---|---|
| Function input | | FX | Decimal | Bit | — |
| Function output | | FY | Decimal | Bit | — |
| Function register | | FD | Decimal | Word | 4 words/1 point ∗1 |
| Special relay | | SM | Decimal | Bit | — |
| Special register | | SD | Decimal | Word | — |
| Input relay | | X | Hexadecimal | Bit | Octal for FXCPU |
| Output relay | | Y | Hexadecimal | Bit | Octal for FXCPU |
| Internal relay | | M | Decimal | Bit | ∗2 |
| Latch relay | | L | Decimal | Bit | ∗2 |
| Annunciator | | F | Decimal | Bit | — |
| Edge relay | | V | Decimal | Bit | — |
| Link relay | | B | Decimal | Bit | — |
| Data register | | D | Decimal | Word | — |
| Link register | | W | Hexadecimal | Word | — |
| Timer | Contact | TS | Decimal | Bit | — |
| | Coil | TC | Decimal | Bit | — |
| | Present value | TN | Decimal | Word | — |
| Counter | Contact | CS | Decimal | Bit | — |
| | Coil | CC | Decimal | Bit | — |
| | Present value | CN | Decimal | Word | For FXCPU, 200 or more is 32-bit data. |
| Retentive timer | Contact | SS | Decimal | Bit | For ACPU, use timer to specify. |
| | Coil | SC | Decimal | Bit | For ACPU, use timer to specify. |
| | Present value | SN | Decimal | Word | For ACPU, use timer to specify. |
| Link special relay | | SB | Hexadecimal | Bit | — |
| Link special register | | SW | Hexadecimal | Word | — |
| Step relay | | S | Decimal | Bit | ∗2 |

Bit: Bit device Word: Word device

∗1: For batch operation, operation is performed continuously in units of one word.
For random operation, only the first one word is read.

∗2: For the QCPU (A mode) and ACPU, the M, L and S devices have the same regions independently of the device setting in the parameters.

| Device | | Device Name | Device No. Type | Representation | Remarks |
|---|---|---|---|---|---|
| Accumulator | | A | Decimal | Word | ∗5 |
| Index register | | Z | Decimal | Word | ∗5 |
| | | V | Decimal | Word | ∗5 |
| File register | | R | Decimal | Word | ∗3 |
| | | ZR | Decimal | Word | — |
| Extended file register | | ER∗\R | Decimal | Word | ∗4 |
| Direct link∗6 | Link input | J∗\X | Hexadecimal | Bit | ∗4 |
| | Link output | J∗\Y | Hexadecimal | Bit | ∗4 |
| | Link relay | J∗\B | Hexadecimal | Bit | ∗4 |
| | Link special relay | J∗\SB | Hexadecimal | Bit | ∗4 |
| | Link register | J∗\W | Hexadecimal | Bit | ∗4 |
| | Link special register | J∗\SW | Hexadecimal | Word | ∗4 |
| Special direct buffer memory∗7 | | U∗\G | Hexadecimal /decimal | Word | ∗4, ∗8 |

Bit: Bit device Word: Word device

∗3: To specify the extended file register, describe "\" between the block number part and file register part.
Specifying R∗∗ specifies R of block No. 0.
Specifying ER0\R∗∗ returns an error.
Specifying ER∗∗\R∗∗ does not enable extension representation (indirect specification, digit specification).

∗4: For direct specification, describe "\" between the direct specification part and device specification part.

∗5: Cannot be used when E71 is relayed.

∗6: For J∗, specify the network number.

∗7: Specify the special module I/O number (hexadecimal) for U∗, and the buffer memory address (decimal) for G∗∗.
(Example: Specify "U20\G100" when the special module I/O number is 200H and the buffer memory address is 100.)

∗8: FXCPU cannot be used.

### (2) For CC-Link communication only

For CC-Link communication only, the devices in the following table can be used when own board access is made. They cannot be used for other communication paths.

| Device | Device Name | Device No. Type | Representation | Remarks |
|---|---|---|---|---|
| Special relay | SM | Bit | Decimal | Special relay of own board |
| Special register | SD | Word | Decimal | Special register of own board |
| Link special register (for CC-Link) | SB | Bit | Hexadecimal | Link special relay of own board |
| Link special register (for CC-Link) | SW | Word | Hexadecimal | Link special register of own board |
| Remote input | X | Bit | Hexadecimal | RX |
| Remote output | Y | Bit | Hexadecimal | RY |
| Link register | W | Word | Hexadecimal | — |
| Remote register (write area for CC-Link) | WW | Word | Hexadecimal | RWw |
| Remote register (read area for CC-Link) | WR | Word | Hexadecimal | RWr |
| Buffer memory | ML | Word | Hexadecimal | Buffer memory of own station CC-Link module |
| Random access buffer | MC | Word | Hexadecimal | Random access buffer in buffer memory of own station CC-Link module |
| Automatic refresh buffer | MF | Bit | Hexadecimal | Automatic refresh buffer of own station CC-Link module |

### (3) About device extension representation

The following table indicates whether the device extension representations are usable or not for the available CPUs.
They cannot be used with ReadDeviceBlock and WriteDeviceBlock.
When the ActAJ71E71TCP control or ActAJ71QE71TCP control is used, device expansion representation is unusable.

| Device Extension Representation | Target CPU | | | | | |
|---|---|---|---|---|---|---|
| | QCPU | | QnACPU | ACPU | FXCPU | Motion controller CPU |
| | Q mode | A mode | | | | |
| Digit specification (example: K4M0) ＊2 | ○ | ○ | ○ | ○ | ○ | ○ |
| Bit specification (example: D0.1) ＊3 | ○ | ○ | ○ | ○ | ○ | ○ |
| Index qualification (example: M100Z0) ＊4 | ○ | × | ○＊1 | × | × | × |

○: Usable ×: Unusable

＊1: Unusable when QE71 is relayed.
＊2: FX/FX, DX/DY and T/C/ST (contact, coil) cannot be specified.
＊3: Z, V, T/C/ST (present value) cannot be specified.
＊4: FX/FX, DX/DY, T/C/ST (contact, coil), Z and S cannot be specified.

## 2.4 Accessible Devices and Ranges

Refer to the ACT operating manual for the accessible devices and ranges for corresponding communication.

# MEMO

# 3 DETAILS OF THE ACT CONTROLS

This chapter describes the details of the ACT controls, the details of the properties, and the possessed property list.

## 3.1 Details of the ACT Controls

The following table lists the definitions and usable setting types of the ACT controls.

| Control Name | Definition | Usable Setting Type |
|---|---|---|
| ActEasyIF | Can communicate with any communication path.<br>Use the communication settings utility to set the information for communication. | Utility setting type |
| ActQJ71E71TCP | Used for Ethernet communication where the connected module is the Q series-compatible E71 (TCP/IP communication). | Program setting type |
| ActQJ71E71UDP | Used for Ethernet communication where the connected module is the Q series-compatible E71 (UDP/IP communication). | Program setting type |
| ActAJ71QE71TCP | Used for Ethernet communication where the connected module is the QE71 (TCP/IP communication). | Program setting type |
| ActAJ71QE71UDP | Used for Ethernet communication where the connected module is the QE71 (UDP/IP communication). | Program setting type |
| ActAJ71E71TCP | Used for Ethernet communication where the connected module is the E71 (TCP/IP communication). | Program setting type |
| ActAJ71E71UDP | Used for Ethernet communication where the connected module is the E71 (UDP/IP communication). | Program setting type |
| ActQCPUQ | Used for CPU COM communication where the connected PLC CPU is the QCPU (Q mode). | Program setting type |
| ActQCPUA | Used for CPU COM communication where the connected PLC CPU is the QCPU (A mode). | Program setting type |
| ActQnACPU | Used for CPU COM communication where the connected PLC CPU is the QnACPU. | Program setting type |
| ActACPU | Used for CPU COM communication where the connected PLC CPU is the ACPU (including motion controller CPU). | Program setting type |
| ActFXCPU | Used for CPU COM communication where the connected PLC CPU is the FXCPU. | Program setting type |
| ActQJ71C24 | Used for computer link communication where the connected module is the Q series-compatible C24. | Program setting type |
| ActAJ71QC24 | Used for computer link communication where the connected module is the QC24(N). | Program setting type |
| ActAJ71UC24 | Used for computer link communication where the connected module is the UC24. | Program setting type |
| ActAJ71C24 | Used for computer link communication where the connected module is the C24. | Program setting type |
| ActQCPUQUSB | Used for USB communication where the connected PLC CPU is the QCPU (Q mode). | Program setting type |
| ActCCG4QnA | Used for CC-Link G4 communication where the connected module is the AJ65BT-G4 (QnA mode). | Program setting type |
| ActCCG4A | Used for CC-Link G4 communication where the connected module is the AJ65BT-G4 (A mode). | Program setting type |
| ActMnet10BD | Used for MELSECNET/10 communication. | Program setting type |
| ActCCBD | Used for CC-Link communication. | Program setting type |
| ActAnUBD | Used for CPU board communication. | Program setting type |
| ActLLt | Used for ladder logic test communication. | Program setting type |

## 3.2 Details of the Properties

The following tables give the details of the properties which must be set to create a user application.

**POINT**

When entering a property value directly into the property window of VB or VC++, change a character string such as a hexadecimal number or CPU type into a decimal property value.

<table>
<tr><th>Property Name (Type)</th><th>Description</th></tr>
<tr><td>ActLogicalCtationNumber<br>(LONG)</td><td>Logical station number set on the communication settings utility.</td></tr>
<tr><td>ActNetworkNumber<br>(LONG)</td><td>Specify the network number on the MELSECNET/10(H). (Specify "0x00" when specifying the own station.)<br>Specify as follows for multidrop connection (via Q series-compatible C24, QJ61BT11).
<table>
<tr><th>ActIntelligentPreferenceBit value</th><th>Description</th></tr>
<tr><td>0x00</td><td>Specify the own network.</td></tr>
<tr><td>0x01</td><td>Specify another network of multidrop destination.</td></tr>
</table></td></tr>
<tr><td>ActStationNumber<br>(LONG)</td><td>Specify the station number for MELSECNET/10(H) or CC-Link. (Specify "0x00" when specifying the own station.)<br>Handled as the own station when access to the CPU of the CPU board is made.<br>Specify as follows for multidrop connection (via Q series-compatible C24, QJ61BT11).
<table>
<tr><th>ActIntelligentPreferenceBit value</th><th>Description</th></tr>
<tr><td>0x00</td><td>Specify the own network.</td></tr>
<tr><td>0x01</td><td>Specify another network of multidrop destination.</td></tr>
</table></td></tr>
<tr><td>ActUnitNumber<br>(LONG)</td><td>Specify the module number of the computer link module or the station number when the target is the QCPU-compatible intelligent special function module. However, specify "00x0" when setting the QnA series own station (module loaded to the own station CPU).<br>Invalid when the target is not the computer link communication or QCPU-compatible intelligent special function module.<br>For multidrop link, specify the module number of the target computer link module.</td></tr>
<tr><td>ActConnectUnitNumber<br>(LONG)</td><td>Specify the module number of the computer link module, QE71 or Q series-compatible E71.<br>For multidrop link, specify the module number of the requesting computer link module. For multidrop link via CPU COM communication, however, the module number of the requesting station is not needed (specify "00x0").<br>Specify "0x00" for other than multidrop link.<br>For the QE71 and Q series-compatible E71, specify the relay target station number (fixed to "0x00" for access within the own network).<br>For access to another network via MELSECNET/10, specify the station number set in the parameter of the connected Ethernet module.</td></tr>
<tr><td>ActIONumber<br>(LONG)</td><td>Specify the module I/O number.<br>For multidrop link or intelligent special function module access, specify the actual I/O number (first I/O number÷16) of the target computer link module or intelligent special function module (specify the I/O number of the relayed or requesting station for multidrop link).<br>Specify "0x3FF" when making access to another station via the own station CPU or network.</td></tr>
</table>

<table>
<tr><th>Property Name(Type)</th><th>Description</th></tr>
<tr><td>ActCpuType<br>(LONG)</td><td>Specify the target CPU to communicate with.<br>In the parameter, specify any of the CPU types in the following table.
<table>
<tr><th>Property value<br>(Property window input value)</th><th>Target CPU</th><th>Property value<br>(Property window input value)</th><th>Target CPU</th></tr>
<tr><td>CPU_Q02CPU (0x22)</td><td>Q02(H)CPU</td><td>CPU_A3NCPU (0x10A)</td><td>A3NCPU</td></tr>
<tr><td>CPU_Q06CPU (0x23)</td><td>Q06HCPU</td><td rowspan="2">CPU_A2ACPU (0x10C)</td><td rowspan="2">A2ACPU (－S1),<br>A2ACPUP21/R21(－S1)</td></tr>
<tr><td>CPU_Q12CPU (0x24)</td><td>Q12HCPU</td></tr>
<tr><td>CPU_Q25CPU (0x25)</td><td>Q25HCPU</td><td>CPU_A3ACPU (0x10D)</td><td>A3ACPU,<br>A3ACPUP21/R21</td></tr>
<tr><td>CPU_Q02CPU_A (0x141)</td><td>Q02(H)CPU-A</td><td rowspan="2">CPU_A2UCPU (0x10E)</td><td rowspan="2">A2UCPU (－S1),<br>A2USCPU (－S1),<br>A2ASCPU (－S1)</td></tr>
<tr><td>CPU_Q06CPU_A (0x142)</td><td>Q06HCPU-A</td></tr>
<tr><td>CPU_Q2ACPU (0x11)</td><td>Q2ACPU, Q2ASCPU,<br>Q2ASHCPU</td><td>CPU_A2USHS1CPU (0x10F)</td><td>A2USHCPU-S1CPU,<br>CPU board</td></tr>
<tr><td>CPU_Q2AS1CPU (0x12)</td><td>Q2ACPU-S1,<br>Q2ASCPU(－S1),<br>Q2ASHCPU(－S1)</td><td>CPU_A3UCPU (0x110)</td><td>A3UCPU,<br>A2ASCPU-S30</td></tr>
<tr><td></td><td></td><td>CPU_A4UCPU (0x111)</td><td>A4UCPU</td></tr>
<tr><td>CPU_Q3ACPU (0x13)</td><td>Q3ACPU</td><td>CPU_FX0CPU (0x201)</td><td>FX0, FX0S</td></tr>
<tr><td>CPU_Q4ACPU (0x14)</td><td>Q4ACPU, Q4ARCPU</td><td>CPU_FX0NCPU (0x202)</td><td>FX0N</td></tr>
<tr><td>CPU_A0J2HCPU (0x102)</td><td>A0J2HCPU</td><td>CPU_FX1CPU (0x203)</td><td>FX1</td></tr>
<tr><td>CPU_A1FXCPU (0x103)</td><td>A1FXCPU</td><td>CPU_FX2CPU (0x204)</td><td>FX2, FX2C</td></tr>
<tr><td>CPU_A1SCPU (0x104)</td><td>A1SCPU(－S1),<br>A1SCPUC24-R2,<br>A1SJCPU</td><td>CPU_FX2NCPU (0x205)</td><td>FX2N, FX2NC</td></tr>
<tr><td></td><td></td><td>CPU_FX1SCPU (0x206)</td><td>FX1S</td></tr>
<tr><td>CPU_A1SHCPU (0x105)</td><td>A1SHCPU, A1SJHCPU</td><td>CPU_FX1NCPU (0x207)</td><td>FX1N</td></tr>
<tr><td>CPU_A1NCPU (0x106)</td><td>A1NCPU</td><td>CPU_A171SHCPU (0x601)</td><td>A171SHCPU</td></tr>
<tr><td rowspan="2">CPU_A2CCPU (0x107)</td><td rowspan="2">A2CCPU,<br>A2CCPUC24 (－PRF),<br>A2CJCPU</td><td>CPU_A172SHCPU (0x602)</td><td>A172SHCPU</td></tr>
<tr><td>CPU_A273UHCPU (0x603)</td><td>A273UHCPU (－S3)</td></tr>
<tr><td>CPU_A2NCPU (0x108)</td><td>A2NCPU (－S1),<br>A2SCPU (－S1)</td><td>CPU_A173UHCPU (0x604)</td><td>A173UHCPU (－S1)</td></tr>
<tr><td>CPU_A2SHCPU (0x109)</td><td>A2SHCPU (－S1)</td><td>CPU_BOARD (0x401)</td><td>For own board access＊1</td></tr>
</table>
*1: Except CPU board</td></tr>
</table>

<table>
<tr><th>Property Name(Type)</th><th>Description</th></tr>
<tr><td>ActPortNumber<br>(LONG)</td><td>Specify the connection port number of the IBM-PC/AT compatible.<br>When the Ethernet module is connected, set any value as the port number of the requesting source (IBM-PC/AT compatible).<br>When "=0" was specified as the port number, the MELSECNET/10 routing system should be the automatic response system. (When the system selected is other than the automatic response system via QE71, you should set the fixed value "5001".)<br>Also, when the control for network board is used, specify the first board as PORT_1, and the second and subsequent boards as PORT_2, PORT_3 ...
<table>
<tr><th>Property value<br>(Property window input value)</th><th>Description</th></tr>
<tr><td>PORT_1 (0x01)</td><td>Communication port 1</td></tr>
<tr><td>PORT_2 (0x02)</td><td>Communication port 2</td></tr>
<tr><td>PORT_3 (0x03)</td><td>Communication port 3</td></tr>
<tr><td>PORT_4 (0x04)</td><td>Communication port 4</td></tr>
<tr><td>PORT_5 (0x05)</td><td>Communication port 5</td></tr>
<tr><td>PORT_6 (0x06)</td><td>Communication port 6</td></tr>
<tr><td>PORT_7 (0x07)</td><td>Communication port 7</td></tr>
<tr><td>PORT_8 (0x08)</td><td>Communication port 8</td></tr>
<tr><td>PORT_9 (0x09)</td><td>Communication port 9</td></tr>
<tr><td>PORT_10 (0x0A)</td><td>Communication port 10</td></tr>
</table></td></tr>
<tr><td>ActBaudRate<br>(LONG)</td><td>Specify the baudrate for computer link communication.
<table>
<tr><th>Property value<br>(Property window input value)</th><th>Description</th><th>Property value<br>(Property window input value)</th><th>Description</th></tr>
<tr><td>BAUDRATE_300<br>(300)</td><td>300bps</td><td>BAUDRATE_9600<br>(9600)</td><td>9600bps</td></tr>
<tr><td>BAUDRATE_600<br>(600)</td><td>600bps</td><td>BAUDRATE_19200<br>(19200)</td><td>19200bps</td></tr>
<tr><td>BAUDRATE_1200<br>(1200)</td><td>1200bps</td><td>BAUDRATE_38400<br>(38400)</td><td>38400bps</td></tr>
<tr><td>BAUDRATE_2400<br>(2400)</td><td>2400bps</td><td>BAUDRATE_57600<br>(57600)</td><td>57600bps</td></tr>
<tr><td>BAUDRATE_4800<br>(4800)</td><td>4800bps</td><td>BAUDRATE_115200<br>(115200)</td><td>115200bps</td></tr>
</table></td></tr>
<tr><td>ActDataBit(LONG)</td><td>Specify the number of bits (7 or 8) of the byte data sent and received for computer link communication.</td></tr>
<tr><td>ActParity<br>(LONG)</td><td>Specify the parity system used for computer link communication.
<table>
<tr><th>Property value<br>(Property window input value)</th><th>Description</th></tr>
<tr><td>NO_PARITY (0)</td><td>No parity</td></tr>
<tr><td>ODD_PARITY (1)</td><td>Odd</td></tr>
<tr><td>EVEN_PARITY (2)</td><td>Even</td></tr>
</table></td></tr>
</table>

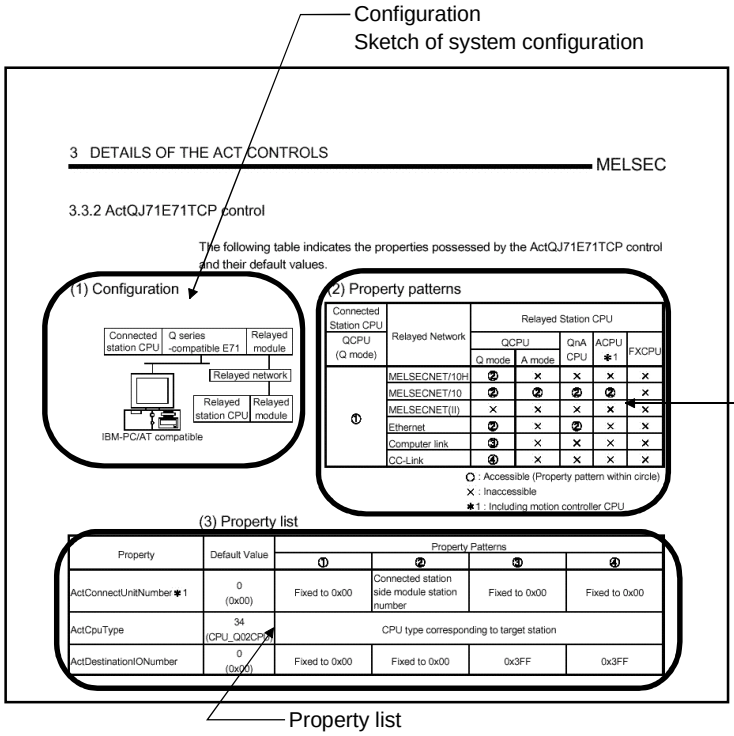
<table>
<tr><th>Property Name(Type)</th><th>Description</th></tr>
<tr><td>ActStopBit<br>(LONG)</td><td>Specify the number of stop bits used for computer link communication
<table>
<tr><th>Property value<br>(Property window input value)</th><th>Description</th></tr>
<tr><td>STOPBIT_ONE (0)</td><td>1 stop bit</td></tr>
<tr><td>STOPBITS_TWO (2)</td><td>2 stop bits</td></tr>
</table></td></tr>
<tr><td>ActControl<br>(LONG)</td><td>Specify the control setting of the signal line.
<table>
<tr><th>Property value<br>(Property window input value)</th><th>Description</th></tr>
<tr><td>TRC_DTR (0x01)</td><td>DTR control</td></tr>
<tr><td>TRC_RTS (0x02)</td><td>RTS control</td></tr>
<tr><td>TRC_DRT_AND_RTS (0x07)</td><td>DTR control and RTS control</td></tr>
<tr><td>TRC_DTR_OR_RTS (0x08)</td><td>DTR control or RTS control</td></tr>
</table></td></tr>
<tr><td>ActHostAddress(BSTR)</td><td>Pointer which indicates the connection host name (IP address) for Ethernet communication.</td></tr>
<tr><td>ActCpuTimeOut(LONG)</td><td>Specify the CPU watchdog timer for Ethernet communication. (Unit = "×250ms")</td></tr>
<tr><td>ActTimeOut(LONG)</td><td>Set the time-out value of communication between the IBM-PC/AT compatible and PLC. (Unit = "ms")</td></tr>
<tr><td>ActSumCheck<br>(LONG)</td><td>Specify whether sumcheck is made or not.<br>Valid only via computer link module.
<table>
<tr><th>Property value<br>(Property window input value)</th><th>Description</th></tr>
<tr><td>NO_SUM_CHECK (0)</td><td>Without sumcheck</td></tr>
<tr><td>SUM_CHECK (1)</td><td>With sumcheck</td></tr>
</table></td></tr>
<tr><td>ActSourceNetworkNumber<br>(LONG)</td><td>Specify the requesting network number when the QE71 or Q series-compatible E71 is specified.<br>Specify the same network number as for the connected QE71 or Q series-compatible E71 (network number specified in the network parameter).</td></tr>
<tr><td>ActSourceStationNumber<br>(LONG)</td><td>Specify the requesting station number (IBM-PC/AT compatible side station number) when the QE71 or Q series-compatible E71 is specified.<br>Make setting to avoid setting the same station number as that of the QE71 set within the same Ethernet loop.</td></tr>
<tr><td>ActDestinationPort<br>Number<br>(LONG)</td><td>Specify the port number of the target when Ethernet communication is specified.<br>For access to another network, specify the relay destination port number.<br>For other than the automatic response system, make setting as indicated in the following table.
<table>
<tr><th>Communication</th><th>Setting</th></tr>
<tr><td>QE71(UDP/IP)</td><td>Fixed to "5001"</td></tr>
<tr><td>Q series-compatible E71 (TCP/IP)</td><td>Fixed to "5002"</td></tr>
<tr><td>Q series-compatible E71 (UDP/IP)</td><td>Fixed to "5001"</td></tr>
</table></td></tr>
<tr><td>ActDestinationIONumber<br>(LONG)</td><td>For multidrop connection (via Q series-compatible C24/CC-Link), specify the actual I/O number (first I/O÷16) of the last access target station. (When the target is the intelligent special function module)<br>When the target is the CPU, specify "0x3FF".</td></tr>
</table>

<table>
<tr><th>Property Name(Type)</th><th>Description</th></tr>
<tr><td>ActMultiDropChannel Number<br>(LONG)</td><td>For multidrop connection (via Q series-compatible C24/CC-Link), specify the multidrop connection channel number (Ch1/Ch2).<br>Invalid for other connections.</td></tr>
<tr><td>ActThroughNetworkType<br>(LONG)</td><td>You can select the MELSECNET/10H or MELSECNET/10 mode to make access to the own station QCPOU (Q mode) or to the QCPU (Q mode) via the MELSECNET/10H when using the ActQJ71C24, ActQJ71E71TCP, ActQJ71E71UDP, ActQCPUQ or ActQCPUQUSB control.<br>When the control used is other than the above, the mode is fixed to the MELSECNET/10 mode.
<table>
<tr><th>Property value</th><th>Description</th></tr>
<tr><td>0x00</td><td>MELSECNET/10H mode</td></tr>
<tr><td>0x01</td><td>MELSECNET/10 mode</td></tr>
</table></td></tr>
<tr><td>ActIntelligent PreferenceBit<br>(LONG)</td><td>For multidrop connection (via Q series-compatible C24/CC-Link), specify whether the network of the multidrop link destination will be relayed or not. (To differentiate the own network module.)
<table>
<tr><th>Property value</th><th>Description</th></tr>
<tr><td>0x00</td><td>Another network of multidrop link destination is not accessed.</td></tr>
<tr><td>0x01</td><td>Another network of multidrop link destination is accessed.</td></tr>
</table></td></tr>
<tr><td>ActDidPropertyBit<br>(LONG)</td><td>For access to the Q series-compatible own station intelligent special function module (intelligent special function module load on the own station CPU), making the following setting invalid makes it unnecessary to specify "ActUnitNumber". (Only "ActIONumber" is used to specify the module I/O number.)
<table>
<tr><th>Property value</th><th>Description</th></tr>
<tr><td>0x00</td><td>Module number is made valid.</td></tr>
<tr><td>0x01</td><td>Module number is made invalid.</td></tr>
</table></td></tr>
<tr><td>ActDsidPropetyBit<br>(LONG)</td><td>For multidrop connection (via Q series-compatible C24/CC-Link), making the following setting invalid makes it unnecessary to specify "ActDestinationIONumber".<br>However, when the following setting is made invalid, "ActDidPropertyBit" must be made valid. (Use "ActUnitNumber" to specify.)
<table>
<tr><th>Property value</th><th>Description</th></tr>
<tr><td>0x00</td><td>I/O number of the last access target station is made valid.</td></tr>
<tr><td>0x01</td><td>I/O number of the last access target station is made invalid.</td></tr>
</table></td></tr>
</table>

## 3.3 Lists of Properties Possessed by the ACT Controls

This section lists the properties possessed by the ACT controls and their default values. How to use the manual in Section 3.3.1 to Section 3.3.23 is provided below.

<How to use the manual in Section 3.3.1 to Section 3.3.23>




(1) Property
Gives the property name.

(2) Default value
- Gives the default value of the property.
- The default values used when the properties are changed in the program are given within the "parentheses".

(3) Property pattern
Gives the property settings necessary to make communication settings.
Refer to the "property pattern table" for the property pattern numbers.

**POINT**

The default values indicated are the property values shown in the property window of VB or VC++.
The default values of the properties, whose values must be changed in other than decimal when changed in a program, are indicated in parentheses.

### 3.3.1 ActEasyIF control

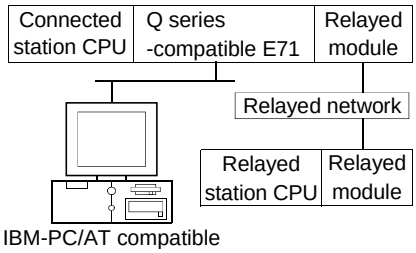
The following table indicates the property possessed by the ActEasyIF control and its default value.

| Property | Default Value | Property Pattern |
|---|---|---|
| ActLogicalStationNumber | 0 | Logical station number set on the communication settings utility |

## 3.3.2 ActQJ71E71TCP control

The following table indicates the properties possessed by the ActQJ71E71TCP control and their default values.

### (1) Configuration




### (2) Property patterns

| Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|
| QCPU (Q mode) | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | | Q mode | A mode | | | |
| ① | MELSECNET/10H | ② | × | × | × | × |
| | MELSECNET/10 | ② | ② | ② | ② | × |
| | MELSECNET(II) | × | × | × | × | × |
| | Ethernet | ② | × | ② | × | × |
| | Computer link | ③ | × | × | × | × |
| | CC-Link | ④ | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU

### (3) Property list

| Property | Default Value | Property Patterns | | | |
|---|---|---|---|---|---|
| | | ① | ② | ③ | ④ |
| ActConnectUnitNumber ∗1 | 0 (0x00) | Fixed to 0x00 | Connected station side module station number | Fixed to 0x00 | Fixed to 0x00 |
| ActCpuType | 34 (CPU_Q02CPU) | CPU type corresponding to target station | | | |
| ActDestinationIONumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | 0x3FF | 0x3FF |
| ActDidPropertyBit | 1 (0x01) | 0x01 (invalid) | 0x01 (invalid) | 0x00 (valid) | 0x00 (valid) |
| ActDsidPropertyBit | 1 (0x01) | 0x01 (invalid) | 0x01 (invalid) | 0x00 (valid) | 0x00 (valid) |
| ActHostAddress | 1.1.1.1 | Host name or IP address of connected station side module | | | |
| ActIONumber | 1023 (0x3FF) | Fixed to 0x3FF | Fixed to 0x3FF | Connected station side relayed module I/O address | Connected station side relayed module I/O address |
| ActMultiDropChannelNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Fixed to 0x02 | Fixed to 0x00 |
| ActNetworkNumber ∗2 | 1 (0x01) | Network number of target station side module | Network number of target station side module | Connected station side Q series-compatible E71 network number | Connected station side Q series-compatible E71 network number |
| ActSourceNetworkNumber ∗3 | 1 (0x01) | IBM-PC/AT compatible side network number | | | |
| ActSourceStationNumber ∗4 | 2 (0x02) | IBM-PC/AT compatible side station number | | | |

∗1: For access to another station via MELSECNET/10 (for the property pattern of ② ), specify the station number of the connected station side Q series-compatible E71 set in the Ethernet parameter of the connected station side Q series-compatible E71.
∗2: For the property pattern of ① or ②, specify the value set in the target station side parameter for ActNetworkNumber and ActStationNumber.
∗3: Specify the same network number as the MELSECNET/10 network number set to the Q series-compatible E71 in the Ethernet parameter setting of the target station side Q series-compatible E71.
∗4: Specify the station number on the IBM-PC/AT compatible side to avoid setting the same station number as set to the Q series-compatible E71 within the same Ethernet loop.

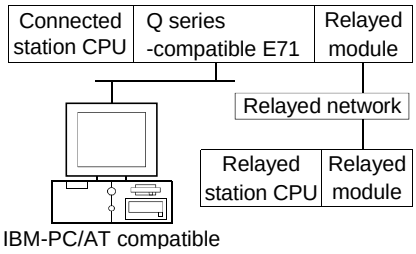
| Property | Default Value | Property Patterns | | | |
|---|---|---|---|---|---|
| | | ① | ② | ③ | ④ |
| ActStationNumber ∗2 | 1 (0x01) | Connected station side module station number | Connected station side module station number | Connected station side Q series-compatible E71 station number | Connected station side Q series-compatible E71 station number |
| ActThroughNetworkType | 0 (0x00) | QCPU (Q mode): 0x00 (MELSECNET/10H only), other than QCPU (Q mode): 0x01 (including MELSECNET/10). Note that the setting must be the same as set in the network parameter of the GPP function. | | | |
| ActTimeOut | 10000 | Any value specified by user in ms units. | | | |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Target station side module station number | Target station side module station number (valid) |

∗2: For the property pattern of ① or ②, specify the value set in the target station side parameter for ActNetworkNumber and ActStationNumber.

## 3.3.3 ActQJ71E71UDP control

The following table indicates the properties possessed by the ActQJ71E71UDP control and their default values.

### (1) Configuration




### (2) Property patterns

| Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|
| QCPU (Q mode) | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | | Q mode | A mode | | | |
| ① | MELSECNET/10H | ② | × | × | × | × |
| | MELSECNET/10 | ② | ② | ② | ② | × |
| | MELSECNET(II) | × | × | × | × | × |
| | Ethernet | ② | × | ② | × | × |
| | Computer link | ③ | × | × | × | × |
| | CC-Link | ④ | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU

### (3) Property list

| Property | Default Value | Property Patterns | | | |
|---|---|---|---|---|---|
| | | ① | ② | ③ | ④ |
| ActConnectUnitNumber ∗1 | 0 (0x00) | Fixed to 0x00 | Connected station side module station number | Fixed to 0x00 | Fixed to 0x00 |
| ActCpuType | 34 (CPU_Q02CPU) | CPU type corresponding to target station | | | |
| ActDestinationIONumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | 0x3FF | 0x3FF |
| ActDidPropertyBit | 1 (0x01) | 0x01 (invalid) | 0x01 (invalid) | 0x00 (valid) | 0x00 (valid) |
| ActDsidPropertyBit | 1 (0x01) | 0x01 (invalid) | 0x01 (invalid) | 0x00 (valid) | 0x00 (valid) |
| ActHostAddress | 1.1.1.1 | Host name or IP address of connected station side module | | | |
| ActIONumber | 1023 (0x3FF) | Fixed to 0x3FF | Fixed to 0x3FF | Connected station side relayed module I/O address | Connected station side relayed module I/O address |
| ActMultiDropChannelNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Fixed to 0x02 | Fixed to 0x00 |
| ActNetworkNumber ∗2 | 1 (0x01) | Network number of target station side module | Network number of target station side module | Connected station side Q series-compatible E71 network number | Connected station side Q series-compatible E71 network number |
| ActPortNumber ∗4 | 5001 | IBM-PC/AT compatible side port number | | | |
| ActSourceNetworkNumber ∗3 | 1 (0x01) | IBM-PC/AT compatible side network number | | | |

∗1: For access to another station via MELSECNET/10 (for the property pattern of ②), specify the station number of the connected station side Q series-compatible E71 set in the Ethernet parameter of the connected station side Q series-compatible E71.
∗2: For the property pattern of ① or ②, specify the value set in the target station side parameter for ActNetworkNumber and ActStationNumber.
∗3: Specify the same network number as the MELSECNET/10 network number set to the Q series-compatible E71 in the Ethernet parameter setting of the target station side Q series-compatible E71.
∗4: Do not use 1 to 1024 of ActPortNumber.

| Property | Default Value | Property Patterns | | | |
|---|---|---|---|---|---|
| | | ① | ② | ③ | ④ |
| ActSourceStationNumber＊5 | 2 (0x02) | IBM-PC/AT compatible side station number | | | |
| ActStationNumber＊2 | 1 (0x01) | Target station side module station number | Target station side module station number | Connected station side Q series-compatible E71 station number | Connected station side Q series-compatible E71 station number |
| ActThroughNetworkType | 0 (0x00) | QCPU (Q mode): 0x00 (MELSECNET/10H only), other than QCPU (Q mode): 0x01 (including MELSECNET/10). Note that the setting must be the same as set in the network parameter of the GPP function. | | | |
| ActTimeOut | 10000 | Any value specified by user in ms units. | | | |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Target station side module station number | Target station side module station number |

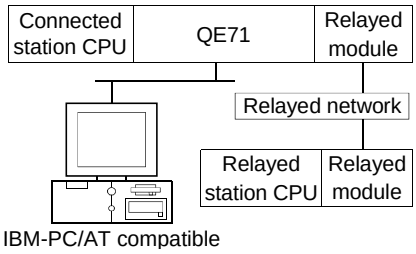
＊2: For the property pattern of ① or ②, specify the value set in the target station side parameter for ActNetworkNumber and ActStationNumber.

＊5: Specify the station number on the IBM-PC/AT compatible side to avoid setting the same station number as set to the Q series-compatible E71 within the same Ethernet loop.

## 3.3.4 ActAJ71QE71TCP control

The following table indicates the properties possessed by the ActAJ71QE71TCP control and their default values.

### (1) Configuration




### (2) Property patterns

| Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|
| QnACPU | | QCPU | | QnA CPU | ACPU ＊1 | FXCPU |
| | | Q mode | A mode | | | |
| ① | MELSECNET/10H | × | × | × | × | × |
| | MELSECNET/10 | × | × | ② | × | × |
| | MELSECNET(II) | × | × | × | × | × |
| | Ethernet | × | × | × | × | × |
| | Computer link | × | × | × | × | × |
| | CC-Link | × | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
＊1 : Including motion controller CPU

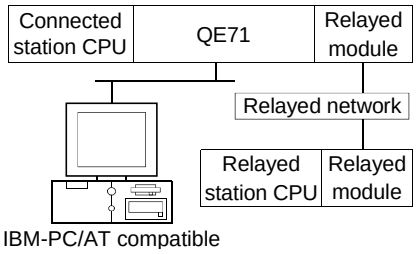
### (3) Property list

| Property | Default Value | Property Patterns | |
|---|---|---|---|
| | | ① | ② |
| ActCpuTimeOut | 40 | Any value specified by user in 250ms units | |
| ActCpuType | 17 (CPU_Q2ACPU) | CPU type corresponding to target station | |
| ActDestinationPortNumber | 1280 (0x500) | Port number of connected station side module | |
| ActHostAddress | 1.1.1.1 | Host name or IP address of connected station side module | |
| ActNetworkNumber | 0 (0x00) | 0x00 | Target station side module network number |
| ActStationNumber | 255 (0xFF) | 0xFF | Target station side module station number |
| ActTimeOut | 10000 | Any value specified by user in ms units | |

## 3.3.5 ActAJ71QE71UDP control

The following table indicates the properties possessed by the ActAJ71QE71UDP control and their default values.

### (1) Configuration




### (2) Property patterns

| Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|
| QnACPU | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | | Q mode | A mode | | | |
| ① | MELSECNET/10H | × | × | × | × | × |
| | MELSECNET/10 | × | × | ② | × | × |
| | MELSECNET(II) | × | × | × | × | × |
| | Ethernet | × | × | ② | × | × |
| | Computer link | × | × | ③ | × | × |
| | CC-Link | × | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU

### (3) Property list

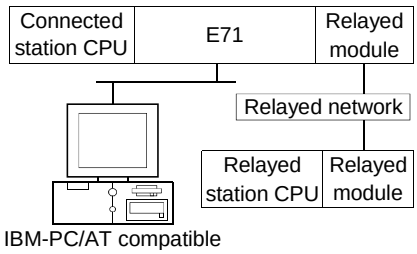
| Property | Default Value | Property Patterns | | |
|---|---|---|---|---|
| | | ① | ② | ③ |
| ActConnectUnitNumber ∗1 | 0 (0x00) | Fixed to 0x00 | Connected station side module station number | Fixed to 0x00 |
| ActCpuType | 17 (CPU_Q2ACPU) | CPU type corresponding to target station | | |
| ActHostAddress | 1.1.1.1 | Host name or IP address of connected station side module | | |
| ActIONumber | 1023 (0x3FF) | Fixed to 0x3FF | Fixed to 0x3FF | Connected station side relayed module I/O address |
| ActNetworkNumber ∗2 | 1 (0x01) | Target station side module network number | Target station side module network number | Connected station side QE71 network number |
| ActPortNumber ∗3 ∗6 | 5001 | IBM-PC/AT compatible side port number | | |
| ActSourceNetworkNumber ∗4 | 1 (0x01) | IBM-PC/AT compatible side network number | | |
| ActSourceStationNumber ∗5 | 2 (0x02) | IBM-PC/AT compatible side station number | | |
| ActStationNumber ∗2 | 1 (0x01) | Target station side module station number | Target station side module station number | Connected station side QE71 station number |
| ActHostAddress | 1.1.1.1 | Host name or IP address of connected station side module | | |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Target station side module station number |

∗1: For access to another station via MELSECNET/10 (for the property pattern of ②), specify the station number of the connected station side QE71 set in the Ethernet parameter of the connected station side QE71.
∗2: For the property pattern of ① or ②, specify the value set in the target station side parameter for ActNetworkNumber and ActStationNumber.
∗3: Specify fixed "5001" when the Ethernet parameter setting of the connected station side QE71 is other than the "automatic response system". Specify fixed "0" when the Ethernet parameter setting of the connected station side QE71 is the "automatic response system".
∗4: Specify the same network number as the MELSECNET/10 network number set to the QE71 in the Ethernet parameter setting of the target station side QE71.
∗5: Specify the station number on the IBM-PC/AT compatible side to avoid setting the same station number as set to the QE71 within the same Ethernet loop.
∗6: Do not use 1 to 1024 of ActPortNumber.

## 3.3.6 ActAJ71E71TCP control

The following table indicates the properties possessed by the ActAJ71E71TCP control and their default values.

### (1) Configuration




### (2) Property patterns

| Connected Station CPU | | | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|---|
| QCPU (A mode) | QnA CPU | ACPU ＊1 | | QCPU | | QnA CPU | ACPU ＊1 | FXCPU |
| | | | | Q mode | A mode | | | |
| ① | ①＊2 | ① | MELSECNET/10H | × | × | × | × | × |
| | | | MELSECNET/10 | × | ② | ②＊2 | ② | × |
| | | | MELSECNET(II) | × | ② | ②＊2 | ② | × |
| | | | Ethernet | × | × | × | × | × |
| | | | Computer link | × | × | × | × | × |
| | | | CC-Link | × | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
＊1 : Including motion controller CPU
＊2 : Operates as the one equivalent to AnACPU.

### (3) Property list

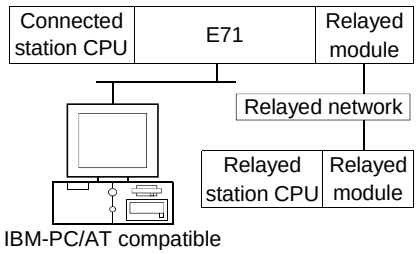
| Property | Default Value | Property Patterns | |
|---|---|---|---|
| | | ① | ② |
| ActCpuTimeOut | 40 | Any value specified by user in 250ms units | |
| ActCpuType | 262 (CPU_A1NCPU) | CPU type corresponding to target station | |
| ActDestinationPortNumber | 1280 (0x500) | Port number of connected station side module | |
| ActHostAddress | 1.1.1.1 | Host name or IP address of connected station side module | |
| ActStationNumber＊1 | 255 (0xFF) | Fixed to 0xFF | Target station side module station number |
| ActTimeOut | 10000 | Any value specified by user in ms units | |

＊1: Note the following points depending on whether the connected station side MELSECNET/10 module is the control station or ordinary station.
When the connected station side MELSECNET/10 module is the control station..... Specify the actual station number of the target station side MELSECNET/10 module in ActStationNumber.
When the connected station side MELSECNET/10 module is the ordinary station.... Always set the target station side MELSECNET/10 module as the control station and specify "0x00" in ActStationNumber.

## 3.3.7 ActAJ71E71UDP control

The following table indicates the properties possessed by the ActAJ71E71UDP control and their default values.

### (1) Configuration




### (2) Property patterns

| Connected Station CPU | | | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|---|
| QCPU (A mode) | QnA CPU | ACPU ＊1 | | QCPU | | QnA CPU | ACPU ＊1 | FXCPU |
| | | | | Q mode | A mode | | | |
| ① | ①＊2 | ① | MELSECNET/10H | × | × | × | × | × |
| | | | MELSECNET/10 | × | ② | ②＊2 | ② | × |
| | | | MELSECNET(II) | × | ② | ②＊2 | ② | × |
| | | | Ethernet | × | × | × | × | × |
| | | | Computer link | × | × | × | × | × |
| | | | CC-Link | × | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
＊1 : Including motion controller CPU
＊2 : Operates as the one equivalent to AnACPU.

### (3) Property list

| Property | Default Value | Property Patterns | |
|---|---|---|---|
| | | ① | ② |
| ActCpuTimeOut | 40 | Any value specified by user in 250ms units | |
| ActCpuType | 262 (CPU_A1NCPU) | CPU type corresponding to target station | |
| ActDestinationPortNumber | 1280 (0x500) | Port number of connected station side module | |
| ActHostAddress | 1.1.1.1 | Host name or IP address of connected station side module | |
| ActPortNumber＊1 | 0 | IBM-PC/AT compatible side port number | |
| ActStationNumber＊2 | 255 (0xFF) | Fixed to 0xFF | Target station side module station number |
| ActTimeOut | 10000 | Any value specified by user in ms units | |

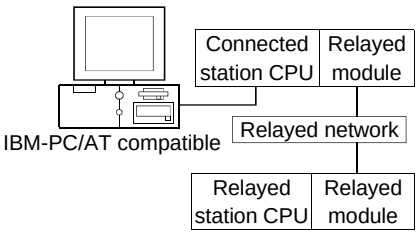
＊1: 0 ..... The free port number of the IBM-PC/AT compatible is assigned automatically.
Other than 0 ..... The specified port number is used to generate the UDP socket.
Do not use 1 to 1024 of ActPortNumber.

＊2: Note the following points depending on whether the connected station side MELSECNET/10 module is the control station or ordinary station.
When the connected station side MELSECNET/10 module is the control station ... Specify the actual station number of the target station side MELSECNET/10 module in ActStationNumber.
When the connected station side MELSECNET/10 module is the ordinary station ... Always set the target station side MELSECNET/10 module as the control station and specify "0x00" in ActStationNumber.

## 3.3.8 ActQCPUQ control

The following table indicates the properties possessed by the ActQCPUQ control and their default values.

### (1) Configuration




### (2) Property patterns

| Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|
| QCPU (Q mode) | | QCPU | | QnA CPU | ACPU ＊1 | FXCPU |
| | | Q mode | A mode | | | |
| ① | MELSECNET/10H | ② | × | × | × | × |
| | MELSECNET/10 | ② | ② | ② | ② | × |
| | MELSECNET(II) | × | × | × | × | × |
| | Ethernet | ② | × | ② | × | × |
| | Computer link | ③ | × | ③ | × | × |
| | CC-Link | ④ | ④＊2 | ④＊2 | ④＊2 | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
＊1 : Including motion controller CPU
＊2 : Use the QnA or ACPU side CC-Link module whose ROM version is "S" or later.

### (3) Property list

| Property | Default Value | Property Patterns | | | |
|---|---|---|---|---|---|
| | | ① | ②＊2 | ③ | ④ |
| ActBaudRate | 19200 (BAUDRATE_19200) | BAUDRATE_9600, BAUDRATE_19200, BAUDRATE_38400, BAUDRATE_57600, BAUDRATE_115200 | | | |
| ActControl | 8 (TCR_DTR_OR_RTS) | Depending on used cable. | | | |
| ActCpuType | 34 (CPU_Q02CPU) | CPU type corresponding to target station | | | |
| ActDestinationIONumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Fixed to 0x3FF | Fixed to 0x3FF |
| ActDidPropertyBit | 1 (0x01) | 0x01 (invalid) | 0x01 (invalid) | 0x00 (valid) | 0x00 (valid) |
| ActDisdPropertyBit | 1 (0x01) | 0x01 (invalid) | 0x01 (invalid) | 0x00 (valid) | 0x00 (valid) |
| ActIntelligentPreferenceBit | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | 0x01 (target station is QCPU (Q mode), 0x00 (target station is other than QCPU (Q mode)) | 0x01 (target station is QCPU (Q mode), 0x00 (target station is other than QCPU (Q mode)) |
| ActIONumber＊1 | 1023 (0x3FF) | Fixed to 0x3FF | Fixed to 0x3FF | Connected station side module I/O address | Connected station side module I/O address |
| ActMultiDropChannelNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Fixed to 0x02 | Fixed to 0x02 |

＊1: As the I/O address, specify the value found by dividing the actual first I/O number by 16.

＊2: Note the following points when making access via the Ethernet module (Q series-compatible E71, QE71).

- For ActNetworkNumber and ActStationNumber, specify the value set in the parameter setting of the target station side Q series-compatible E71 or QE71.
- Set the "MNET/10 routing information" in the parameter setting of the Q series-compatible E71 or QE71. Also, when making setting, specify other than the automatic response system (any of the IP address calculation system, table conversion system and combined system) as the "MNET/10 routing system".

| Property | Default Value | Property Patterns | | | |
|---|---|---|---|---|---|
| | | ① | ②＊2 | ③ | ④ |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Target station side module network number | Fixed to 0x00 | Fixed to 0x00 |
| ActPortNumber | 1 (PORT_1) | IBM-PC/AT compatible side COM port number | | | |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side module station number | Fixed to 0xFF | Fixed to 0xFF |
| ActThroughNetworkType | 0 (0x00) | QCPU (Q mode): 0x00 (MELSECNET/10H only), other than QCPU (Q mode): 0x01 (including MELSECNET/10). Note that the setting must be the same as set in the network parameter of the GPP function. | | | |
| ActTimeOut | 10000 | Any value specified by user in ms units | | | |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Target station side module station number | Target station side module station number |

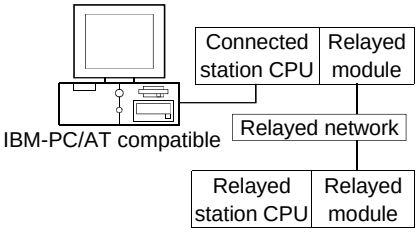
＊2: Note the following points when making access via the Ethernet module (Q series-compatible E71, QE71).

- For ActNetworkNumber and ActStationNumber, specify the value set in the parameter setting of the target station side Q series-compatible E71 or QE71.
- Set the "MNET/10 routing information" in the parameter setting of the Q series-compatible E71 or QE71. Also, when making setting, specify other than the automatic response system (any of the IP address calculation system, table conversion system and combined system) as the "MNET/10 routing system".

## 3.3.9 ActQCPUA control

The following table indicates the properties possessed by the ActQCPUA control and their default values.

### (1) Configuration




### (2) Property patterns

| Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|
| QCPU (A mode) | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | | Q mode | A mode | | | |
| ① | MELSECNET/10H | × | × | × | × | × |
| | MELSECNET/10 | × | ② | × | ② | × |
| | MELSECNET(II) | × | ③ | × | ③ | × |
| | Ethernet | × | × | × | × | × |
| | Computer link | × | × | × | × | × |
| | CC-Link | × | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU

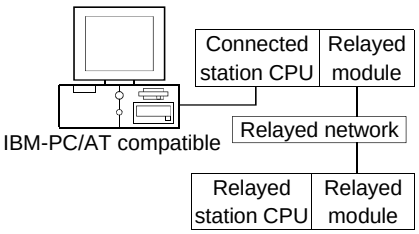
### (3) Property list

| Property | Default Value | Property Patterns | | |
|---|---|---|---|---|
| | | ① | ② | ③ |
| ActBaudRate | 9600 (BAUDRATE_9600) | BAUDRATE_9600, BAUDRATE_19200, BAUDRATE_38400, BAUDRATE_57600, BAUDRATE_115200 | | |
| ActControl | 8 (TCR_DTR_OR_RTS) | Depending on used cable. | | |
| ActCpuType | 321 (CPU_Q02CPU_A) | CPU type corresponding to target station | | |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Target station side module network number | Fixed to 0x00 |
| ActPortNumber | 1 (PORT_1) | IBM-PC/AT compatible side COM port number | | |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side module station number | Target station side module station number |
| ActTimeOut | 10000 | Any value specified by user in ms units. | | |

## 3.3.10 ActQnACPU control

The following table indicates the properties possessed by the ActQnACPU control and their default values.

### (1) Configuration




### (2) Property patterns

| Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|
| QnACPU | | QCPU | | QnA CPU | ACPU ＊1 | FXCPU |
| | | Q mode | A mode | | | |
| ① | MELSECNET/10H | × | × | × | × | × |
| | MELSECNET/10 | × | × | ② | × | × |
| | MELSECNET(II) | × | × | ③ | × | × |
| | Ethernet | × | × | ② | × | × |
| | Computer link | × | × | ④ | × | × |
| | CC-Link | × | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
＊1 : Including motion controller CPU

### (3) Property list

| Property | Default Value | Property Patterns | | | |
|---|---|---|---|---|---|
| | | ① | ②＊2 | ③ | ④ |
| ActBaudRate | 19200 (BAUDRATE_19200) | BAUDRATE_9600, BAUDRATE_19200, BAUDRATE_38400＊3 | | | |
| ActControl | 8 (TCR_DTR_OR_RTS) | Depending on used cable. | | | |
| ActCpuType | 17 (CPU_Q2ACPU) | CPU type corresponding to target station | | | |
| ActIONumber＊1 | 1023 (0x3FF) | Fixed to 0x3FF | Fixed to 0x3FF | Fixed to 0x3FF | Connected station side module I/O address |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Target station side module network number | Fixed to 0x00 | Fixed to 0x00 |
| ActPortNumber | 1 (PORT_1) | IBM-PC/AT compatible side COM port number | | | |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side module station number | Target station side module station number | Fixed to 0xFF |
| ActTimeOut | 10000 | Any value specified by user in ms units. | | | |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Fixed to 0x00 | Target station side module station number |

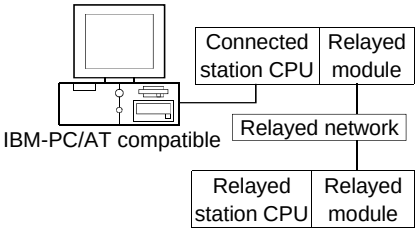
＊1: As the I/O address, specify the value found by dividing the actual first I/O number by 16.
＊2: Note the following points when making access via the Ethernet module (QE71).
- For ActNetworkNumber and ActStationNumber, specify the value set in the parameter setting of the target station side QE71.
- Set the "MNET/10 routing information" in the parameter setting of the QE71. Also, when making setting, specify other than the automatic response system (any of the IP address calculation system, table conversion system and combined system) as the "MNET/10 routing system".

＊3: Usable for only the QnACPU version 9707B or later.

## 3.3.11 ActACPU control

The following table indicates the properties possessed by the ActACPU control and their default values.

### (1) Configuration




### (2) Property patterns

| Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|
| ACPU＊1 | | QCPU | | QnA CPU | ACPU ＊1 | FXCPU |
| | | Q mode | A mode | | | |
| ① | MELSECNET/10H | × | × | × | × | × |
| | MELSECNET/10 | × | ② | × | ② | × |
| | MELSECNET(II) | × | ③ | × | ③ | × |
| | Ethernet | × | × | × | × | × |
| | Computer link | × | × | × | × | × |
| | CC-Link | × | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
＊1 : Including motion controller CPU

### (3) Property list

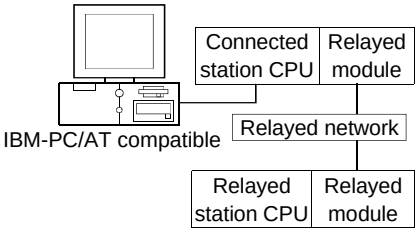
| Property | Default Value | Property Patterns | | |
|---|---|---|---|---|
| | | ① | ② | ③ |
| ActBaudRate | 9600 (BAUDRATE_9600) | Fixed to BAUDRATE_9600＊1 | | |
| ActControl | 8 (TCR_DTR_OR_RTS) | Depending on used cable. | | |
| ActCpuType | 262 (CPU_A1NCPU) | CPU type corresponding to target station | | |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Target station side module network number | Fixed to 0x00 |
| ActPortNumber | 1 (PORT_1) | IBM-PC/AT compatible side COM port number | | |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side module station number | Target station side module station number |
| ActTimeOut | 10000 | Any value specified by user in ms units. | | |

＊1: BAUDRATE_9600 may be used only when the connected station CPU is the A2USHCPU-S1.

## 3.3.12 ActFXCPU control

The following table indicates the properties possessed by the ActFXCPU control and their default values.

### (1) Configuration




### (2) Property patterns

| Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|
| FXCPU | | QCPU | | QnA CPU | ACPU ＊1 | FXCPU |
| | | Q mode | A mode | | | |
| ① | MELSECNET/10H | × | × | × | × | × |
| | MELSECNET/10 | × | × | × | × | × |
| | MELSECNET(II) | × | × | × | × | × |
| | Ethernet | × | × | × | × | × |
| | Computer link | × | × | × | × | × |
| | CC-Link | × | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
＊1 : Including motion controller CPU

### (3) Property list

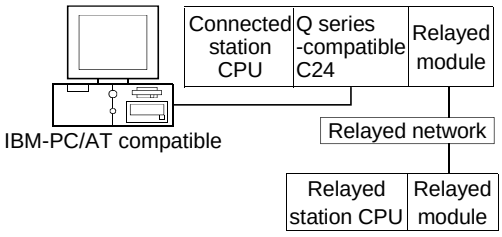
| Property | Default Value | Property Patterns |
|---|---|---|
| | | ① |
| ActControl | 8 (TCR_DTR_OR_RTS) | Depending on used cable. |
| ActCpuType | 513 (CPU_FX0CPU) | CPU type corresponding to target station |
| ActPortNumber | 1 (PORT_1) | IBM-PC/AT compatible side COM port number |
| ActTimeOut | 10000 | Any value specified by user in ms units. |

## 3.3.13 ActQJ71C24 control

The following table indicates the properties possessed by the ActQJ71C24 control and their default values.

(1) When there is relayed module in addition to connected station side Q series-compatible C24

(a) Configuration




(b) Property patterns

| Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|
| QCPU (Q mode) | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | | Q mode | A mode | | | |
| ① | MELSECNET/10H | ② | × | × | × | × |
| | MELSECNET/10 | ② | ② | ② | ② | × |
| | MELSECNET(II) | × | × | × | × | × |
| | Ethernet | ② | × | ② | × | × |
| | Computer link | ③ | × | ③ | × | × |
| | CC-Link | ④ | ④ | ④ | ④ | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU

(c) Property list

| Property | Default Value | Property Patterns | | | |
|---|---|---|---|---|---|
| | | ① | ②∗2 | ③ | ④ |
| ActBaudRate | 19200 (BAUDRATE_19200) | Match to the setting of Q series-compatible C24. | | | |
| ActConnectUnitNumber | 0 (0x00) | Connected station side module station number | | | |
| ActControl | 8 (TCR_DTR_OR_RTS) | Depending on used cable. | | | |
| ActCpuType | 34 (CPU_Q02CPU) | CPU type corresponding to target station | | | |
| ActDestinationIONumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | 0x3FF | 0x3FF |
| ActDidPropertyBit | 1 (0x01) | 0x01 (invalid) | 0x01 (invalid) | 0x00 (valid) | 0x00 (valid) |
| ActDisdPropertyBit | 1 (0x01) | 0x01 (invalid) | 0x01 (invalid) | 0x00 (valid) | 0x00 (valid) |
| ActIntelligentPreferenceBit | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Fixed to 0x00 | Fixed to 0x00 |
| ActIONumber∗1 | 1023 (0x3FF) | Fixed to 0x3FF | Fixed to 0x3FF | Connected station side module I/O address | Connected station side module I/O address |
| ActMultiDropChannelNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Fixed to 0x02 | Fixed to 0x00 |

∗1: As the I/O address, specify the value found by dividing the actual first I/O number by 16.
∗2: Note the following points when making access via the Ethernet module (Q series-compatible E71, QE71).
- For ActNetworkNumber and ActStationNumber, specify the value set in the parameter setting of the target station side Q series-compatible E71 or QE71.
- Set the "MNET/10 routing information" in the parameter setting of the Q series-compatible E71 or QE71. Also, when making setting, specify other than the automatic response system (any of the IP address calculation system, table conversion system and combined system) as the "MNET/10 routing system".

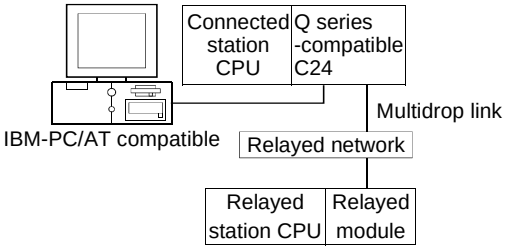
| Property | Default Value | Property Patterns | | | |
|---|---|---|---|---|---|
| | | ① | ②＊2 | ③ | ④ |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Target station side module network number | Fixed to 0x00 | Fixed to 0x00 |
| ActParity | 1 (ODD_PARITY) | Match to the setting of Q series-compatible C24. | | | |
| ActPortNumber | 1 (PORT_1) | IBM-PC/AT compatible side COM port number | | | |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side module station number | Fixed to 0xFF | Fixed to 0xFF |
| ActThroughNetworkType | 0 (0x00) | QCPU (Q mode): 0x00 (MELSECNET/10H only), other than QCPU (Q mode): 0x01 (including MELSECNET/10). Note that the setting must be the same as set in the network parameter of the GPP function. | | | |
| ActTimeOut | 10000 | Any value specified by user in ms units | | | |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Target station side module station number | Target station side module station number |

＊2: Note the following points when making access via the Ethernet module (Q series-compatible E71, QE71).

- For ActNetworkNumber and ActStationNumber, specify the value set in the parameter setting of the target station side Q series-compatible E71 or QE71.
- Set the "MNET/10 routing information" in the parameter setting of the Q series-compatible E71 or QE71. Also, when making setting, specify other than the automatic response system (any of the IP address calculation system, table conversion system and combined system) as the "MNET/10 routing system".

### (2) When connected station side Q series-compatible C24 is used for multidrop link with relayed module

#### (a) Configuration




#### (b) Property patterns

| Connected Station CPU | | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|
| QCPU (Q mode) | | | QCPU | | QnA CPU | ACPU ＊1 | FXCPU |
| | | | Q mode | A mode | | | |
| Independent mode ＊2 | ① | Computer link | ② | × | ② | × | × |
| Synchronous mode ＊2 | | | ③ | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
＊1 : Including motion controller CPU
＊2 : Indicates the CH2 side setting (CH1 side fixed to independent mode)

#### (c) Property list

| Property | Default Value | Property Patterns | | |
|---|---|---|---|---|
| | | ① | ② | ③ |
| ActBaudRate | 19200 (BAUDRATE_19200) | Match to the setting of Q series-compatible C24. | | |
| ActConnectUnitNumber | 0 (0x00) | Connected station side module station number | | |
| ActControl | 8 (TCR_DTR_OR_RTS) | Depending on used cable. | | |
| ActCpuType | 34 (CPU_Q02CPU) | CPU type corresponding to target station | | |
| ActDestinationIONumber | 0 (0x00) | Fixed to 0x00 | 0x3FF | Fixed to 0x00 |
| ActDidPropertyBit | 1 (0x01) | 0x01 (invalid) | 0x00 (valid) | 0x01 (invalid) |
| ActDsidPropertyBit | 1 (0x01) | 0x01 (invalid) | 0x00 (valid) | 0x01 (invalid) |
| ActIntelligentPreferenceBit | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Fixed to 0x00 |
| ActIONumber＊1 | 1023 (0x3FF) | Fixed to 0x3FF | Connected station side module I/O address | Fixed to 0x3FF |
| ActMultiDropChannelNumber | 0 (0x00) | Fixed to 0x00 | Multidrop channel number | Fixed to 0x00 |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Target station side module network number | Fixed to 0x00 |
| ActParity | 1 (ODD_PARITY) | Match to the setting of Q series-compatible C24. | | |
| ActPortNumber | 1 (PORT_1) | IBM-PC/AT compatible side COM port number | | |
| ActStationNumber | 255 (0xFF) | Fixed to 0x0FF | Fixed to 0x0FF | Fixed to 0x0FF |
| ActThroughNetworkType | 0 (0x00) | QCPU (Q mode): 0x00 (MELSECNET/10H only), other than QCPU (Q mode): 0x01 (including MELSECNET/10). Note that the setting must be the same as set in the network parameter of the GPP function. | | |

＊1: As the I/O address, specify the value found by dividing the actual first I/O number by 16.

| Property | Default Value | Property Patterns | | |
|---|---|---|---|---|
| | | ① | ② | ③ |
| ActTimeOut | 10000 | Any value specified by user in ms units | | |
| ActUnitNumber | 0<br>(0x00) | Fixed to 0x00 | Target station side module station number | Fixed to 0x00 |

**POINT**

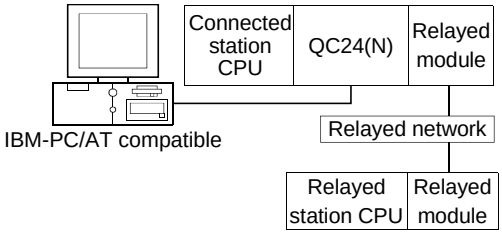
When the connected station side Q series-compatible C24 is set to the synchronous mode, always set the "sumcheck (SW06)" transmission specification software switch setting of the Q series-compatible C24 parameters to Yes (ON). If it is set to No (OFF), a communication error will occur, disabling proper communication.

## 3.3.14 ActAJ71QC24 control

The following table indicates the properties possessed by the ActAJ71QC24 control and their default values.

(1) When there is relayed module in addition to connected station side QC24(N)

(a) Configuration




(b) Property patterns

| Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|
| QnACPU | | QCPU | | QnA CPU | ACPU ＊1 | FXCPU |
| | | Q mode | A mode | | | |
| ① | MELSECNET/10H | × | × | × | × | × |
| | MELSECNET/10 | × | × | ② | × | × |
| | MELSECNET(II) | × | × | ③ | × | × |
| | Ethernet | × | × | ② | × | × |
| | Computer link | × | × | ④ | × | × |
| | CC-Link | × | × | ④ | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
＊1 : Including motion controller CPU

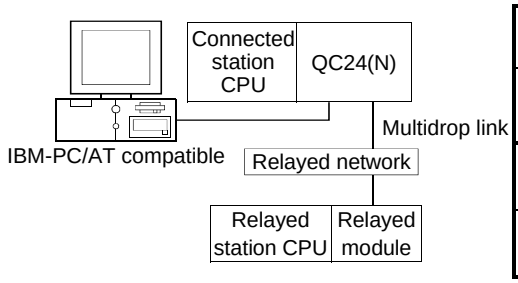
(c) Property list

| Property | Default Value | Property Patterns | | | |
|---|---|---|---|---|---|
| | | ① | ②＊2 | ③ | ④ |
| ActBaudRate | 19200 (BAUDRATE_19200) | Match to the setting of QC24(N). | | | |
| ActConnectUnitNumber | 0 (0x00) | Connected station side module station number | | | |
| ActControl | 8 (TCR_DTR_OR_RTS) | Depending on used cable. | | | |
| ActCpuType | 17 (CPU_Q2ACPU) | CPU type corresponding to target station | | | |
| ActIONumber＊1 | 1023 (0x3FF) | Fixed to 0x3FF | Fixed to 0x3FF | Fixed to 0x3FF | Connected station side module I/O address |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Target station side module network number | Fixed to 0x00 | Fixed to 0x00 |
| ActParity | 1 (ODD_PARITY) | Match to the setting of QC24(N). | | | |
| ActPortNumber | 1 (PORT_1) | IBM-PC/AT compatible side COM port number | | | |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side module station number | Target station side module station number | Fixed to 0xFF |
| ActTimeOut | 10000 | Any value specified by user in ms units | | | |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Fixed to 0x00 | Target station side module station number |

＊1: As the I/O address, specify the value found by dividing the actual first I/O number by 16.
＊2: Note the following points when making access via the Ethernet module (QE71).
- For ActNetworkNumber and ActStationNumber, specify the value set in the parameter setting of the target station side QE71.
- Set the "MNET/10 routing information" in the parameter setting of the QE71. Also, when making setting, specify other than the automatic response system (any of the IP address calculation system, table conversion system and combined system) as the "MNET/10 routing system".

## (2) When connected station side QC24(N) is used for multidrop link with relayed module

### (a) Configuration




### (b) Property patterns

| Connected Station CPU | | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|
| QnACPU | | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | | | Q mode | A mode | | | |
| Independent mode ∗2 | ① | Computer link | × | × | ② | × | × |
| Synchronous mode ∗2 | | | × | × | ③ | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU
∗2 : Indicates the CH2 side setting (CH1 side fixed to independent mode)

### (c) Property list

| Property | Default Value | Property Patterns | | |
|---|---|---|---|---|
| | | ① | ② | ③ |
| ActBaudRate | 19200 (BAUDRATE_19200) | Match to the setting of QC24(N). | | |
| ActConnectUnitNumber | 0 (0x00) | Connected station side module station number | | |
| ActControl | 8 (TCR_DTR_OR_RTS) | Depending on used cable. | | |
| ActCpuType | 17 (CPU_Q2ACPU) | CPU type corresponding to target station | | |
| ActIONumber ∗1 | 1023 (0x3FF) | Fixed to 0x3FF | Connected station side module I/O address | Fixed to 0x3FF |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Target station side module network number | Fixed to 0x00 |
| ActParity | 1 (ODD_PARITY) | Match to the setting of QC24(N). | | |
| ActPortNumber | 1 (PORT_1) | IBM-PC/AT compatible side COM port number | | |
| ActStationNumber | 255 (0xFF) | Fixed to 0x0FF | Fixed to 0x0FF | Fixed to 0x0FF |
| ActTimeOut | 10000 | Any value specified by user in ms units | | |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | Target station side module station number | Fixed to 0x00 |

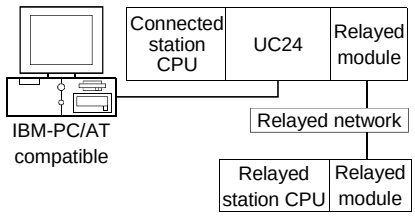
∗1: As the I/O address, specify the value found by dividing the actual first I/O number by 16.

## 3.3.15 ActAJ71UC24 control

The following table indicates the properties possessed by the ActAJ71UC24 control and their default values.

### (1) When there is relayed module in addition to connected station side UC24

#### (a) Configuration




#### (b) Property patterns

| Connected Station CPU | | | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|---|
| QCPU (A mode) | QnA CPU | ACPU ＊1 | | QCPU | | QnA CPU | ACPU ＊1 | FXCPU |
| | | | | Q mode | A mode | | | |
| ① | ①＊2 | ① | MELSECNET/10H | × | × | × | × | × |
| | | | MELSECNET/10 | × | ② | ②＊2 | ② | × |
| | | | MELSECNET(II) | × | ③ | ③＊2 | ③ | × |
| | | | Ethernet | × | × | × | × | × |
| | | | Computer link | × | × | × | × | × |
| | | | CC-Link | × | × | × | × | × |

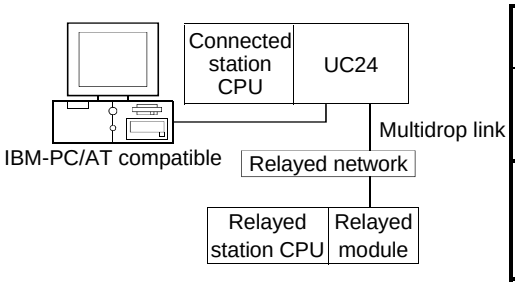
○ : Accessible (Property pattern within circle)
× : Inaccessible
＊1 : Including motion controller CPU
＊2 : Operates as the one eqSuivalent to AnACPU.

#### (c) Property list

| Property | Default Value | Property Patterns | | |
|---|---|---|---|---|
| | | ① | ② | ③ |
| ActBaudRate | 19200 (BAUDRATE_19200) | Match to the setting of UC24. | | |
| ActControl | 8 (TCR_DTR_OR_RTS) | Depending on used cable. | | |
| ActCpuType | 262 (CPU_A1NCPU) | CPU type corresponding to target station | | |
| ActDataBits | 8 (DATABIT_8) | Match to the setting of UC24. | | |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Target station side module network number | Fixed to 0x00 |
| ActParity | 1 (ODD_PARITY) | Match to the setting of UC24. | | |
| ActPortNumber | 1 (PORT_1) | IBM-PC/AT compatible side COM port number | | |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side module station number | Target station side module station number |
| ActStopBits | 0 (STOPBIT_ONE) | Match to the setting of UC24. | | |
| ActSumCheck | 1 (SUM_CHECK) | Match to the setting of UC24. | | |
| ActTimeOut | 10000 | Any value specified by user in ms units | | |
| ActUnitNumber | 0 (0x00) | Target station side module station number | Connected station side module station number | Connected station side module station number |

### (2) When connected station side UC24 is used for multidrop link with relayed module

#### (a) Configuration




#### (b) Property patterns

| Connected Station CPU | | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|
| QCPU (A mode), QnACPU ∗3, ACPU ∗1 | | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | | | Q mode | A mode | | | |
| Independent mode ∗2 | ① | Computer link | × | ② | ② ∗3 | ② | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU
∗2 : Use the mode setting switch and main channel setting to make setting.
∗3 : Operates as the one equivalent to AnACPU.

#### (c) Property list

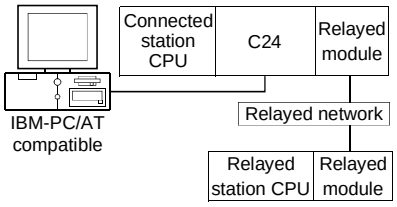
| Property | Default Value | Property Patterns |
|---|---|---|
| | | ① |
| ActBaudRate | 19200 (BAUDRATE_19200) | Match to the setting of UC24. |
| ActControl | 8 (TCR_DTR_OR_RTS) | Depending on used cable. |
| ActCpuType | 262 (CPU_A1NCPU) | CPU type corresponding to target station |
| ActDataBits | 8 (DATABIT_8) | Match to the setting of UC24. |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 |
| ActParity | 1 (ODD_PARITY) | Match to the setting of UC24. |
| ActPortNumber | 1 (PORT_1) | IBM-PC/AT compatible side COM port number |
| ActStationNumber | 255 (0xFF) | Fixed to 0x0FF |
| ActStopBits | 0 (STOPBIT_ONE) | Match to the setting of UC24. |
| ActSumCheck | 1 (SUM_CHECK) | Match to the setting of UC24. |
| ActTimeOut | 10000 | Any value specified by user in ms units |
| ActUnitNumber | 0 (0x00) | Target station side module station number |

## 3.3.16 ActAJ71C24 control

The following table indicates the properties possessed by the ActAJ71C24 control and their default values.

### (1) When there is relayed module in addition to connected station side C24

(a) Configuration




(b) Property patterns

| Connected Station CPU | | | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|---|
| QCPU (A mode) | QnA CPU | ACPU ∗1 | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | | | | Q mode | A mode | | | |
| ① | ①∗2 | ① | MELSECNET/10H | × | × | × | × | × |
| | | | MELSECNET/10 | × | ② | ②∗2 | ② | × |
| | | | MELSECNET(II) | × | ② | ②∗2 | ② | × |
| | | | Ethernet | × | × | × | × | × |
| | | | Computer link | × | × | × | × | × |
| | | | CC-Link | × | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU
∗2 : Operates as the one equivalent to AnACPU.

(c) Property list

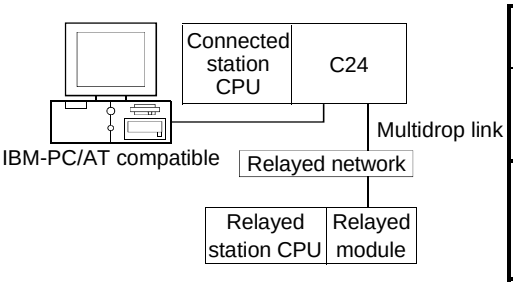
| Property | Default Value | Property Patterns | |
|---|---|---|---|
| | | ① | ② ∗2 |
| ActBaudRate | 19200 (BAUDRATE _19200) | Match to the setting of C24. | |
| ActControl | 8 (TCR_DTR_OR _RTS) | Depending on used cable. | |
| ActCpuType | 262 (CPU_A1NCPU) | CPU type corresponding to target station | |
| ActDataBits | 8 (DATABIT_8) | Match to the setting of C24. | |
| ActParity | 1 (ODD_PARITY) | Match to the setting of C24. | |
| ActPortNumber | 1 (PORT_1) | IBM-PC/AT compatible side COM port number | |
| ActStationNumber ∗1 | 255 (0xFF) | Fixed to 0xFF | Target station side module station number |
| ActStopBits | 0 (STOPBIT _ONE) | Match to the setting of C24. | |
| ActSumCheck | 1 (SUM_CHECK) | Match to the setting of C24. | |
| ActTimeOut | 10000 | Any value specified by user in ms units | |
| ActUnitNumebr | 0 (0x00) | Target station side module station number | Connected station side module station number |

∗1: Note the following points depending on whether the connected station side MELSECNET/10 module is the control station or ordinary station.
When the connected station side MELSECNET/10 module is the control station... Specify the actual station number of the target station side MELSECNET/10 module in ActStationNumber.
When the connected station side MELSECNET/10 module is the ordinary station... Always set the target station side MELSECNET/10 module as the control station and specify "0x00" in ActStationNumber.

∗2: Access via network is enabled only to the network on the side specified in "valid module for another station access" in the connected station side network parameters.

### (2) When connected station side C24 is used for multidrop link with relayed module

#### (a) Configuration




#### (b) Property patterns

| Connected Station CPU | | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|
| QCPU (A mode), QnACPU ∗3, ACPU ∗1 | | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | | | Q mode | A mode | | | |
| Independent mode ∗2 | ① | Computer link | × | ① | ①∗3 | ① | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU
∗2 : Use the mode setting switch and main channel setting to make setting.
∗3 : Operates as the one equivalent to AnACPU.

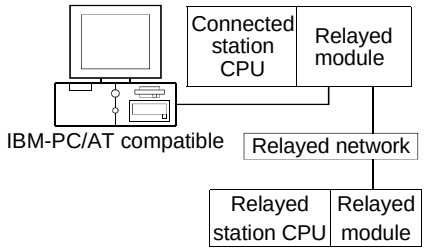
#### (c) Property list

| Property | Default Value | Property Patterns |
|---|---|---|
| | | ① |
| ActBaudRate | 19200 (BAUDRATE_19200) | Match to the setting of C24. |
| ActControl | 8 (TCR_DTR_OR_RTS) | Depending on used cable. |
| ActCpuType | 262 (CPU_A1NCPU) | CPU type corresponding to target station |
| ActDataBits | 8 (DATABIT_8) | Match to the setting of C24. |
| ActParity | 1 (ODD_PARITY) | Match to the setting of C24. |
| ActPortNumber | 1 (PORT_1) | IBM-PC/AT compatible side COM port number |
| ActStationNumber | 255 (0xFF) | Fixed to 0x0FF |
| ActStopBits | 0 (STOPBIT_ONE) | Match to the setting of C24. |
| ActSumCheck | 1 (SUM_CHECK) | Match to the setting of C24. |
| ActTimeOut | 10000 | Any value specified by user in ms units |
| ActUnitNumebr | 0 (0x00) | Target station side module station number |

## 3.3.17 ActQCPUQUSB control

The following table indicates the properties possessed by the ActQCPUQUSB control and their default values.

### (1) Configuration




### (2) Property patterns

| Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|
| QCPU (Q mode) | | QCPU | | QnA CPU | ACPU ＊1 | FXCPU |
| | | Q mode | A mode | | | |
| ① | MELSECNET/10H | ② | × | × | × | × |
| | MELSECNET/10 | ② | ② | ② | ② | × |
| | MELSECNET(II) | × | × | × | × | × |
| | Ethernet | ② | × | ② | × | × |
| | Computer link | ③ | × | ③ | × | × |
| | CC-Link | ④ | ④＊2 | ④＊2 | ④＊2 | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
＊1 : Including motion controller CPU
＊2 : *2: Use the QnA or ACPU side CC-Link module whose ROM version is "S" or later.

### (3) Property list

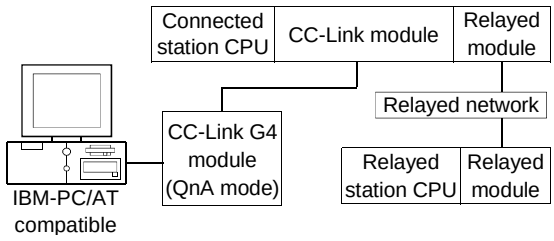
| Property | Default Value | Property Patterns | | | |
|---|---|---|---|---|---|
| | | ① | ②＊2 | ③ | ④ |
| ActCpuType | 34 (CPU_Q02CPU) | CPU type corresponding to target station | | | |
| ActDestinationIONumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Fixed to 0x3FF | Fixed to 0x3FF |
| ActDidPropertyBit | 1 (0x01) | 0x01 (invalid) | 0x01 (invalid) | 0x00 (valid) | 0x00 (valid) |
| ActDisdPropertyBit | 1 (0x01) | 0x01 (invalid) | 0x01 (invalid) | 0x00 (valid) | 0x00 (valid) |
| ActIntelligentPreferenceBit | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | 0x01 (target station is QCPU (Q mode), 0x00 (target station is other than QCPU (Q mode)) | 0x01 (target station is QCPU (Q mode), 0x00 (target station is other than QCPU (Q mode)) |
| ActIONumber＊1 | 1023 (0x3FF) | Fixed to 0x3FF | Fixed to 0x3FF | Connected station side module I/O address | Connected station side module I/O address |
| ActMultiDropChannelNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Fixed to 0x02 | Fixed to 0x00 |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Target station side module network number | Fixed to 0x00 | Fixed to 0x00 |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side module station number | Fixed to 0xFF | Fixed to 0xFF |
| ActThroughNetworkType | 0 (0x00) | QCPU (Q mode): 0x00 (MELSECNET/10H only), other than QCPU (Q mode): 0x01 (including MELSECNET/10). Note that the setting must be the same as set in the network parameter of the GPP function. | | | |
| ActTimeOut | 10000 | Any value specified by user in ms units | | | |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Target station side module station number | Target station side module station number |

＊1: As the I/O address, specify the value found by dividing the actual first I/O number by 16.
＊2: Note the following points when making access via the Ethernet module (Q series-compatible E71, QE71).
- For ActNetworkNumber and ActStationNumber, specify the value set in the parameter setting of the target station side Q series-compatible E71 or QE71.
- Set the "MNET/10 routing information" in the parameter setting of the Q series-compatible E71 or QE71. Also, when making setting, specify other than the automatic response system (any of the IP address calculation system, table conversion system and combined system) as the "MNET/10 routing system".

## 3.3.18 ActCCG4QnA control

The following table indicates the properties possessed by the ActCCG4QnA control and their default values.

### (1) Configuration




### (2) Property patterns

| Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|
| QnACPU | | QCPU | | QnA CPU | ACPU ＊1 | FXCPU |
| | | Q mode | A mode | | | |
| ① | MELSECNET/10H | × | × | × | × | × |
| | MELSECNET/10 | × | × | ② | × | × |
| | MELSECNET(II) | × | × | ③ | × | × |
| | Ethernet | × | × | ② | × | × |
| | Computer link | × | × | ④ | × | × |
| | CC-Link | × | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
＊1 : Including motion controller CPU

### (3) Property list

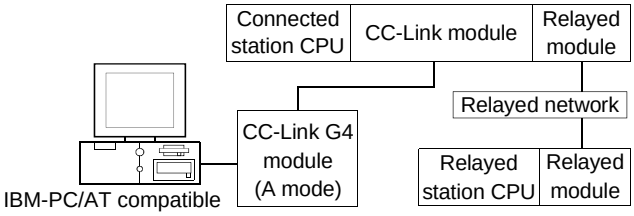
| Property | Default Value | Property Patterns | | | |
|---|---|---|---|---|---|
| | | ① | ②＊2 | ③ | ④ |
| ActBaudRate | 19200 (BAUDRATE_19200) | Match to the setting of CC-Link G4 module. | | | |
| ActConnectUnitNumber | 0 (0x00) | Connected station side CC-Link module station number | | | |
| ActControl | 8 (TCR_DTR_OR_RTS) | Depending on used cable. | | | |
| ActCpuType | 17 (CPU_Q2ACPU) | CPU type corresponding to target station | | | |
| ActIONumber＊1 | 1023 (0x3FF) | Fixed to 0x3FF | Fixed to 0x3FF | Fixed to 0x3FF | Connected station side relayed module I/O address |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Target station side module network number | Fixed to 0x00 | Fixed to 0x00 |
| ActPortNumber | 1 (PORT_1) | IBM-PC/AT compatible side COM port number | | | |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side module station number | Target station side module station number | Fixed to 0xFF |
| ActTimeOut | 10000 | Any value specified by user in ms units | | | |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Fixed to 0x00 | Target station side module station number |

＊1: As the I/O address, specify the value found by dividing the actual first I/O number by 16.
＊2: Note the following points when making access via the Ethernet module (QE71).
- For ActNetworkNumber and ActStationNumber, specify the value set in the parameter setting of the target station side QE71.
- Set the "MNET/10 routing information" in the parameter setting of the QE71. Also, when making setting, specify other than the automatic response system (any of the IP address calculation system, table conversion system and combined system) as the "MNET/10 routing system".

## 3.3.19 ActCCG4A control

The following table indicates the properties possessed by the ActCCG4A control and their default values.

### (1) Configuration




### (2) Property patterns

| Connected Station CPU | | | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|---|
| QCPU (A mode) | QnA CPU | ACPU ＊1 | | QCPU | | QnA CPU | ACPU ＊1 | FXCPU |
| | | | | Q mode | A mode | | | |
| ① | × | ① | MELSECNET/10H | × | × | × | × | × |
| | | | MELSECNET/10 | × | × | × | × | × |
| | | | MELSECNET(II) | × | × | × | × | × |
| | | | Ethernet | × | × | × | × | × |
| | | | Computer link | × | × | × | × | × |
| | | | CC-Link | × | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
＊1 : Including motion controller CPU

### (3) Property list

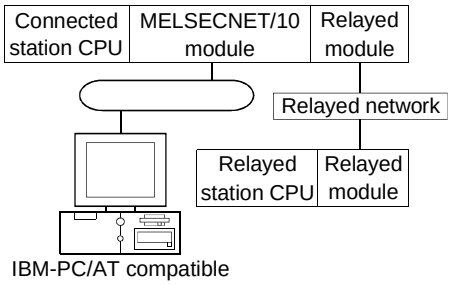
| Property | Default Value | Property Patterns |
|---|---|---|
| | | ① |
| ActControl | 8 (TCR_DTR_OR_RTS) | Depending on used cable. |
| ActCpuType | 262 (CPU_A1NCPU) | CPU type corresponding to target station |
| ActPortNumber | 1 (PORT_1) | IBM-PC/AT compatible side COM port number |
| ActStationNumber | 0 (0x00) | Target station side module station number |
| ActTimeOut | 10000 | Any value specified by user in ms units |

## 3.3.20 ActMnet10BD control

The following table indicates the properties possessed by the ActMnet10BD control and their default values.

### (1) When connected station CPU is QCPU (Q mode)

#### (a) Configuration




#### (b) Property patterns

| Own Board | Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|
| | QCPU (Q mode) | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | | | Q mode | A mode | | | |
| ① | ② | MELSECNET/10H | ②∗2 | × | × | × | × |
| | | MELSECNET/10 | ② | ② | ② | ② | × |
| | | MELSECNET(II) | × | × | × | × | × |
| | | Ethernet | ② | × | × | × | × |
| | | Computer link | ③ | × | × | × | × |
| | | CC-Link | ④ | × | × | × | × |

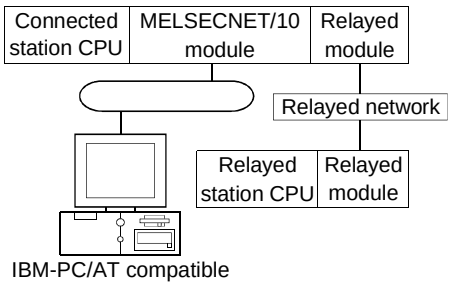
○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU
∗2 : Accessible with the performance of MELSECNET/10

#### (c) Property list

| Property | Default Value | Property Patterns | | | |
|---|---|---|---|---|---|
| | | ① | ② | ③ | ④ |
| ActCpuType | 1025 (CPU_BOARD) | CPU type corresponding to target station | | | |
| ActDestinationIONumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | 0x3FF | 0x3FF |
| ActDidPropertyBit | 0 (0x00) | 0x00 (valid) | 0x01 (invalid) | 0x00 (valid) | 0x00 (valid) |
| ActDsidPropertyBit | 0 (0x00) | 0x00 (valid) | 0x01 (invalid) | 0x00 (valid) | 0x00 (valid) |
| ActIONumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x3FF | Connected station side relayed module I/O address | Connected station side relayed module I/O address |
| ActMultiDropChannelNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Fixed to 0x02 | Fixed to 0x00 |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Target station side module network number | Connected station side module network number | Connected station side module network number |
| ActPortNumber | 1 (PORT_1) | Board No. of IBM-PC/AT compatible side MELSECNET/10 board, PORT 1 to PORT 4 (first to fourth boards) | | | |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Connected station side module station number | Connected station side module station number | Connected station side module station number |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Connected station side module station number | Connected station side module station number |

## (2) When connected station CPU is QCPU (A mode) or ACPU

### (a) Configuration




### (b) Property patterns

| Own Board | Connected Station CPU | | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|---|
| | QCPU (A mode) | ACPU＊1 | | QCPU | | QnA CPU | ACPU ＊1 | FXCPU |
| | | | | Q mode | A mode | | | |
| ① | ② | ② | MELSECNET/10H | × | × | × | × | × |
| | | | MELSECNET/10 | ② | ② | ② | ② | × |
| | | | MELSECNET(II) | × | × | × | × | × |
| | | | Ethernet | × | × | × | × | × |
| | | | Computer link | × | × | × | × | × |
| | | | CC-Link | × | × | × | × | × |

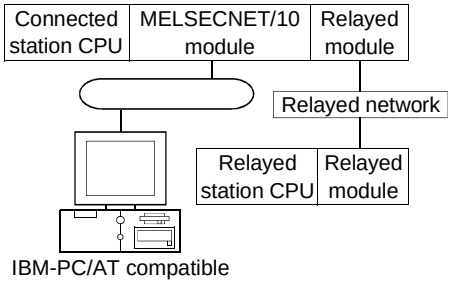
○ : Accessible (Property pattern within circle)
× : Inaccessible
＊1 : Including motion controller CPU

### (c) Property list

| Property | Default Value | Property Patterns | |
|---|---|---|---|
| | | ① | ② |
| ActCpuType | 1025 (CPU_BOARD) | CPU type corresponding to target station | |
| ActDestinationIONumber | 0 (0x00) | Fixed to 0x00 | |
| ActDidPropertyBit | 0 (0x00) | Fixed to 0x00 | |
| ActDsidPropertyBit | 0 (0x00) | Fixed to 0x00 | |
| ActIONumber | 0 (0x00) | Fixed to 0x00 | |
| ActMultiDropChannelNumber | 0 (0x00) | Fixed to 0x00 | |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Target station side module network number |
| ActPortNumber | 1 (PORT_1) | Board No. of IBM-PC/AT compatible side MELSECNET/10 board, PORT 1 to PORT 4 (first to fourth boards) | |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side module station number |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | |

### (3) When connected station CPU is QnACPU

#### (a) Configuration




#### (b) Property patterns

| Own Board | Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|
| | QnACPU | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | | | Q mode | A mode | | | |
| ① | ② | MELSECNET/10H | × | × | × | × | × |
| | | MELSECNET/10 | ② | ② | ② | ② | × |
| | | MELSECNET(II) | × | × | × | × | × |
| | | Ethernet | × | × | × | ② | × |
| | | Computer link | × | × | × | ③ | × |
| | | CC-Link | × | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU

#### (c) Property list

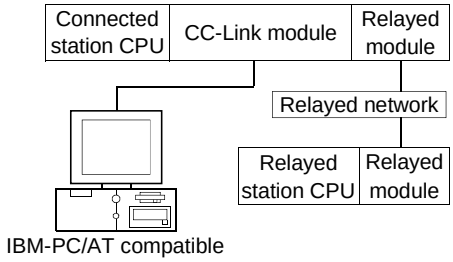
| Property | Default Value | Property Patterns | | |
|---|---|---|---|---|
| | | ① | ② | ③ |
| ActCpuType | 1025 (CPU_BOARD) | CPU type corresponding to target station | | |
| ActDestinationIONumber | 0 (0x00) | Fixed to 0x00 | | |
| ActDidPropertyBit | 0 (0x00) | Fixed to 0x00 | | |
| ActDsidPropertyBit | 0 (0x00) | Fixed to 0x00 | | |
| ActIONumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x3FF | Connected station side relayed module I/O address |
| ActMultiDropChannelNumber | 0 (0x00) | Fixed to 0x00 | | |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Target station side module network number | Connected station side module network number |
| ActPortNumber | 1 (PORT_1) | Board No. of IBM-PC/AT compatible side MELSECNET/10 board, PORT 1 to PORT 4 (first to fourth boards) | | |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side module station number | Connected station side module station number |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Target station side module station number |

## 3.3.21 ActCCBD control

The following table indicates the properties possessed by the ActCCBD control and their default values.

### (1) When connected station CPU is QCPU (Q mode)

#### (a) Configuration




#### (b) Property patterns

| Own Board | Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|
| | QCPU (Q mode) | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | | | Q mode | A mode | | | |
| ① | ② | MELSECNET/10H | ③ | × | × | × | × |
| | | MELSECNET/10 | ③ | × | × | × | × |
| | | MELSECNET(II) | × | × | × | × | × |
| | | Ethernet | ③ | × | × | × | × |
| | | Computer link | × | × | × | × | × |
| | | CC-Link | × | × | × | × | × |

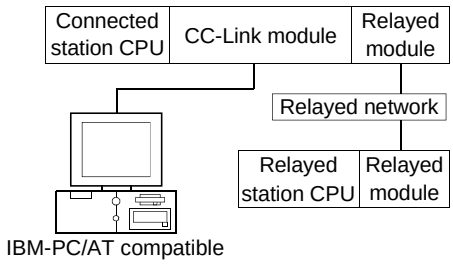
○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU

#### (c) Property list

| Property | Default Value | Property Patterns | | |
|---|---|---|---|---|
| | | ① | ② | ③ |
| ActCpuType | 1025 (CPU_BOARD) | CPU type corresponding to target station | | |
| ActDestinationIONumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x3FF | 0x3FF |
| ActIONumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x3FF | Fixed to 0x3FF |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Target station side module network number |
| ActPortNumber | 1 (PORT_1) | Board No. of IBM-PC/AT compatible side CC-Link board, PORT 1 to PORT 4 (first to fourth boards) | | |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side CC-Link module station number | Target station side module station number |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Connected station side module station number |

## (2) When connected station CPU is QCPU (A mode)

### (a) Configuration




### (b) Property patterns

| Own Board | Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|
| | QCPU (A mode) | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | | | Q mode | A mode | | | |
| ① | ② | MELSECNET/10H | × | × | × | × | × |
| | | MELSECNET/10 | × | × | × | × | × |
| | | MELSECNET(II) | × | × | × | × | × |
| | | Ethernet | × | × | × | × | × |
| | | Computer link | × | × | × | × | × |
| | | CC-Link | × | × | × | × | × |

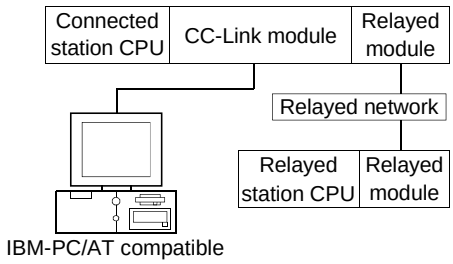
○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU

### (c) Property list

| Property | Default Value | Property Patterns | |
|---|---|---|---|
| | | ① | ② |
| ActCpuType | 1025 (CPU_BOARD) | CPU type corresponding to target station | |
| ActDestinationIONumber | 0 (0x00) | Fixed to 0x00 | |
| ActIONumber | 0 (0x00) | Fixed to 0x00 | |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | |
| ActPortNumber | 1 (PORT_1) | Board No. of IBM-PC/AT compatible side CC-Link board, PORT 1 to PORT 4 (first to fourth boards) | |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side CC-Link module station number |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | |

### (3) When connected station CPU is QnACPU

#### (a) Configuration




#### (b) Property patterns

| Own Board | Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|
| | | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | QnACPU | | Q mode | A mode | | | |
| ① | ② | MELSECNET/10H | × | × | × | × | × |
| | | MELSECNET/10 | × | × | ③ | × | × |
| | | MELSECNET(II) | × | × | × | × | × |
| | | Ethernet | × | × | ③ | × | × |
| | | Computer link | × | × | × | × | × |
| | | CC-Link | × | × | × | × | × |

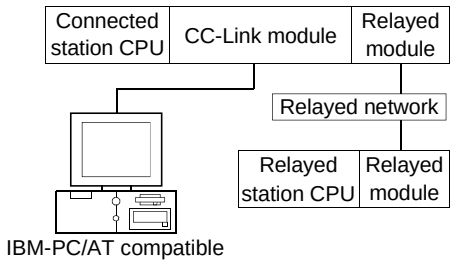
○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU

#### (c) Property list

| Property | Default Value | Property Patterns | | |
|---|---|---|---|---|
| | | ① | ② | ③ |
| ActCpuType | 1025 (CPU_BOARD) | CPU type corresponding to target station | | |
| ActDestinationIONumber | 0 (0x00) | Fixed to 0x00 | | |
| ActIONumber | 0 (0x00) | Fixed to 0x3FF | | |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Target station side module network number |
| ActPortNumber | 1 (PORT_1) | Board No. of IBM-PC/AT compatible side CC-Link board, PORT 1 to PORT 4 (first to fourth boards) | | |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side CC-Link module station number | Target station side module station number |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | Fixed to 0x00 | Target station side CC-Link module station number |

## (4) When connected station CPU is ACPU

### (a) Configuration




### (b) Property patterns

| Own Board | Connected Station CPU | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|---|
| | | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | ACPU ∗1 | | Q mode | A mode | | | |
| ① | ② | MELSECNET/10H | × | × | × | × | × |
| | | MELSECNET/10 | × | × | × | × | × |
| | | MELSECNET(II) | × | × | × | × | × |
| | | Ethernet | × | × | × | × | × |
| | | Computer link | × | × | × | × | × |
| | | CC-Link | × | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU

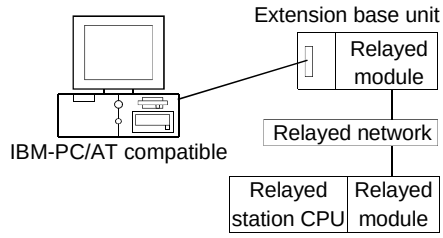
### (c) Property list

| Property | Default Value | Property Patterns | |
|---|---|---|---|
| | | ① | ② |
| ActCpuType | 1025 (CPU_BOARD) | CPU type corresponding to target station | |
| ActDestinationIONumber | 0 (0x00) | Fixed to 0x00 | |
| ActIONumber | 0 (0x00) | Fixed to 0x00 | |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | |
| ActPortNumber | 1 (PORT_1) | Board No. of IBM-PC/AT compatible side CC-Link board, PORT 1 to PORT 4 (first to fourth boards) | |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side module station number |
| ActUnitNumber | 0 (0x00) | Fixed to 0x00 | |

## 3.3.22 ActAnUBD control

The following table indicates the properties possessed by the ActAnUBD control and their default values.

### (1) Configuration




### (2) Property patterns

| Connected Station CPU / Own Board | Relayed Network | Relayed Station CPU | | | | |
|---|---|---|---|---|---|---|
| | | QCPU | | QnA CPU | ACPU ∗1 | FXCPU |
| | | Q mode | A mode | | | |
| ① | MELSECNET/10H | × | × | × | × | × |
| | MELSECNET/10 | × | ② | ②∗2 | ② | × |
| | MELSECNET(II) | × | ③ | ③∗2 | ③ | × |
| | Ethernet | × | × | × | × | × |
| | Computer link | × | × | × | × | × |
| | CC-Link | × | × | × | × | × |

○ : Accessible (Property pattern within circle)
× : Inaccessible
∗1 : Including motion controller CPU
∗2 : Operates as the one equivalent to AnACPU.

### (3) Property list

| Property | Default Value | Property Patterns | | |
|---|---|---|---|---|
| | | ① | ② | ③ |
| ActCpuType | 271 (CPU _A2USHS1CPU) | CPU type corresponding to target station | | |
| ActNetworkNumber | 0 (0x00) | Fixed to 0x00 | Target station side module network number | Fixed to 0x00 |
| ActStationNumber | 255 (0xFF) | Fixed to 0xFF | Target station side module station number | Target station side module station number |

### 3.3.23 ActLLT control

The following table indicates the properties possessed by the ActLLT control and their default values.

| Property | Default Value | Property Pattern |
|---|---|---|
| ActCpuType | 34 (CPU_Q02CPU) | CPU type corresponding to target station |
| ActTimeOut | 10000 | Any value specified by user in ms units |

# 4 FUNCTIONS

This chapter provides the programming instructions and function details (dispatch interface, custom interface).

| POINT |
|---|
| (1) For interface selection, we recommend you to choose the dispatch interface which is simpler.<br>(2) For programming, refer to "Section 4.1 Programming Instructions". |

## 4.1 Programming Instructions

This section gives the instructions for programming.



### (1) Instructions common to VB and VC++

(a) Multithread

When performing multithread programming, follow the rules of COM and ActiveX controls.

For details, refer to the rules and reference books of COM and ActiveX controls.

| POINT |
|---|
| (1) The ActiveX controls used on ACT are those of the STA model.<br>(2) When passed to another apartment, the interface pointer must be marshaled. Provide synchronization using the CoMarshallerThreadInterfaceInStream or CoGetInterfaceAndReleadseStream COM function. |

### (2) Instructions for use of VB

Only the dispatch interface is usable.

### (3) Instructions for use of VC++ (dispatch interface, custom interface)

(a) Both the dispatch interface and custom interface are usable.

(b) BSTR* type

In the functions which acquire the methods and properties using the BSTR pointer type, memory must be secured inside the ActiveX controls and released in user programs. (This is based on the rules of COM and ActiveX controls.)

(Example)

```
BSTR    szCpuName;
LONG    lCpuCode;
Obj.GetCpuType(&szCpuName, &lCpuCode );
MessgBox( “CpuName = %s, CpuCode = %d”, szCpuName, lCpuCode );
SysFreeString( szCpuName );
```

### (4) Instructions for use of VC++ (custom interface)

(a) HRESULT type

Use the SUCCEEDED or FAILED macro to check whether the HRESULT type, i.e. returned value of COM, resulted in normal or abnormal termination.

(Example)

```
HRESULT     hResult;
LONG        lRet;
hResult = Obj.Open( &lRet );
if( SUCCEEDE( hResult ) ) {
        if( lRet = SUCCESS ) {

        } else {
            MessgeBox( “Communication Error = %x”, lRet );
        }
} else {
        MessgeBox( “COM ERROR Occurd” );
}
```

## 4.2 Details of the Functions (Dispatch Interface)

This section explains the details of the functions.
The details of the functions in this section assume that the dispatch interface is used.
For the custom interface, refer to "Section 4.3 Details of the Functions (Custom Interface)".

### 4.2.1 Open (Communication line opening)

#### (1) Applicable ACT controls

This function is available for all ACT controls.

#### (2) Feature

Opens the communication line.

#### (3) Format

VB, VC++ : lRet = object.Close()

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |

#### (4) Explanation

The line is connected on the basis of the value set to the property for Open method.

#### (5) Returned value

Normal termination : 0 is returned.
Abnormal termination : A value other than 0 is returned.
(Refer to Chapter 6 ERROR CODES.)

**POINT**

(1) If the property for Open method is changed after completion of Open, the other end of communication is not changed.
To change the communication settings, close the communication line once, then set the other end of communication, and open the communication line again.

(2) Open may terminate normally if the CPU type entered into the ActCpuType property is different from the CPU used for communication.
In such a case, the connection range, usable methods and device range may be reduced, for example.
When performing Open, set the correct CPU type to the ActCpuType property.

## 4.2.2 Close (Communication line closing)

### (1) Applicable ACT controls

This function is available for all ACT controls.

### (2) Feature

Closes the communication line.

### (3) Format

VB, VC++ : lRet = object.Close()

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |

### (4) Explanation

The line connected using the Open function is closed.

### (5) Returned value

Normal termination : 0 is returned.
Abnormal termination : A value other than 0 is returned.
(Refer to Chapter 6 ERROR CODES.)

## 4.2.3 ReadDeviceBlock (Device batch-read)

### (1) Applicable ACT controls

This function is available for all ACT controls.

### (2) Feature

Batch-reads data from devices.

### (3) Format

VB : lRet = object.ReadDeviceBlock(szDevice, lSize, lData(0))

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| String | szDevice | Device name | Input |
| Long | lSize | Number of read points | Input |
| Long | lData(n) | Read device values | Output |

VC++ : lRet = object.ReadDeviceBlock(szDevice, lSize, ∗lplData)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| CString | szDevice | Device name | Input |
| Long | lSize | Number of read points | Input |
| Long | ∗lplData | Read device values | Output |

### (4) Explanation

- The device values for lSize are batch-read from the devices, beginning with the device specified in szDevice.
- The read device values are stored in lData or lplData.

### (5) Device specifying methods

Specify the devices in the following methods.

<When bit device is specified>
(Example) 3 points from M0

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ∗1 | M0 to M15∗2 |
| ∗1 | M16 to M31∗2 |
| ∗1 | M32 to M47∗2 |

<When word device is specified>
(Example) 3 points from D0

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ∗1 | D0 |
| ∗1 | D1 |
| ∗1 | D3 |

<When CN200 and later of FXCPU are specified>
(Example) 6 points from CN200 ∗3:

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ∗1 | L of CN200 |
| ∗1 | H of CN200 |
| ∗1 | L of CN201 |
| ∗1 | H of CN201 |
| ∗1 | L of CN202 |
| ∗1 | H of CN202 |

<When FD device is specified (4-word device)>
(Example) 6 points from FD0

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ∗1 | LL of FD0 |
| ∗1 | LH of FD0 |
| ∗1 | HL of FD0 |
| ∗1 | HH of FD0 |
| ∗1 | LL of FD1 |
| ∗1 | LH of FD1 |

∗1: Not used. (0 is stored.)
∗2: Lower bits are stored in device number order.
∗3: For CN200 or later of FXCPU, 2 words are read from 2 points. Read from 1 point will result in an error.

(6) Returned value

Normal termination : 0 is returned.
Abnormal termination : Any value other than 0 is returned.
(Refer to Chapter 6 ERROR CODES.)

POINT

(1) The maximum number of read points that may be specified in lSize should satisfy the following range.
Read starting device number + number of read points ≤ last device number
(2) When the bit device is specified, a multiple of 16 may be specified as the device number.
(3) For lData or lplData, prepare a memory area having the number of points specified in lSize.
If there is no memory area, a critical phenomenon such as an application error may occur.

## 4.2.4 WriteDeviceBlock (Device batch-write)

### (1) Applicable ACT controls

This function is available for all ACT controls.

### (2) Feature

Batch-writes data to devices.

### (3) Format

VB : lRet = object.WriteDeviceBlock(szDevice, lSize, lData(0))

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| String | szDevice | Device name | Input |
| Long | lSize | Number of read points | Input |
| Long | lData(n) | Written device values | Input |

VC++ : lRet = object.WriteDeviceBlock(szDevice, lSize, ∗lplData)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| CString | szDevice | Device name | Input |
| Long | lSize | Number of read points | Input |
| Long | ∗lplData | Written device values | Input |

### (4) Explanation

- The device values for lSize are batch-written to the devices, beginning with the device specified in szDevice.
- The written device values are stored in lData or lplData.

### (5) Device specifying methods

Specify the devices in the following methods.

<When bit device is specified>
(Example) 3 points from M0

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ∗1 | M0 to M15∗2 |
| ∗1 | M16 to M31∗2 |
| ∗1 | M32 to M47∗2 |

<When word device is specified>
(Example) 3 points from D0

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ∗1 | D0 |
| ∗1 | D1 |
| ∗1 | D2 |

<When CN200 and later of FXCPU are specified>
(Example) 6 points from CN200 ∗3:

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ∗1 | L of CN200 |
| ∗1 | H of CN200 |
| ∗1 | L of CN201 |
| ∗1 | H of CN201 |
| ∗1 | L of CN202 |
| ∗1 | H of CN202 |

<When FD device is specified (4-word device)>
(Example) 6 points from FD0

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ∗1 | LL of FD0 |
| ∗1 | LH of FD0 |
| ∗1 | HL of FD0 |
| ∗1 | HH of FD0 |
| ∗1 | LL of FD1 |
| ∗1 | LH of FD1 |

∗1: Not used. (0 is stored.)
∗2: Lower bits are stored in device number order.
∗3: For CN200 or later of FXCPU, 2 words are written from 2 points. Write from 1 point will result in an error.

(6) Returned value

Normal termination : 0 is returned.
Abnormal termination : Any value other than 0 is returned.
(Refer to Chapter 6 ERROR CODES.)

POINT

(1) The maximum number of write points that may be specified in lSize should satisfy the following range.
Write starting device number + number of write points ≤ last device number
(2) When the bit device is specified, a multiple of 16 may be specified as the device number.
(3) For lData or lplData, prepare a memory area having the number of points specified in lSize.
If there is no memory area, a critical phenomenon such as an application error may occur.

## 4.2.5 ReadDeviceRandom (Device random-read)

### (1) Applicable ACT controls

This function is available for all ACT controls.

### (2) Feature

Reads data randomly from devices.

### (3) Format

VB : lRet = object.ReadDeviceRandom(szDeviceList, lSize, lData(0))

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| String | szDeviceList | Device name | Input |
| Long | lSize | Number of read points | Input |
| Long | lData(n) | Read device values | Output |

VC++ : lRet = object.ReadDeviceRandom(szDeviceList, lSize, ∗lplData)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| CString | szDeviceList | Device name | Input |
| Long | lSize | Number of read points | Input |
| Long | ∗lplData | Read device values | Output |

### (4) Explanation

- The device values for lSize are read from the device group specified in szDeviceList.
- The read device values are stored in lData or lplData.
- Using the line feed symbol (\n), separate the devices in the character string specified in the device list.
  The last device need not be followed by the line feed symbol. (Example: D0\nM0\n....)

### (5) Device specifying methods

Specify the devices in the following methods.

(Example) When "M0\D0\K8M0" is specified (Number of points is 3)

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ∗1 | M0 |
| ∗1 | D0 |
| M16 to M31∗2 | M0 to M15∗2 |

(Example) When " D0\CN200\D1" including CN200 and later of FXCPU is specified (number of points is 3 in total) ∗3

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ∗1 | D0 |
| H of CN200 | L of CN200 |
| ∗1 | D1 |

(Example) When "D0\FD0\D1" including FD device is specified (Number of points is 3)

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ∗1 | D0 |
| ∗1 | LL of FD |
| ∗1 | D1 |

∗1: Not used. (0 is stored.)
∗2: Lower bits are stored in device number order.
∗3: For CN200 or later of FXCPU, 2 words are read from 1 point by random read.

(6) Returned value

Normal termination : 0 is returned.
Abnormal termination : Any value other than 0 is returned.
(Refer to Chapter 6 ERROR CODES.)

POINT

(1) The maximum number of read points that may be specified in lSize is up to 0x7FFFFFFF points.
(2) For lData or lplData, prepare a memory area having the number of points specified in lSize.
If there is no memory area, a critical phenomenon such as an application error may occur.

## 4.2.6 WriteDeviceRandom (Device random-write)

### (1) Applicable ACT controls

This function is available for all ACT controls.

### (2) Feature

Writes data randomly to devices.

### (3) Format

VB : lRet = object.WriteDeviceRandom(szDeviceList, lSize, lData(0))

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| String | szDeviceList | Device name | Input |
| Long | lSize | Number of read points | Input |
| Long | lData(n) | Written device values | Input |

VC++ : lRet = object.WriteDeviceRandom(szDeviceList, lSize, ∗lplData)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| CString | szDeviceList | Device name | Input |
| Long | lSize | Number of read points | Input |
| Long | ∗lplData | Written device values | Intput |

### (4) Explanation

- The device values for lSize are written to the devices specified in szDeviceList.
- The written device values are stored in lData or lplData.
- Using the line feed symbol (\n), separate the devices in the character string specified in the device list.
  The last device need not be followed by the line feed symbol. (Example: D0\nM0\n....)

### (5) Device specifying methods

Specify the devices in the following methods.

(Example) When "M0\D0\K8M0" is specified (Number of points is 3)

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ∗1 | M0 |
| ∗1 | D0 |
| M16 to M31∗2 | M0 to M15∗2 |

(Example) When " D0\CN200\D1" including CN200 and later of FXCPU is specified (number of points is 3 in total) ∗3

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ∗1 | D0 |
| H of CN200 | L of CN200 |
| ∗1 | D1 |

(Example) When "D0\FD0\D1" including FD device is specified (Number of points is 3)

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ∗1 | D0 |
| ∗1 | LL of FD |
| ∗1 | D1 |

∗1: Not used. (0 is stored.)
∗2: Lower bits are stored in device number order.
∗3: For CN200 or later of FXCPU, 2 words are written to 1 point by random write.

(6) Returned value

Normal termination : 0 is returned.

Abnormal termination : Any value other than 0 is returned.

(Refer to Chapter 6 ERROR CODES.)

POINT

(1) The maximum number of write points that may be specified in lSize is up to 0x7FFFFFFF points.

(2) For lData or lplData, prepare a memory area having the number of points specified in lSize.

If there is no memory area, a critical phenomenon such as an application error may occur.

## 4.2.7 SetDevice (Device data setting)

### (1) Applicable ACT controls

This function is available for all ACT controls.

### (2) Feature

Sets one point of device.

### (3) Format

VB : lRet = object.SetDevice(szDevice, lData)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| String | szDevice | Device name | Input |
| Long | lData | Set data | Input |

VC++ : lRet = object.SetDevice(szDevice, ＊lplData)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| CString | szDevice | Device name | Input |
| Long | ＊lplData | Set data | Intput |

### (4) Explanation

- The operation specified in lData or lplData is performed for one point of device specified in szDevice.
- When the bit device is specified, the least significant bit of the lData value or lplData value becomes valid.

### (5) Device specifying methods

Specify the devices in the following methods.

<When bit device is specified>

(Example) M0

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ＊1 | M0 |

<When word device is specified>

(Example) D0

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ＊1 | D0 |

<When double-word device is specified>

(Example) K8M0

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| M16 to M31＊2 | M0 to M15＊2 |

<When CN200 or later of FXCPU is specified>

(Example) CN200

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| H of CN200 | L of CN200 |

＊1: Not used. (0 is stored.)

＊2: Lower bits are stored in device number order.

### (6) Returned value

Normal termination : 0 is returned.

Abnormal termination : Any value other than 0 is returned.

(Refer to Chapter 6 ERROR CODES.)

## 4.2.8 GetDevice (Device data acquisition)

### (1) Applicable ACT controls

This function is available for all ACT controls.

### (2) Feature

Acquires data from one point of device.

### (3) Format

VB : lRet = object.GetDevice(szDevice, lData)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| String | szDevice | Device name | Input |
| Long | lData | Set data | Output |

VC++ : lRet = object.GetDevice(szDevice, ＊lplData)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| CString | szDevice | Device name | Input |
| Long | ＊lplData | Set data | Output |

### (4) Explanation

The data of one point of device specified in szDevice is stored into lData or lplData.

### (5) Device specifying methods

Specify the devices in the following methods.

<When bit device is specified>
(Example) M0

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ＊1 | M0 |

<When word device is specified>
(Example) D0

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| ＊1 | D0 |

<When double-word device is specified>
(Example) K8M0

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| M16 to M31＊2 | M0 to M15＊2 |

<When CN200 or later of FXCPU is specified>
(Example) CN200

| 2 Upper Bytes | 2 Lower Bytes |
|---|---|
| H of CN200 | L of CN200 |

＊1: Not used. (0 is stored.)

＊2: Lower bits are stored in device number order.

### (6) Returned value

Normal termination : 0 is returned.

Abnormal termination : Any value other than 0 is returned.
(Refer to Chapter 6 ERROR CODES.)

## 4.2.9 ReadBuffer (Buffer memory read)

### (1) Applicable ACT controls

The applicable ACT controls are indicated below.

| Control Name | Usability |
|---|---|
| ActEasyIF | ○ |
| ActQCPUQ | ○ |
| ActQCPUA | ○ |
| ActQnACPU | ○ |
| ActACPU | ○ |
| ActFXCPU | ○*4 |
| ActQJ71C24 | ○ |
| ActAJ71QC24 | ○ |
| ActAJ71UC24 | × |
| ActAJ71C24 | × |
| ActQJ71E71TCP | ○ |
| ActQJ71E71UDP | ○ |
| ActAJ71QE71TCP | × |

| Control Name | Usability |
|---|---|
| ActAJ71QE71UDP | ○*1, *2 |
| ActAJ71E71TCP | × |
| ActAJ71E71UDP | ○ |
| ActQCPUQUSB | ○ |
| ActCCG4QnA | ○ |
| ActCCG4A | ○ |
| ActMnet10BD | ○*3 |
| ActCCBD | ○*3 |
| ActAnUBD | ○*6 |
| ActLLT | ○*5 |

○: Usable ×: Unusable

*1: An error is returned if access to the AnUCPU, QCPU (A mode), A173UHCPU(-S1) or A273UH-S3) is made.
*2: An error is returned if access to the QnACPU is made.
*3: An error is returned if own board access is made.
*4: An error is returned if the CPU is other than FX2N and FX2NC.
*5: An error is returned if the CPU is other than FX0N, FX2, FX2C, FX2N and FX2NC.
*6: An error is returned if access to the QnACPU is made via the MELSECNET/10 or MELSECNET(II).

### (2) Feature

Reads the buffer memory values of the special function module.

### (3) Format

VB :lRet = object.ReadBuffer(lStartIO, lAddress, lReadSize, iData(0))

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| Long | lStartIO | First I/O number of module from where values will be read | Input |
| Long | lAddress | Buffer memory address | Input |
| Long | lReadSize | Read size | Input |
| Integer | iData(n) | Values read from buffer memory | Output |

VC++ :lRet = object.ReadBuffer(lStartIO, lAddress, lReadSize *lpsData)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| Long | lStartIO | First I/O number of module from where values will be read | Input |
| Long | lAddress | Buffer memory address | Input |
| Long | lReadSize | Read size | Input |
| Short | *lpsData | Values read from buffer memory | Output |

(4) Explanation

- As the module I/O number specified in lStartIO, specify a value found by dividing the actual I/O number by 16.
- The buffer values for lReadSize at the buffer memory address specified in lAddress in the special function module located at the first I/O number specified in lStartIO are read.
- When using the ActFXCPU control or ActLLT control, specify the block number (0 to 7) of the special expansion equipment as the module's first I/O number and any of 0 to 32767 as the buffer memory address.

(5) Returned value

Normal termination : 0 is returned.
Abnormal termination : Any value other than 0 is returned.
(Refer to Chapter 6 ERROR CODES.)

POINT

(1) An error is returned if access to the motion controller CPU is made.
(2) For iData or lpsData, prepare a memory area having the number of points specified in lReadSize.
If there is no memory area, a critical phenomenon such as an application error may occur.
(3) When buffer memory read (ReadBuffer) is performed for the QCPU (Q mode), read operation may be performed for only the Q series-dedicated module.

## 4.2.10 WriteBuffer (Buffer memory write)

### (1) Applicable ACT controls

The applicable ACT controls are indicated below.

| Control Name | Usability |
|---|---|
| ActEasyIF | ○ |
| ActQCPUQ | ○ |
| ActQCPUA | ○ |
| ActQnACPU | ○ |
| ActACPU | ○ |
| ActFXCPU | ○*4 |
| ActQJ71C24 | ○ |
| ActAJ71QC24 | ○ |
| ActAJ71UC24 | × |
| ActAJ71C24 | × |
| ActQJ71E71TCP | ○ |
| ActQJ71E71UDP | ○ |
| ActAJ71QE71TCP | × |

| Control Name | Usability |
|---|---|
| ActAJ71QE71UDP | ○*1, *2 |
| ActAJ71E71TCP | × |
| ActAJ71E71UDP | ○ |
| ActQCPUQUSB | ○ |
| ActCCG4QnA | ○ |
| ActCCG4A | ○ |
| ActMnet10BD | ○*3 |
| ActCCBD | ○*3 |
| ActAnUBD | ○*6 |
| ActLLT | ○*5 |

○: Usable ×: Unusable

*1: An error is returned if access to the AnUCPU, QCPU (A mode), A173UHCPU(-S1) or A273UH(-S3) is made.
*2: An error is returned if access to the QnACPU is made.
*3: An error is returned if own board access is made.
*4: An error is returned if the CPU is other than FX2N and FX2NC.
*5: An error is returned if the CPU is other than FX0N, FX2, FX2C, FX2N and FX2NC.
*6: An error is returned if access to the QnACPU is made via the MELSECNET/10 or MELSECNET(II).

### (2) Feature

Writes values to the buffer memory of the special function module.

### (3) Format

VB : lRet = object.WriteBuffer(lStartIO, lAddress, lWriteSize, iData(0))

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| Long | lStartIO | First I/O number of module to where values will be written | Input |
| Long | lAddress | Buffer memory address | Input |
| Long | lWriteSize | Write size | Input |
| Integer | iData(n) | Values written to buffer memory | Input |

VC++ : lRet = object.ReadBuffer(lStartIO, lAddress, lWriteSize *lpsData)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| Long | lStartIO | First I/O number of module to where values will be written | Input |
| Long | lAddress | Buffer memory address | Input |
| Long | lWriteSize | Write size | Input |
| Short | *lpsData | Values written to buffer memory | Input |

(4) Explanation

- As the module I/O number specified in lStartIO, specify a value found by dividing the actual I/O number by 16.
- The buffer values for lWriteSize at the buffer memory address specified in lAddress in the special function module located at the first I/O number specified in lStartIO are written.
- When using the ActFXCPU control or ActLLT control, specify the block number (0 to 7) of the special expansion equipment as the module's first I/O number and any of 0 to 32767 as the buffer memory address.

(5) Returned value

Normal termination : 0 is returned.
Abnormal termination : Any value other than 0 is returned.
(Refer to Chapter 6 ERROR CODES.)

POINT

(1) An error is returned if access to the motion controller CPU is made.
(2) For iData or lpsData, prepare a memory area having the number of points specified in lWriteSize.
If there is no memory area, a critical phenomenon such as an application error may occur.
(3) When buffer memory write (WriteBuffer) is performed for the QCPU (Q mode), write operation may be performed for only the Q series-dedicated module.

## 4.2.11 GetClockData (Clock data read)

### (1) Applicable ACT controls

The applicable ACT controls are indicated below.

| Control Name | Usability |
|---|---|
| ActEasyIF | ○ |
| ActQCPUQ | ○ |
| ActQCPUA | ○ |
| ActQnACPU | ○ |
| ActACPU | ○ |
| ActFXCPU | ○ |
| ActQJ71C24 | ○ |
| ActAJ71QC24 | ○∗2 |
| ActAJ71UC24 | ○∗2 |
| ActAJ71C24 | ○ |
| ActQJ71E71TCP | ○ |
| ActQJ71E71UDP | ○ |
| ActAJ71QE71TCP | × |

| Control Name | Usability |
|---|---|
| ActAJ71QE71UDP | ○ |
| ActAJ71E71TCP | × |
| ActAJ71E71UDP | ○ |
| ActQCPUQUSB | ○ |
| ActCCG4QnA | ○ |
| ActCCG4A | ○ |
| ActMnet10BD | ○∗1 |
| ActCCBD | ○∗1 |
| ActAnUBD | ○∗3 |
| ActLLT | × |

○: Usable ×: Unusable

∗1: An error is returned if own board access is made.
∗2: An error is returned if access to the QnACPU is made.
∗3: An error is returned if access to the QnACPU is made via the MELSECNET/10 or MELSECNET(II).

### (2) Feature

Reads time from the clock data of the PLC CPU.

### (3) Format

VB : lRet = object.GetClockData(iYear, iMonth, iDay, iDayOfWeek, iHour, iMinute, iSecond)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| Integer | iYear | Read year value | Output |
| Integer | iManth | Read month value | Output |
| Integer | iDay | Read day value | Output |
| Integer | iDayOfWeek | Read day-of-week value | Output |
| Integer | iHour | Read hour value | Output |
| Integer | iMinute | Read minute value | Output |
| Integer | iSecond | Read second value | Output |

VC++ : lRet = object.ReadBuffer(∗lpsYear, ∗lpsMonth, ∗lpsDay, ∗lpsDayOfWeek, ∗lpsHour, ∗lpsMinute, ∗lpsSecond)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| Short | ∗lpsYear | Read year value | Output |
| Short | ∗lpsMonth | Read month value | Output |
| Short | ∗lpsDay | Read day value | Output |
| Short | ∗lpsDaYOfWeek | Read day-of-week value | Output |
| Short | ∗lpsHour | Read hour value | Output |
| Short | ∗lpsMinute | Read minute value | Output |
| Short | ∗lpsSecond | Read second value | Output |

(4) Explanation

• An error is returned if correct clock data is not set to the PLC CPU.
• As the value stored into iYear or lpsYear, a four-digit year is returned for the QCPU (Q mode) or a two-digit year for any other CPU.
Note that the year for the QCPU (Q mode) is between 1980 and 2079.
• The value stored into iDayOfWeek or lpsDayOfWeek is as follows.

| Value | Day of Week |
|---|---|
| 0 | Sunday |
| 1 | Monday |
| 2 | Tuesday |
| 3 | Wednesday |
| 4 | Thursday |
| 5 | Friday |
| 6 | Saturday |

(5) Returned value

Normal termination : 0 is returned.
Abnormal termination : Any value other than 0 is returned.
(Refer to Chapter 6 ERROR CODES.)

POINT

(1) Clock data cannot be read from the A0J2HCPU, A2CCPU and A2CJCPU as they do not have clock data.
(2) For the QCPU (A mode) and ACPU, clock data can be set only when the target station is in the STOP status.
(3) For the FXCPU, clock data can be read from the FX1N, FX1S, FX2N or FX2NC when it has a built-in clock, or from the FX2 or FX2C when it is fitted with the RTC cassette.
An error is returned if the FXCPU is other than the FX1N, FX1S, FX2, FX2C, FX2N and FX2NC.
(4) Note that an error of transfer time is produced in clock setting.

## 4.2.12 SetClockData (Clock data write)

### (1) Applicable ACT controls

The applicable ACT controls are indicated below.

| Control Name | Usability |
|---|---|
| ActEasyIF | ○ |
| ActQCPUQ | ○ |
| ActQCPUA | ○ |
| ActQnACPU | ○ |
| ActACPU | ○ |
| ActFXCPU | ○ |
| ActQJ71C24 | ○ |
| ActAJ71QC24 | ○*2 |
| ActAJ71UC24 | ○*2 |
| ActAJ71C24 | ○ |
| ActQJ71E71TCP | ○ |
| ActQJ71E71UDP | ○ |
| ActAJ71QE71TCP | × |

| Control Name | Usability |
|---|---|
| ActAJ71QE71UDP | ○ |
| ActAJ71E71TCP | × |
| ActAJ71E71UDP | ○ |
| ActQCPUQUSB | ○ |
| ActCCG4QnA | ○ |
| ActCCG4A | ○ |
| ActMnet10BD | ○*1 |
| ActCCBD | ○*1 |
| ActAnUBD | ○*3 |
| ActLLT | × |

○: Usable ×: Unusable

*1: An error is returned if own board access is made.
*2: An error is returned if access to the QnACPU is made.
*3: An error is returned if access to the QnACPU is made via the MELSECNET/10 or MELSECNET(II).

### (2) Feature

Writes time to the clock data of the PLC CPU.

### (3) Format

VB : lRet = object.SetClockData(iYear, iMonth, iDay, iDayOfWeek, iHour, iMinute, iSecond)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| Integer | iYear | Year value to be written | Input |
| Integer | iManth | Month value to be written | Input |
| Integer | iDay | Day value to be written | Input |
| Integer | iDayOfWeek | Day-of-week value to be written | Input |
| Integer | iHour | Hour value to be written | Input |
| Integer | iMinute | Minute value to be written | Input |
| Integer | iSecond | Second value to be written | Input |

VC++ : lRet = object.ReadBuffer(sYear, sMonth, sDay, sDayOfWeek, sHour, sMinute, sSecond)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| Short | sYear | Year value to be written | Input |
| Short | sMonth | Month value to be written | Input |
| Short | sDay | Day value to be written | Input |
| Short | sDaYOfWeek | Day-of-week value to be written | Input |
| Short | sHour | Hour value to be written | Input |
| Short | sMinute | Minute value to be written | Input |
| Short | sSecond | Second value to be written | Input |

(4) Explanation

- An error is returned if the clock data to be set are not correct values.
- As to the value specified in iYear or sYear, a four-digit year is valid for the QCPU (Q mode) or a two-digit year for any other CPU.
  Note that the year valid for the QCPU (Q mode) is between 1980 and 2079.
  An error will occur if a four-digit year is set to any CPU other than the QCPU (Q mode).
- The value to be specified in iDayOfWeek or sDayOfWeek is as follows.

| Value | Day of Week |
|---|---|
| 0 | Sunday |
| 1 | Monday |
| 2 | Tuesday |
| 3 | Wednesday |
| 4 | Thursday |
| 5 | Friday |
| 6 | Saturday |

(5) Returned value

Normal termination : 0 is returned.
Abnormal termination : Any value other than 0 is returned.
(Refer to Chapter 6 ERROR CODES.)

POINT

(1) Clock data cannot be read from the A0J2HCPU, A2CCPU and A2CJCPU as they do not have clock data.
(2) For the QCPU (A mode) and ACPU, clock data can be set only when the target station is in the STOP status.
(3) For the QCPU (A mode) and ACPU, the clock setting special relay "M9028" changes to OFF after clock data setting.
(4) For the FXCPU, clock setting can be made to the FX1N, FX1S, FX2N or FX2NC when it has a built-in clock, or to the FX2 or FX2C when it is fitted with the RTC cassette.
An error is returned if the FXCPU is other than the FX1N, FX1S, FX2, FX2C, FX2N and FX2NC.
(5) Note that an error of transfer time is produced in clock setting.

## 4.2.13 GetCpuType (PLC CPU type read)

### (1) Applicable ACT controls

This function is available for all ACT controls∗1.

∗1: MELSECNET/10 board will result in an error if own board access is made.

### (2) Feature

Reds the type character string and type code of the PLC CPU.

### (3) Format

VB : lRet = object.GetCpuType(szCpuName, lCpuType)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| String | szCpuName | PLC CPU type character string | Output |
| Long | lCpuType | PLC CPU type code | Output |

VC++ : lRet = object.GetCpuType(∗szCpuType, ∗lplCpuType)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| BSTR | ∗szCpuName | PLC CPU type character string | Output |
| Long | ∗lplCpuType | PLC CPU type code | Output |

### (4) Explanation

- The type of the PLC which is making communication is stored into szCpuName and its type code into lCpuType or lplCpuType.
- The PLC CPU type character string is returned in UNICODE.

## (5) CPU type character string and type code

The following table lists the CPU type character strings and type codes read using GetCpuType.

### (a) Type character string list

| CPU/Network Board Type | Type Character String | | CPU/Network Board Type | Type Character String | |
|---|---|---|---|---|---|
| | CPU/Network Board Type | When LLT is connected | | CPU/Network Board Type | When LLT is connected |
| Q02CPU | Q02CPU | Q02CPU | A2ACPUP21/R21-S1 | A2AS1 | A2AS1 |
| Q02HCPU | Q02HCPU | Q02CPU | A2UCPU | A2U | A2U |
| Q06HCPU | Q06HCPU | Q06HCPU | A2UCPU-S1 | A2US1 | A2U |
| Q12HCPU | Q12HCPU | Q12HCPU | A2USCPU | A2U | A2U |
| Q25HCPU | Q25HCPU | Q25HCPU | A2USCPU-S1 | A2US1 | A2U |
| Q02CPU-A | Q02CPU | Q02CPU-A | A2ASCPU | A2U | A2U |
| Q02HCPU-A | Q02HCPU | Q02CPU-A | A2ASCPU-S1 | A2US1 | A2U |
| Q06HCPU-A | Q06HCPU | Q06HCPU-A | A2ASCPU-S30 | A3U | A3U |
| Q2ACPU | Q2ACPU | Q2ACPU | A2USHCPU-S1 | A2USH | A2USH |
| Q2ACPU-S1 | Q2ACPU-S1 | Q2ACPU-S1 | A3NCPU | A3N | A3N |
| Q2ASCPU | Q2ACPU | Q2ACPU | A3ACPU | A3A | A3A |
| Q2ASCPU-S1 | Q2ACPU-S1 | Q2ACPU-S1 | A3ACPUP21/R21 | A3A | A3A |
| Q2ASHCPU | Q2ACPU | Q2ACPU | A3UCPU | A3U | A3U |
| Q2ASHCPU-S1 | Q2ACPU-S1 | Q2ACPU-S1 | A4UCPU | A4U | A4U |
| Q3ACPU | Q3ACPU | Q3ACPU | A1FXCPU | A1FX | A1FX |
| Q4ACPU | Q4ACPU | Q4ACPU | $FX_0$ | $FX_0$/$FX_{0S}$ | $FX_0$/$FX_{0S}$ |
| Q4ARCPU | Q4ACPU | Q4ACPU | $FX_{0S}$ | $FX_0$/$FX_{0S}$ | $FX_0$/$FX_{0S}$ |
| A0J2HCPU | A0J2H | A0J2H | $FX_{0N}$ | $FX_{0N}$ | $FX_{0N}$ |
| A1SCPU | A1S | A1S | $FX_1$ | $FX_1$ | $FX_1$ |
| A1SCPU-S1 | A1S | A1S | $FX_{1S}$ | $FX_{1S}$ | $FX_{1S}$ |
| A1SCPUC24-R2 | A1S | A1S | $FX_{1N}$ | $FX_{1N}$ | $FX_{1N}$ |
| A1SHCPU | A1SH | A1SH | $FX_2$ | $FX_2$/$FX_{2C}$ | $FX_2$/$FX_{2C}$ |
| A1SJCPU | A1S | A1S | $FX_{2C}$ | $FX_2$/$FX_{2C}$ | $FX_2$/$FX_{2C}$ |
| A1SJHCPU | A1SH | A1SH | $FX_{2N}$ | $FX_{2N}$/$FX_{2NC}$ | $FX_{2N}$/$FX_{2NC}$ |
| A1NCPU | A1N | A1N | $FX_{2NC}$ | $FX_{2N}$/$FX_{2NC}$ | $FX_{2N}$/$FX_{2NC}$ |
| A2CCPU | A2C | A2C | A171SHCPU | A171SH | A171SH |
| A2CCPUC24 | A2C | A2C | A172SHCPU | A172SH | A172SH |
| A2CCPUC24-PRF | A2C | A2C | A173UHCPU | A173UHCPU | A173UH |
| A2CJCPU | A2C | A2C | A173UHCPU-S1 | A173UHCPU-S1 | A173UH |
| A2NCPU | A2N | A2N | A273UHCPU | A273UH | A273UH |
| A2NCPU-S1 | A2N | A2N | A273UHCPU-S3 | A273UH | A273UH |
| A2SCPU | A2S | A2N | A70BDE-J71QLP23(GE) | A70BDE-J71QLP23 | — |
| A2SCPU-S1 | A2S | A2N | A70BDE-J71QBR13 | A70BDE-J71QBR13 | — |
| A2SHCPU | A2SH | A2SH | A70BDE-J71QLR23 | A70BDE-J71QLR23 | — |
| A2SHCPU-S1 | A2SH | A2SH | A80BDE-J61BT11 | A80BDE-J61BT11 | — |
| A2ACPU | A2A | A2AS1 | A80BDE-J61BT13 | A80BDE-J61BT13 | — |
| A2ACPU-S1 | A2AS1 | A2AS1 | A80BDE-A2USH-S1 | A2USH-S1 | — |
| A2ACPUP21/R21 | A2AS1 | A2AS1 | | | |

(b) Type code list

| CPU/Network Board Type | Type Code | | CPU/Network Board Type | Type Code | |
|---|---|---|---|---|---|
| | When CPU/own board is connected | When LLT is connected | | When CPU/own board is connected | When LLT is connected |
| Q02CPU | 41H | 41H | A2ACPUP21/R21-S1 | 93H | 93H |
| Q02HCPU | 41H | 41H | A2UCPU | 82H | 82H |
| Q06HCPU | 42H | 42H | A2UCPU-S1 | 83H | 83H |
| Q12HCPU | 43H | 43H | A2USCPU | 82H | 82H |
| Q25HCPU | 44H | 44H | A2USCPU-S1 | 83H | 82H |
| Q02CPU-A | 141H | 141H | A2ASCPU | 82H | 82H |
| Q02HCPU-A | 141H | 141H | A2ASCPU-S1 | 82H | 82H |
| Q06HCPU-A | 142H | 142H | A2ASCPU-S30 | 94H | 84H |
| Q2ACPU | 21H | 21H | A2USHCPU-S1 | 84H | 84H |
| Q2ACPU-S1 | 22H | 22H | A3NCPU | A3H | A3H |
| Q2ASCPU | 21H | 21H | A3ACPU | 94H | 94H |
| Q2ASCPU-S1 | 22H | 22H | A3ACPUP21/R21 | 94H | 94H |
| Q2ASHCPU | 21H | 21H | A3UCPU | 84H | 84H |
| Q2ASHCPU-S1 | 22H | 22H | A4UCPU | 85H | 85H |
| Q3ACPU | 23H | 23H | A1FXCPU | A2H | A2H |
| Q4ACPU | 24H | 24H | $FX_0$ | F0H | F0H |
| Q4ARCPU | 24H | 24H | $FX_{0S}$ | F0H | F0H |
| A0J2HCPU | 98H | 98H | $FX_{0N}$ | 8EH | 8EH |
| A1SCPU | 98H | 98H | $FX_1$ | F1H | F1H |
| A1SCPU-S1 | 98H | 98H | $FX_{1S}$ | F2H | F2H |
| A1SCPUC24-R2 | 98H | 98H | $FX_{1N}$ | 9EH | 9EH |
| A1SHCPU | A3H | A3H | $FX_2$ | 8DH | 8DH |
| A1SJCPU | 98H | 98H | $FX_{2C}$ | 8DH | 8DH |
| A1SJHCPU | A3H | A3H | $FX_{2N}$ | 9DH | 9DH |
| A1NCPU | A1H | A1H | $FX_{2NC}$ | 9DH | 9DH |
| A2CCPU | 9AH | 9AH | A171SHCPU | A3H | A3H |
| A2CCPUC24 | 9AH | 9AH | A172SHCPU | A3H | A3H |
| A2CCPUC24-PRF | 9AH | 9AH | A173UHCPU | 84H | 84H |
| A2CJCPU | 9AH | 9AH | A173UHCPU-S1 | 84H | 84H |
| A2NCPU | A2H | A2H | A273UHCPU | 84H | 84H |
| A2NCPU-S1 | A2H | A2H | A273UHCPU-S3 | 84H | 84H |
| A2SCPU | A2H | A2H | A70BDE-J71QLP23(GE) | 90H | — |
| A2SCPU-S1 | A2H | A2H | A70BDE-J71QBR13 | 90H | — |
| A2SHCPU | A3H | A3H | A70BDE-J71QLR23 | 90H | — |
| A2SHCPU-S1 | A3H | A3H | A80BDE-J61BT11 | 90H | — |
| A2ACPU | 92H | 92H | A80BDE-J61BT13 | 90H | — |
| A2ACPU-S1 | 93H | 93H | A80BDE-A2USH-S1 | 84H | — |
| A2ACPUP21/R21 | 92H | 93H | | | |

1) When using the TCP/IP of the E71 or QE71, refer to the manual of the corresponding module.
2) When access to the AnUCPU, QnACPU, QCPU (A mode) or A273UHCPU(-S3) is made from the C24 or E71, the type code equivalent to that of the AnACPU is returned. (92H, 93H, 94H)
3) When access to the AnUCPU, QnACPU, QCPU (A mode) or A273UHCPU(-S3) is made from the C24, E71 or UC24 via the network, the type code equivalent to that of the AnACPU is returned. (92H, 93H, 94H)

4) When access to the AnUCPU, QCPU (A mode) or A273UHCPU(-S3) is made from the AnNCPU or AnACPU via the network by CPU COM communication, the type code equivalent to that of the AnACPU is returned. (92H, 93H, 94H)

5) When access to the QnACPU or QCPU (A mode) is made from the CPU board, the type code equivalent to that of the AnACPU (92H, 93H, 94H) is returned for the QnACPU or the type code equivalent to that of the A4UCPU (85H) is returned for the QCPU (A mode).

6) When access to the QCPU (A mode) is made from the UC24, the type code equivalent to that of the A4UCPU (85H) is returned.

7) When access to the QCPU (A mode) is made from the CC-Link G4 module, the type code equivalent to that of the A4UCPU (85H) is returned.

### (5) Returned value

Normal termination : 0 is returned.
Normal termination : Abnormal termination: A value other than 0 is returned. (Refer to Chapter 6 ERROR CODES.)

## 4.2.14 SetCpuStatus (Remote control)

### (1) Applicable ACT controls

The applicable ACT controls are indicated below.

| Control Name | Usability |
|---|---|
| ActEasyIF | ○ |
| ActQCPUQ | ○ |
| ActQCPUA | ○ |
| ActQnACPU | ○ |
| ActACPU | ○ |
| ActFXCPU | ○ |
| ActQJ71C24 | ○ |
| ActAJ71QC24 | ○ |
| ActAJ71UC24 | ○*4 |
| ActAJ71C24 | ○*4 |
| ActQJ71E71TCP | ○ |
| ActQJ71E71UDP | ○ |
| ActAJ71QE71TCP | ○*1 |

| Control Name | Usability |
|---|---|
| ActAJ71QE71UDP | ○ |
| ActAJ71E71TCP | ○*1 |
| ActAJ71E71UDP | ○ |
| ActQCPUQUSB | ○ |
| ActCCG4QnA | ○ |
| ActCCG4A | ○ |
| ActMnet10BD | ○*2 |
| ActCCBD | ○*2 |
| ActAnUBD | ○*3 |
| ActLLT | ○ |

○: Usable ×: Unusable

*1: An error is returned when remote operation is performed for the own station.
*2: An error is returned when own board access is made.
*3: When access to the QnACPU is made via the MELSECNET/10 or MELSECNET(II), making PAUSE specification for the QnACPU results in an error.
*4: An error is returned if PAUSE specification is made.

### (2) Feature

Performs remote operation of the PLC CPU.

### (3) Format

VB : lRet = object.SetCpuStatus(lOperation)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| Long | lOperation | Remote run/stop/pause | Input |

VC++ : lRet = object.SetCpuStatus(lOperation)

| | | | |
|---|---|---|---|
| Long | lRet | Returned value | Output |
| Long | lOperation | Remote run/stop/pause | Input |

### (4) Explanation

• The operation specified in lOperation is performed.
Specifying any value other than the following will result in an error.

| Value | Operation |
|---|---|
| 0 | Remote run |
| 1 | Remote stop |
| 2 | Remote pause |

(5) Returned value

Normal termination : 0 is returned.
Abnormal termination : A value other than 0 is returned.
(Refer to Chapter 6 ERROR CODES.)

| POINT |
| --- |
| Since the FXCPU does not have the PAUSE switch as the PLC CPU, an error is returned if remote pause is specified in SetCpuStatus. |

## 4.3 Details of the Functions (Custom Interface)

This section explains the details of the functions.
The details of the functions in this section assume that the custom interface is used.
The custom interface may be used on only VC++.
For the dispatch interface, refer to "Section 4.2 Details of the Functions (Dispatch Interface)".
This section describes only the formats of the functions.
For details of other than the formats, refer to "Section 4.2 Details of the Functions (Dispatch Interface)".

### 4.3.1 Open (Communication line opening)

hResult = object.Open(∗lplRetCode )

| | | | |
|---|---|---|---|
| HRESULT | hResult | Returned value of COM | Output |
| LONG | ∗lplRetCode | Returned value of communication function | Output |

### 4.3.2 Close (Communication line closing)

hResult = object.Close(∗lplRetCode )

| | | | |
|---|---|---|---|
| HRESULT | hResult | Returned value of COM | Output |
| LONG | ∗lplRetCode | Returned value of communication function | Output |

### 4.3.3 ReadDeviceBlock (Device batch-read)

hResult = object.ReadDeviceBlock( szDevice, lSize, ∗lplData, ∗lplRetCode )

| | | | |
|---|---|---|---|
| HRESULT | hResult | Returned value of COM | Output |
| BSTR | szDevice | Device name | Input |
| LONG | lSize | Number of read points | Input |
| LONG | ∗lplData | Read device values | Output |
| LONG | ∗lplRetCode | Returned value of communication function | Output |

### 4.3.4 WriteDeviceBlock (Device batch-write)

hResult = object.WriteDeviceBlock( szDevice, lSize, ∗lplData, ∗lplRetCode )

| | | | |
|---|---|---|---|
| HRESULT | hResult | Returned value of COM | Output |
| BSTR | szDevice | Device name | Input |
| LONG | lSize | Number of write points | Input |
| LONG | ∗lplData | Written device values | Input |
| LONG | ∗lplRetCode | Returned value of communication function | Output |

### 4.3.5 ReadDeviceRandom (Device random-read)

hResult = object.ReadDeviceBlock( szDevice, lSize, ∗lplData, ∗lplRetCode )

| | | | |
|---|---|---|---|
| HRESULT | hResult | Returned value of COM | Output |
| BSTR | szDevice | Device name | Input |
| LONG | lSize | Number of read points | Input |
| LONG | ∗lplData | Read device values | Output |
| LONG | ∗lplRetCode | Returned value of communication function | Output |

### 4.3.6 WriteDeviceRandom (Device random-write)

hResult = object.WriteDeviceRandom( szDeviceList, lSize, ∗lplData,
∗lplRetCode )

| | | | |
|---|---|---|---|
| HRESULT | hResult | Returned value of COM | Output |
| BSTR | szDevice | Device name | Input |
| LONG | lSize | Number of write points | Input |
| LONG | ∗lplData | Written device values | Input |
| LONG | ∗lplRetCode | Returned value of communication function | Output |

### 4.3.7 SetDevice (Device data setting)

hResult = object.SetDevice( szDeviceList, ∗lplData, ∗lplRetCode )

| | | | |
|---|---|---|---|
| HRESULT | hResult | Returned value of COM | Output |
| BSTR | szDeviceList | Device name | Input |
| LONG | ∗lplData | Set data | Input |
| LONG | ∗lplRetCode | Returned value of communication function | Output |

### 4.3.8 GetDevice (Device data acquisition)

hResult = object.GetDevice( szDeviceList, ∗lplData, ∗lplRetCode )

| | | | |
|---|---|---|---|
| HRESULT | hResult | Returned value of COM | Output |
| BSTR | szDeviceList | Device name | Input |
| LONG | ∗lplData | Set data | Output |
| LONG | ∗lplRetCode | Returned value of communication function | Output |

### 4.3.9 ReadBuffer (Buffer memory read)

hResult = object.ReadBuffer( lStartIO, lAddress, lReadSize,
∗lpsData, ∗lplRetCode )

| | | | |
|---|---|---|---|
| HRESULT | hResult | Returned value of COM | Output |
| LONG | lStartIO | First I/O number of module from where values will be read | Input |
| LONG | lAddress | Buffer memory address | Input |
| LONG | lReadSize | Read size | Input |
| SHORT | ∗lpsData | Values read from buffer memory | Output |
| LONG | ∗lplRetCode | Returned value of communication function | Output |

### 4.3.10 WriteBuffer (Buffer memory write)

hResult = object.WriteBuffer( lStartIO, lAddress, lWriteSize,
∗lpsData, ∗lplRetCode )

| | | | |
|---|---|---|---|
| HRESULT | hResult | Returned value of COM | Output |
| LONG | lStartIO | First I/O number of module to where values will be written | Input |
| LONG | lAddress | Buffer memory address | Input |
| LONG | lWriteSize | Write size | Input |
| SHORT | ∗lpsData | Values written to buffer memory | Input |
| LONG | ∗lplRetCode | Returned value of communication function | Output |

### 4.3.11 GetClockDSata (Clock data read)

hResult = object.GetClockData(∗lpsYear, ∗lpsMonth, ∗lpsDay,
∗lpsDayOfWeek, ∗lpsHour, ∗lpsMinute, ∗lpsSecond, ∗lplRetCode )

| | | | |
|---|---|---|---|
| HRESULT | hResult | Returned value of COM | Output |
| SHORT | ∗lpsYear | Read year value | Output |
| SHORT | ∗lpsMonth | Read month value | Output |
| SHORT | ∗lpsDay | Read day value | Output |
| SHORT | ∗lpsDayOfWeek | Read day-of-week value | Output |
| SHORT | ∗lpsHour | Read hour value | Output |
| SHORT | ∗lpsMinute | Read minute value | Output |
| SHORT | ∗lpsSecond | Read second value | Output |
| LONG | ∗lplRetCode | Returned value of communication function | Output |

### 4.3.12 SetClockData (Clock data write)

hResult = object.SetClockData( sYear, sMonth, sDay, sDayOfWeek,
sHour, sMinute, sSecond, *lplRetCode )

| | | | |
|---|---|---|---|
| HRESULT | hResult | Returned value of COM | Output |
| SHORT | sYear | Year value to be written | Input |
| SHORT | sMonth | Month value to be written | Input |
| SHORT | sDay | Day value to be written | Input |
| SHORT | sDayOfWeek | Day-of-week value to be written | Input |
| SHORT | sHour | Hour value to be written | Input |
| SHORT | sMinute | Minute value to be written | Input |
| SHORT | sSecond | Second value to be written | Input |
| LONG | *lplRetCode | Returned value of communication function | Output |

### 4.3.13 GetCpuType (PLC CPU type read)

hResult = object.GetDevice(*szDeviceList, *lplData, *lplRetCode )

| | | | |
|---|---|---|---|
| HRESULT | hResult | Returned value of COM | Output |
| BSTR | *szCpuName | PLC CPU type character string | Output |
| LONG | *lplCpuType | PLC CPU type code | Output |
| LONG | *lplRetCode | Returned value of communication function | Output |

### 4.3.14 SetCpuStatus (Remote control)

hResult = object.SetCpuStatus( lOperation, *lplRetCode )

| | | | |
|---|---|---|---|
| HRESULT | hResult | Returned value of COM | Output |
| LONG | lOperation | Remote run/stop/pause | Input |
| LONG | *lplRetCode | Returned value of communication function | Output |

# 5 SAMPLE PROGRAMS

This chapter shows the sample programs created on VB and VC++.

## 5.1 VB Sample Program

This sample program is designed to read the CPU type of the A3UCPU and read/write device values using the ActACpu control or ActEasyIF control.
This sample program was created on Visual Basic 6.0.

### (1) Using method

Load the form and choose the control to be used.

Clicking the Open Communication button opens the communication line through CPU COM communication.

By clicking the GetCpuType button, the type code of the PLC CPU which is currently connecting the line appears in the "Output Data" text box (top) and the CPU type in the "Output Data" text box (bottom).

Entering the device from where you want to read a value into the "Device Type" text box and clicking the GetDevice button shows the device data in the "Output Data" text box (top).

To write a device value to the A3UCPU, enter the device where you want to write a value into the "Device Type" text box and the device value to be written into the "Device Value" text box and click the SetDevice button.

Clicking the Close Communication button closes the communication line.

If an error occurs at the execution of any function, an error code appears in the "Return Value" text box.
If an error has occurred, refer to "CHAPTER 6 ERROR CODES" and eliminate the error cause.

### (2) Precautions for use of the sample program

(a) When using the ActEasyIF control, set the CPU COM communication information to the logical station number "3" on the communication settings utility before starting the sample program running.

(b) When changing the control used, click the Close Communication button to close the communication line once, then change the control, and open the line again.

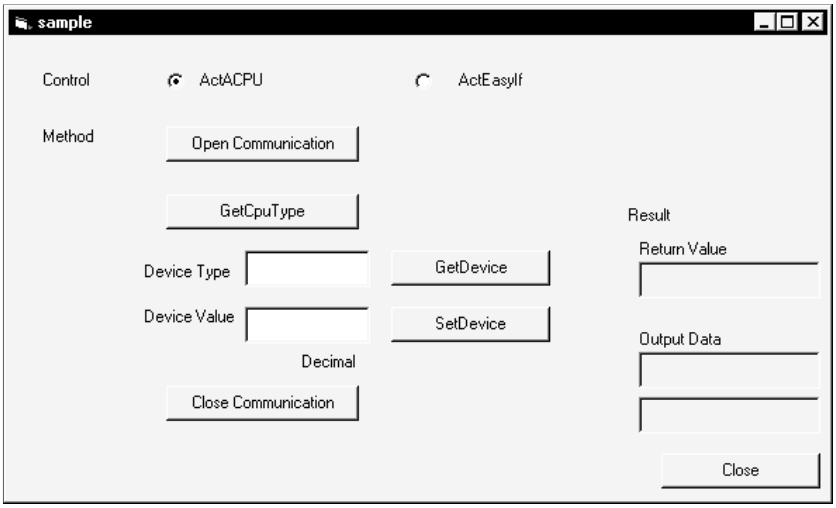
### (3) Sample file list

The sample files are installed into the following folders at default installation.

| | |
|---|---|
| C:\MELSEC\ACT\SAMPLE\VB\Project1.vbp | Project file |
| C:\MELSEC\ACT\SAMPLE\VB\Form1.frm | Source file |
| C:\MELSEC\ACT\SAMPLE\VB\ActDefine.bas | Header file |

### (4) Screen

The sample program screen will be explained.




| Item | | Description |
|---|---|---|
| Control | | Used to choose the control to be used. |
| Open Communication | | Used to open the communication line. |
| GetCpuType | | Used to read the PLC CPU type. |
| Device Type | | Enter the device from/to where a value will be read/written. |
| Device Value | | Enter the device value to be written. |
| Close Communication | | Used to close the communication line. |
| GetDevice | | Used to read the data of the device entered into the "Device Type" text box. |
| SetDevice | | Used to write the data of the device entered into the "Device Type" text box. |
| Return Value | | Shows the result of executing the function. |
| Output Data | Top | Shows the CPU type code and read device value. |
| | Bottom | Shows the CPU type. |

### (5) Program

```
Private Sub CloseButton_Click()

   End

End Sub

Private Sub CloseComButton_Click()
Dim lRet As Long

   On Error GoTo Error 'Error Handler

   'Clear ReturnValue Display
   ResultTxt(0).Text = ""
   ResultTxt(1).Text = ""
   ResultTxt(2).Text = ""

   If Option1(0).Value = True Then
      ' ActACPU Control
      lRet = ActACPU1.Close    'Exec CLOSE Method
   Else
      ' ActEasyIF control
      lRet = ActEasyIF1.Close  'Exec CLOSE Method

   End If
   'Renew ReturnValue
   ResultTxt(0).Text = "0x" + Hex$(lRet)

Exit Sub

Error:
   ErrMsg = Error$(Err)
   MsgBox ErrMsg, ErrType
   End

End Sub

Private Sub GetCpuButton_Click()
Dim lRet As Long
Dim szCpuName As String
Dim lplCpuCode As Long

   On Error GoTo Error 'Error Handler
```

```
   'Clear ReturnValue Display
   ResultTxt(0).Text = ""
   ResultTxt(1).Text = ""
   ResultTxt(2).Text = ""

   If Option1(0).Value = True Then
      ' ActACPU Control
      lRet = ActACPU1.GetCpuType(szCpuName, lplCpuCode)     'Exec Method
   Else
      ' ActEasyIF control
      lRet = ActEasyIF1.GetCpuType(szCpuName, lplCpuCode)   'Exec Method
   End If
   'Renew ReturnValue
   ResultTxt(0).Text = "0x" + Hex$(lRet)
   If lRet = 0 Then
      ResultTxt(1).Text = "0x" + Hex$(lplCpuCode)
      ResultTxt(2).Text = szCpuName
   End If

Exit Sub

Error:
   ErrMsg = Error$(Err)
   MsgBox ErrMsg, ErrType
   End

End Sub

Private Sub GetDeviceButton_Click()
Dim lRet As Long
Dim szDevice As String
Dim lplData As Long

   On Error GoTo Error 'Error Handler

   'Clear ReturnValue Display
   ResultTxt(0).Text = ""
   ResultTxt(1).Text = ""
   ResultTxt(2).Text = ""

   If Text1(0).Text = "" Then
      MsgBox ("Not Enter DeviceType Error")
      Exit Sub
   Else
      szDevice = Text1(0).Text
   End If
```

```
    If Option1(0).Value = True Then
        ' ActACPU Control
        lRet = ActACPU1.GetDevice(szDevice, lplData)     'Exec Method
    Else
        ' ActEasyIF Control
        lRet = ActEasyIF1.GetDevice(szDevice, lplData)   'Exec Method
    End If
    'Renew ReturnValue
    ResultTxt(0).Text = "0x" + Hex$(lRet)
    If lRet = 0 Then
        ResultTxt(1).Text = CStr(lplData) + "(0x" + Hex$(lplData) + ")"
    End If

Exit Sub

Error:
    ErrMsg = Error$(Err)
    MsgBox ErrMsg, ErrType
    End

End Sub

Private Sub OpenComButton_Click()
Dim lRet As Long

    On Error GoTo Error 'Error Handler

    'Clear ReturnValue Display
    ResultTxt(0).Text = ""
    ResultTxt(1).Text = ""
    ResultTxt(2).Text = ""

    If Option1(0).Value = True Then
        ' ActACPU Control
        ' If you don't use default values, please set their properties before OPEN method call.
        '    Example:
        ActACPU1.ActCpuType = CPU_A3UCPU 'Change CPU type => "A3UCPU".....*1
        ActACPU1.ActPortNumber = PORT_2  'Change COM port => "2".....*1
                                 'The other properties are default.
        lRet = ActACPU1.Open    ' Exec OPEN Method
    Else
        ' ActEasyIF Control
        ' If you don't use default values, please set their properties before OPEN method call.
        '    Example:
        ActEasyIF1.ActLogicalStationNumber = 3 'Change the logical station number => "3".....*1
```

```
        lRet = ActEasyIF1.Open  ' Exec OPEN Method

    End If
    ' Renew ReturnValue
    ResultTxt(0).Text = "0x" + Hex$(lRet)

Exit Sub

Error:
    ErrMsg = Error$(Err)
    MsgBox ErrMsg, ErrType
    End

End Sub

Private Sub SetDevice_Click()
Dim lRet As Long
Dim szDevice As String
Dim lplData As Long

    On Error GoTo Error 'Error Handler

    ' Clear ReturnValue Display
    ResultTxt(0).Text = ""
    ResultTxt(1).Text = ""
    ResultTxt(2).Text = ""

    If Text1(0).Text = "" Then
        MsgBox ("Not Enter DeviceType Error")
        Exit Sub
    Else
        If Text1(1).Text = "" Then
            MsgBox ("Not Enter DeviceValue Error")
            Exit Sub
        Else
            If IsNumeric(Text1(1).Text) = False Then
                MsgBox ("Illegal Device Value")
                Exit Sub
            Else
                szDevice = Text1(0).Text
                lplData = CLng(Text1(1).Text)
            End If
        End If
    End If
```

```
    If Option1(0).Value = True Then
        ' ActACPU Control
        lRet = ActACPU1.SetDevice(szDevice, lplData)     'Exec Method
    Else
        ' ActEasyIF Control
        lRet = ActEasyIF1.SetDevice(szDevice, lplData)   'Exec Method
    End If
    ' Renew ReturnValue
    ResultTxt(0).Text = "0x" + Hex$(lRet)

Exit Sub

Error:
    ErrMsg = Error$(Err)
    MsgBox ErrMsg, ErrType
    End

End Sub
```

*1: Property setting may also be made directly on the property page.
When property setting is made on the property, it need not be made in the program.

### (6) For use in another communication path

Run the program after changing the logical station number (only when the utility setting type is used) or the ATC control properties and functions.

## 5.2 VC++ Sample Programs

This section explains the sample programs for VC++ which were created using the dispatch interface and custom interface.
These sample programs were created on Visual C++ 6.0.

### 5.2.1 Dispatch interface

This sample program is designed to read the type of the connection destination CPU and read/write device values using the ActAJ71QE71UDP control or ActEasyIF control on the dispatch interface.

#### (1) Using method

Load the form and choose the control to be used.
Clicking the [Open Communication] button opens the communication line through Ethernet communication.
By clicking the [GetCpuType] button, the type code of the PLC CPU which is currently connecting the line appears in the "Output Data" text box (top) and the CPU type in the "Output Data" text box (bottom).
Entering the device from where you want to read a value into the "Device Name" text box and clicking the [GetDevice] button shows the device data in the "Output Data" text box (top).
To write a device value, enter the device where you want to write a value into the "Device Name" text box and the device value to be written into the "Device Value" text box and click the [SetDevice] button.
Clicking the [Close Communication] button closes the communication line.
If an error occurs at the execution of any function, an error code appears in the "Return Value" text box.
If an error has occurred, refer to "CHAPTER 6 ERROR CODES" and eliminate the error cause.

#### (2) Precautions for use of the sample program

(a) When using the ActEasyIF control, set the Ethernet communication information to the logical station number "2" on the communication settings utility before starting the sample program running.

(b) When changing the control used, click the [Close Communication] button to close the communication line once, then change the control, and open the line again.

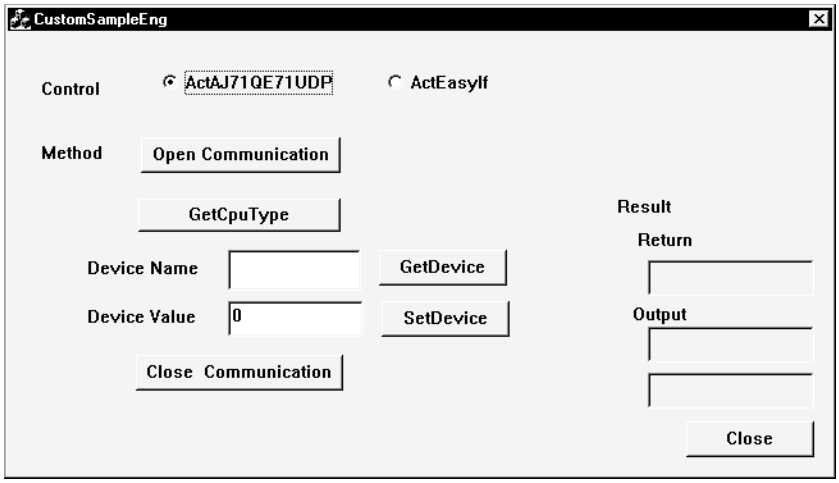
### (3) Sample file list

The sample files are installed into the following folders at default installation.

| | |
|---|---|
| C:\MELSEC\ACT\SAMPLE\VC\SAMPLE\sample.rc | Resource file |
| C:\MELSEC\ACT\SAMPLE\VC\SAMPLE\sample.dsw | Project work space |
| C:\MELSEC\ACT\SAMPLE\VC\SAMPLE\sample.dsp | Project file |

### (4) Screen

The sample program screen will be explained.




| Item | | Description |
|---|---|---|
| Control | | Used to choose the control to be used. |
| Open Communication | | Used to open the communication line. |
| GetCpuType | | Used to read the PLC CPU type. |
| Device Name | | Enter the device from/to where a value will be read/written. |
| Device Value | | Enter the device value to be written. |
| Close Communication | | Used to close the communication line. |
| GetDevice | | Used to read the data of the device entered into the "Device Name" text box. |
| SetDevice | | Used to write the data of the device entered into the "Device Name" text box. |
| Return Value | | Shows the result of executing the function. |
| Output Data | Top | Shows the CPU type code and read device value. |
| | Bottom | Shows the CPU type. |

(5) Program

```
// sampleEngDlg.cpp : implementation file
//

#include "stdafx.h"
#include "sampleEng.h"

/*************************************************/
#include "ActDefine.h"    // ACT Common Macro Header (For Set/Get Property Value)
/*************************************************/

#include "sampleEngDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};
```

```
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSampleEngDlg dialog

CSampleEngDlg::CSampleEngDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CSampleEngDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CSampleEngDlg)
	m_Device = _T("");
	m_DeviceValue = 0;
	m_SelectCntl = 0;
	m_RetVal = _T("");
	m_RetVal2 = _T("");
	m_RetVal3 = _T("");
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CSampleEngDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CSampleEngDlg)
	DDX_Text(pDX, IDC_DEVICE, m_Device);
	DDX_Text(pDX, IDC_DEVVALUE, m_DeviceValue);
	DDX_Radio(pDX, IDC_RADIO1, m_SelectCntl);
	DDX_Text(pDX, IDC_RET, m_RetVal);
	DDX_Text(pDX, IDC_RET2, m_RetVal2);
	DDX_Text(pDX, IDC_RET3, m_RetVal3);
```

```
	DDX_Control(pDX, IDC_ACTEASYIF1, m_ActEasyIF);
	DDX_Control(pDX, IDC_ACTAJ71QE71UDP1, m_ActAJ71QE71UDP);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CSampleEngDlg, CDialog)
	//{{AFX_MSG_MAP(CSampleEngDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_OpenCom, OnOpenCom)
	ON_BN_CLICKED(IDC_GetCpuType, OnGetCpuType)
	ON_BN_CLICKED(IDC_GetDevice, OnGetDevice)
	ON_BN_CLICKED(IDC_SetDevice, OnSetDevice)
	ON_BN_CLICKED(IDC_CloseCom, OnCloseCom)

	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSampleEngDlg message handlers

BOOL CSampleEngDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
```

```
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here

	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CSampleEngDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CSampleEngDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
```

```
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CSampleEngDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

/*****************************************************/
/*  Open Communication Route                         */
/*****************************************************/
void CSampleEngDlg::OnOpenCom()
{
long lRet;
CString MsgStr;
CString Adr = L"1.1.1.2";// HostAddress Example

	CWnd::UpdateData(TRUE);

	// Clear ReturnValue Display
	m_RetVal= "";
	m_RetVal2 = "";
	m_RetVal3 = "";

	try{
		if (m_SelectCntl == 0 ){   // ActAJ71QE71UDP Control
		// If you don't use default values, please set their properties before OPEN method call.
		// (If you call the set-property method after OPEN method, it isn't reflected to the
		// communication.)
		// (You can use methods to set and get the value for all properties.)
		// ---> Example: Change the CPU type to "Q2A-S1" from default value.
		//                Change the HostAddress to "1.1.1.2" from default value.
		//                The other is default.
		  m_ActAJ71QE71UDP.SetActCpuType(CPU_Q2AS1CPU);// Exec set-property
		                                               // method.....*1
		  m_ActAJ71QE71UDP.SetActHostAddress(Adr);     // Exec set-property method .....*1

			lRet = m_ActAJ71QE71UDP.Open();            // Exec OPEN method
		}
		else{                                          // ActEasyIF Control
		// If you don't use default values, please set their properties before OPEN method call.
		// ---> Example: Change the Logical station number to "1" from default value.
			m_ActEasyIF.SetActLogicalStationNumber(1); // Exec set-property method.....*1
			lRet = m_ActEasyIF.Open();                 // Open method exec
		}
		// Renew ReturnValue
```

```
		m_RetVal.Format("0x%08x",lRet);
	}
	catch(COleDispatchException *Exception){
		// OLE IDispatch Interface Error
		MsgStr.LoadString(IDS_STRING103);
		AfxMessageBox(MsgStr, MB_ICONINFORMATION);
		Exception->Delete();
	}

	CWnd::UpdateData(FALSE);

}


/*****************************************************************************/
/*  Get CpuType of the connected CPU (iconfirmation of connecting)  */
/*****************************************************************************/
void CSampleEngDlg::OnGetCpuType()
{
long	lRet;
long	lCpuCode	= 0;
BSTR	szCpuName	= NULL;
CString MsgStr;

	// Clear ReturnValue Display
	m_RetVal= "";
	m_RetVal2 = "";
	m_RetVal3 = "";

	try{
		if (m_SelectCntl == 0 ){   // ActAJ71QE71UDP Control
			lRet = m_ActAJ71QE71UDP.GetCpuType(&szCpuName,&lCpuCode);// Exec
                                                                    //GetCpuType
                                                                    //Method
		}
		else{                                                  // ActEasyIF Control
			lRet = m_ActEasyIF.GetCpuType(&szCpuName,&lCpuCode);// Exec GetCpuType
                                                                  //Method
		}

		if(lRet == 0x00){		// Success
			m_RetVal2.Format("0x%x(%d)",lCpuCode,lCpuCode);	// Cpu Code
			m_RetVal3 = szCpuName;                          // Cpu Name
		}
		// Renew ReturnValue
		m_RetVal.Format("0x%08x",lRet);
	}
```

```
        catch(COleDispatchException *Exception){
                // OLE IDispatch Interface Error
                MsgStr.LoadString(IDS_STRING103);
                AfxMessageBox(MsgStr, MB_ICONINFORMATION);
                Exception->Delete();
        }

        // If the Method has Output value of BSTR type, you have to free the allocated BSTR area.
        ::SysFreeString(szCpuName);

        CWnd::UpdateData(FALSE);
}

/*****************************************************/
/*  Get Device Value                                 */
/*****************************************************/
void CSampleEngDlg::OnGetDevice()
{
long lRet;
long lValue;
CString MsgStr;

        CWnd::UpdateData(TRUE);

        // Clear ReturnValue Display
        m_RetVal= "";
        m_RetVal2 = "";
        m_RetVal3 = "";

        if (m_Device == ""){
                // Not Enter DeviceName Error
                MsgStr.LoadString(IDS_STRING102);
                AfxMessageBox(MsgStr, MB_ICONINFORMATION);
                return;
        }
        try{
                if (m_SelectCntl == 0 ){   // ActAJ71QE71UDP Control
                        lRet = m_ActAJ71QE71UDP.GetDevice(m_Device,&lValue);        // Exec GetDevice
                                                                                    //Method
                }
                else{                                                   // ActEasyIF Control
                        lRet = m_ActEasyIF.GetDevice(m_Device,&lValue);// Exec GetDevice Method
                }
                if(lRet == 0x00){ // Success
                        m_RetVal2.Format("0x%04x(%d)",lValue,lValue); // Device Value
                }
```

```
		// Renew ReturnValue
		m_RetVal.Format("0x%08x",lRet);

	}
	catch(COleDispatchException *Exception){
		// OLE IDispatch Interface Error
		MsgStr.LoadString(IDS_STRING103);
		AfxMessageBox(MsgStr, MB_ICONINFORMATION);
		Exception->Delete();
	}

	CWnd::UpdateData(FALSE);
}

/*****************************************************/
/*  Set Device Value                                 */
/*****************************************************/
void CSampleEngDlg::OnSetDevice()
{
long lValue;
long lRet;
CString MsgStr;

	CWnd::UpdateData(TRUE);

	// Clear ReturnValue Display
	m_RetVal= "";
	m_RetVal2 = "";
	m_RetVal3 = "";

	if (m_Device == ""){
		// Not Enter DeviceName Error
		MsgStr.LoadString(IDS_STRING102);
		AfxMessageBox(MsgStr, MB_ICONINFORMATION);
		return;
	}
	lValue = m_DeviceValue;
	try{
		if (m_SelectCntl == 0 ){   // ActAJ71QE71UDP Control
			lRet = m_ActAJ71QE71UDP.SetDevice(m_Device,lValue);          // Exec SetDevice
                                                                         // Method
		}
		else{                                                  // ActEasyIF Control
			lRet = m_ActEasyIF.SetDevice(m_Device,lValue);          // Exec SetDevice Method
		}
```

```
		// Renew ReturnValue
		m_RetVal.Format("0x%08x",lRet);
	}
	catch(COleDispatchException *Exception){
		// OLE IDispatch Interface Error
		MsgStr.LoadString(IDS_STRING103);
		AfxMessageBox(MsgStr, MB_ICONINFORMATION);
		Exception->Delete();
	}

	CWnd::UpdateData(FALSE);
}


/*****************************************************/
/*  Close Communication Route                        */
/*****************************************************/
void CSampleEngDlg::OnCloseCom()
{
long lRet;
CString MsgStr;

	CWnd::UpdateData(TRUE);

	// Clear ReturnValue Display
	m_RetVal= "";
	m_RetVal2 = "";
	m_RetVal3 = "";

	try{
		if (m_SelectCntl == 0 ){   // ActAJ71QE71UDP Control
			lRet = m_ActAJ71QE71UDP.Close();      // Exec Close Method
		}
		else{                                         // ActEasyIF Control
			lRet = m_ActEasyIF.Close();       // Exec Close Method
		}
		// Renew ReturnValue
		m_RetVal.Format("0x%08x",lRet);
	}
	catch(COleDispatchException *Exception){
		// OLE IDispatch Interface Error
		MsgStr.LoadString(IDS_STRING103);
		AfxMessageBox(MsgStr, MB_ICONINFORMATION);
		Exception->Delete();
	}
```

```
    CWnd::UpdateData(FALSE);

}
```

*1: Property setting may also be made directly on the property page.
When property setting is made on the property, it need not be made in the program.

### (6) For use in another communication path

Run the program after changing the logical station number (only when the utility setting type is used) or the ACT control properties and functions.

## 5.2.2 Custom interface

This sample program is designed to read the type of the connection destination CPU and read/write device values using the ActAJ71QE71UDP control or ActEasyIF control on the custom interface.

### (1) Using method

The using method is the same as that of the sample program for dispatch interface.
Refer to "Section 5.2.1 Dispatch interface, (1) Using method".

### (2) Precautions for use of the sample program

The precautions are the same as those of the sample program for dispatch interface.
Refer to "Section 5.2.1 Dispatch interface, (2) Precautions for use of the sample program".

### (3) Sample file list

The sample files are installed into the following folders at default installation.

| | |
|---|---|
| C:\MELSEC\ACT\SAMPLE\VC\CUSTOMSAMPLE\CustomSample.rc | Resource file |
| C:\MELSEC\ACT\SAMPLE\VC\CUSTOMSAMPLE\CustomSample.dsw | Project work space |
| C:\MELSEC\ACT\SAMPLE\VC\CUSTOMSAMPLE\CustomSample.dsp | Project file |

### (4) Screen

The screen is the same as that of the the sample program for dispatch interface.
Refer to "Section 5.2.1 Dispatch interface, (4) Screen".

(5) Program

```
// CustomSampleEngDlg.cpp : implementation file
//

#include "stdafx.h"
#include "CustomSampleEng.h"

/*************************************************/
#include "ActMulti.h"       // For ActEasyIF Contorol
#include "ActEther.h"       // For Ethernet Communication Contorol
#include "ActDefine.h"      // ACT Common Macro Header
#include "ActMulti_i.c"     // For CustomInterface
#include "ActEther_i.c"     // For CustomInterface
/*************************************************/

#include "CustomSampleEngDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();

// Dialog Data
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CAboutDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:
        //{{AFX_MSG(CAboutDlg)
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

```
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CCustomSampleEngDlg dialog

CCustomSampleEngDlg::CCustomSampleEngDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CCustomSampleEngDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CCustomSampleEngDlg)
	m_Device = _T("");
	m_DeviceValue = 0;
	m_SelectCntl = 0;
	m_RetVal = _T("");
	m_RetVal2 = _T("");
	m_RetVal3 = _T("");
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CCustomSampleEngDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CCustomSampleEngDlg)
	DDX_Text(pDX, IDC_DEVICE, m_Device);
	DDX_Text(pDX, IDC_DEVVALUE, m_DeviceValue);
	DDX_Radio(pDX, IDC_RADIO1, m_SelectCntl);
	DDX_Text(pDX, IDC_RET, m_RetVal);
	DDX_Text(pDX, IDC_RET2, m_RetVal2);
	DDX_Text(pDX, IDC_RET3, m_RetVal3);
```

```
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CCustomSampleEngDlg, CDialog)
	//{{AFX_MSG_MAP(CCustomSampleEngDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_OpenCom, OnOpenCom)
	ON_BN_CLICKED(IDC_GetCpuType, OnGetCpuType)
	ON_BN_CLICKED(IDC_GetDevice, OnGetDevice)
	ON_BN_CLICKED(IDC_SetDevice, OnSetDevice)
	ON_BN_CLICKED(IDC_CloseCom, OnCloseCom)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CCustomSampleEngDlg message handlers

BOOL CCustomSampleEngDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here
```

```
	/*************************************************/
		/* ACT Compornent Instance Create		*/
		// ActAJ71QE71UDP Control
		HRESULT		hr = CoCreateInstance(  CLSID_ActAJ71QE71UDP,
												NULL,
												CLSCTX_INPROC_SERVER,
												IID_IActAJ71QE71UDP,
												(LPVOID*)&mp_IAJ71QE71UDP);
		if(!SUCCEEDED(hr)){
			AfxMessageBox("CoCrateInstance() Failed.");
			exit(0);
		}
		// ActEasyIF Control
		hr = CoCreateInstance(  CLSID_ActEasyIF,
												NULL,
												CLSCTX_INPROC_SERVER,
												IID_IActEasyIF,
												(LPVOID*)&mp_IEasyIF);
		if(!SUCCEEDED(hr)){
			AfxMessageBox("CoCrateInstance() Failed.");
			exit(0);
		}
		/*                                              */
		/*************************************************/

		return TRUE;  // return TRUE  unless you set the focus to a control
}

void CCustomSampleEngDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
		if ((nID & 0xFFF0) == IDM_ABOUTBOX)
		{
			CAboutDlg dlgAbout;
			dlgAbout.DoModal();
		}
		else
		{
			CDialog::OnSysCommand(nID, lParam);
		}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CCustomSampleEngDlg::OnPaint()
```

```
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CCustomSampleEngDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

/*****************************************************/
/*  Open Communication Route                         */
/*****************************************************/
void CCustomSampleEngDlg::OnOpenCom()
{
long	lRet;
HRESULT	hr;
BSTR szAdr = NULL;
wchar_t wsz[] = L"1.1.1.2";	//HostAddress Example
CString MsgStr;

	CWnd::UpdateData(TRUE);

	// Clear ReturnValue Display
	m_RetVal= "";
	m_RetVal2 = "";
	m_RetVal3 = "";
```

```
	if (m_SelectCntl == 0 ){   // ActAJ71QE71UDP Control (Custom Interface)
		// If you don't use default values, please set their properties before OPEN method call.
		// (If you call the set-property method after OPEN method, it isn't reflected to the
		// communication.)
		// (You can use methods to set and get the value for all properties.)
		// ---> Example: Change CPU type to "Q2A-S1" from default value.
		//                Change the Baudrate to 9600bps from default value.
		//                The other is default.
		hr = mp_IAJ71QE71UDP->put_ActCpuType(CPU_Q2AS1CPU);  // Exec set-property method....*1
		if(SUCCEEDED(hr)){     // Compornent Communication is succeeded?
			szAdr = ::SysAllocString(wsz);     // Allocate the BSTR-Type String area.
			                                   // (After use, you have to free it.)
			hr = mp_IAJ71QE71UDP->put_ActHostAddress(szAdr);
			if(SUCCEEDED(hr)){     // Compornent Communication is succeeded?
				hr = mp_IAJ71QE71UDP->Open(&lRet);     // Exec Open Method.....*1
			}
		}
		// Free the allocated area.
		::SysFreeString(szAdr);
	}
	else{                                          // ActEasyIF Control (Custom Interface)
		// If you don't use default values, please set their properties before OPEN method call.
		//---> Example: Change the Logical station number to "2" from default value.
		hr = mp_IEasyIF->put_ActLogicalStationNumber(2);         // Exec set-property method
		if(SUCCEEDED(hr)){     // Compornent Communication is succeeded?
			hr = mp_IEasyIF->Open(&lRet);  // Exec Open Method
		}
	}
	if(SUCCEEDED(hr)){     // Compornent Communication is succeeded?
		// Renew ReturnValue
		m_RetVal.Format("0x%08x",lRet);
	}
	else{     // Failed Compornent Communication
		MsgStr.LoadString(IDS_STRING103);
		AfxMessageBox(MsgStr, MB_ICONINFORMATION);
	}

	CWnd::UpdateData(FALSE);


}

/*****************************************************************************/
/*  Get CpuType of the connected CPU (iconfirmation of connecting)  */
/*****************************************************************************/
```

```
void CCustomSampleEngDlg::OnGetCpuType()
{
long    lRet;
long    lCpuCode        = 0;
BSTR    szCpuName       = NULL;
HRESULT         hr;
CString MsgStr;

        // Clear ReturnValue Display
        m_RetVal= "";
        m_RetVal2 = "";
        m_RetVal3 = "";

        if (m_SelectCntl == 0 ){   // ActAJ71QE71UDP Control (Custom Interface)
                hr = mp_IAJ71QE71UDP->GetCpuType(&szCpuName,&lCpuCode,&lRet); // Exec
                                                                              // GetCpuType
                                                                              // Method
        }
        else{                                                 // ActEasyIF Control (Custom Interface)
                hr = mp_IEasyIF->GetCpuType(&szCpuName,&lCpuCode,&lRet); // Exec GetCpuType
                                                                         // Method
        }
        if(SUCCEEDED(hr)){      // Compornent Communication is succeeded?
                if(lRet == 0x00){ // Success
                        m_RetVal2.Format("0x%x(%d)",lCpuCode,lCpuCode);
                        m_RetVal3 = szCpuName;
                }
                // Renew ReturnValue
                m_RetVal.Format("0x%08x",lRet);
        }
        else{
                MsgStr.LoadString(IDS_STRING103);
                AfxMessageBox(MsgStr, MB_ICONINFORMATION);
        }
        // If the Method has the Output value of BSTR-type, you have to free the allocated area.

        ::SysFreeString(szCpuName);

        CWnd::UpdateData(FALSE);
}

/*****************************************************/
/*  Get Device Value                                 */
/*****************************************************/
void CCustomSampleEngDlg::OnGetDevice()
{
long lRet;
long lValue;
```

```
CString MsgStr;
BSTR szDev = NULL;
HRESULT        hr;

        CWnd::UpdateData(TRUE);

        // Clear ReturnValue Display
        m_RetVal= "";
        m_RetVal2 = "";
        m_RetVal3 = "";

        if (m_Device == ""){
                // Not Enter DeviceName Error
                MsgStr.LoadString(IDS_STRING102);
                AfxMessageBox(MsgStr, MB_ICONINFORMATION);
                return;
        }
        szDev = m_Device.AllocSysString();        // Allocate the BSTR-Type String area.
                                                  // (After use, you have to free it.)
        if (m_SelectCntl == 0 ){   // ActAJ71QE71UDP Control (Custom Interface)
                hr = mp_IAJ71QE71UDP->GetDevice(m_Device.AllocSysString(),&lValue,&lRet); // Exec
                                                                                          //GetDevice
                                                                                          //Method
        }
        else{                                                  // ActEasyIF Control (Custom Interface)
                hr = mp_IEasyIF->GetDevice(m_Device.AllocSysString(),&lValue,&lRet);   // Exec GetDevice
                                                                                       //Method
        }
        if(SUCCEEDED(hr)){      // Compornent Communication is succeeded?
                if(lRet == 0x00){ // Success
                        m_RetVal2.Format("0x%04x(%d)",lValue,lValue); // Device Value
                }
                // Renew ReturnValue
                m_RetVal.Format("0x%08x",lRet);
        }
        else{
                MsgStr.LoadString(IDS_STRING103);
                AfxMessageBox(MsgStr, MB_ICONINFORMATION);
        }
        // Free the allocated area.
        ::SysFreeString(szDev);

        CWnd::UpdateData(FALSE);

}
```

```
/*****************************************************/
/*  Set Device Value                                 */
/*****************************************************/
void CCustomSampleEngDlg::OnSetDevice()
{
long lValue;
long lRet;
CString MsgStr;
HRESULT        hr;
BSTR szDev = NULL;

        CWnd::UpdateData(TRUE);

        // Clear ReturnValue Display
        m_RetVal= "";
        m_RetVal2 = "";
        m_RetVal3 = "";

        if (m_Device == ""){
                // Not Enter DeviceName Error
                MsgStr.LoadString(IDS_STRING102);
                AfxMessageBox(MsgStr, MB_ICONINFORMATION);
                return;
        }
        lValue = m_DeviceValue;
        szDev = m_Device.AllocSysString();        // Allocate the BSTR-Type String area.
                                                  // (After use, you have to free it.)
        if (m_SelectCntl == 0 ){   // ActAJ71QE71UDP Control (Custom Interface)
                hr = mp_IAJ71QE71UDP->SetDevice(m_Device.AllocSysString(),lValue,&lRet); // Exec
                                                                                         //GetDevice
                                                                                         //Method
        }
        else{                                             // ActEasyIF Control (Custom Interface)
                hr = mp_IEasyIF->SetDevice(m_Device.AllocSysString(),lValue,&lRet);      // Exec GetDevice
                                                                                         // Method
        }
        if(SUCCEEDED(hr)){      // Compornent Communication is succeeded?
                // Renew ReturnValue
                m_RetVal.Format("0x%08x",lRet);
        }
        else{
                MsgStr.LoadString(IDS_STRING103);
                AfxMessageBox(MsgStr, MB_ICONINFORMATION);
        }
        // Free the allocated area.
        ::SysFreeString(szDev);
```

```
	CWnd::UpdateData(FALSE);
}

/*****************************************************/
/*  Close Communication Route                        */
/*****************************************************/
void CCustomSampleEngDlg::OnCloseCom()
{
long lRet;
HRESULT	hr;
CString MsgStr;

	CWnd::UpdateData(TRUE);

	// Clear ReturnValue Display
	m_RetVal= "";
	m_RetVal2 = "";
	m_RetVal3 = "";

	if (m_SelectCntl == 0 ){   // ActAJ71QE71UDP Control (Custom Interface)
		hr = mp_IAJ71QE71UDP->Close(&lRet);// Exec Close Method
	}
	else{                                                  // ActEasyIF Control (Custom Interface)
		hr = mp_IEasyIF->Close(&lRet);  // Exec Close Method
	}
	if(SUCCEEDED(hr)){      // Compornent Communication is succeeded?
		// Renew ReturnValue
		m_RetVal.Format("0x%08x",lRet);
	}
	else{
		MsgStr.LoadString(IDS_STRING103);
		AfxMessageBox(MsgStr, MB_ICONINFORMATION);
	}

	CWnd::UpdateData(FALSE);
}


/*****************************************************/
/*  Destroy Window  ( Free ACT Compornent )   */
/*****************************************************/
BOOL CCustomSampleEngDlg::DestroyWindow()
{
	/*****************************************/
	/* Free the Custom-Interface Compornent  */
	mp_IAJ71QE71UDP->Release();
	mp_IEasyIF->Release();
```

```
        /*                                  */
        /*****************************************/

        return CDialog::DestroyWindow();
}
```

*1: Property setting may also be made directly on the property page.
When property setting is made on the property, it need not be made in the program.

### (6) For use in another communication path

Run the program after changing the logical station number (only when the utility setting type is used) or the ACT control properties and functions.

# MEMO

# 6 ERROR CODES

This chapter describes the error codes returned by the ACT controls and the error codes returned by the CPUs, modules and network boards.

## 6.1 Error Codes Returned by the ACT Controls

The following table gives the error codes returned by the ACT controls.

| Error Code | Error Definition | Corrective Action |
|---|---|---|
| 0x01010002 | RUN-time disable error<br>Operation that was performed must not be done during RUN. | Execute after setting to the STOP status. ∗1 |
| 0x01010005 | Sumcheck error<br>Packet sumcheck was abnormal. | Check for system noise. |
| 0x01010010 | PLC No. error<br>Communication could not be made with the specified station number. | Check the station number set on the communication setup utility.<br>Check the station number set to ActStationNumber. |
| 0x01010013 | Other data error<br>Communication cannot be made for some cause. | Check that the system configuration is not an unsupported configuration.<br>Check that the CPU type setting is correct.<br>Exit the program and restart the IBM-PC/AT compatible.<br>Contact our telephone center. |
| 0x01010018 | Remote request error<br>Remote operation is being performed in the path different from the communicating path. | Cancel the remote operation being performed in the other path. |
| 0x01010020 | Link error<br>Link communications could not be made. | Check that reset operation is not performed for the other end of communication, the control station (master station) or the station passed through by routing.<br>Check that the network parameter setting is correct. |
| 0x01800001 | No command error | The corresponding method does not support. |
| 0x01800002 | Memory lock error | Exit the program and restart the IBM-PC/AT compatible. |
| 0x01800003 | Memory securing error | Exit the program and restart the IBM-PC/AT compatible.<br>Exit other programs and secure free memory area. |
| 0x01800004 | DLL load error | Exit the program and restart the IBM-PC/AT compatible.<br>Exit other programs and secure free memory area.<br>Reinstall ACT. |
| 0x01800005 | Resource securing error | Exit the program and restart the IBM-PC/AT compatible.<br>Exit other programs and secure free memory area. |
| 0x01801002 | Multi-line open error | Exit the program and restart the IBM-PC/AT compatible. |
| 0x01801003 | Open not yet executed | Exit the program and restart the IBM-PC/AT compatible. |
| 0x01801005 | Specified port error | Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0x01801006 | Specified module error | Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |

∗1: When the network board is relayed, a time-out error may occur. Check the cable state.

| Error Code | Error Definition | Corrective Action |
|---|---|---|
| 0x01801007 | Specified CPU error | Check the CPU type set to ActCpuType.<br>Check that the system configuration is not an unsupported configuration.<br>Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0x01801008 | Target station access error | Review the target station. |
| 0x0180100C | Registry search failure | Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0x0180100D | GetProcAddress failure | Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0x0180100E | DLL non-load error | Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0x0180100F | Another Object in execution<br>Method cannot be executed because of exclusive control in progress | Execute again after some time. |
| 0x01802001 | Device error<br>The device character string specified in the method is an unauthorised device character string. | Review the device name. |
| 0x01802002 | Device number error<br>The device character string number specified in the method is an unauthorised device number. | Review the device number. |
| 0x01802004 | Sumcheck error<br>The sumcheck value of the received data is abnormal. | Check the module side sumcheck setting.<br>Check the sumcheck property of the control.<br>Check the cable.<br>Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0x01802005 | Size error<br>The number of points specified in the method is unauthorised. | Check the number of points specified in the method.<br>Review the system, e.g. PLC CPU, module setting and cable status.<br>Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0x01802006 | Block number error<br>The block specifying number in the device character string specified in the method is unauthorised. | Review the block specifying number in the device character string specified in the method. |
| 0x01802007 | Receive data error<br>The data received is abnormal. | Review the system, e.g. PLC CPU, module setting and cable status.<br>Check the cable.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x0180200B | PLC type mismatch<br>The CPU type set to the property and the CPU type set on the communication settings utility do not match the CPU type on the other end of communication. | Set the correct CPU type as the CPU type of the property.<br>Set the correct CPU type on the communication settings utility.<br>Review the system, e.g. PLC CPU, module setting and cable status. |

| Error Code | Error Definition | Corrective Action |
|---|---|---|
| 0x01802016 | Station number specifying error<br>The method does not support the operation performed for the specified station number. | Review the station number. |
| 0x0180201C | Clock data write error<br>Write of clock data failed.<br>Clock data cannot be written since the PLC CPU is during RUN. | Place the PLC CPU in the STOP status. |
| 0x01802020 | First I/O number error<br>The first I/O number specified in the method is an unauthorised value. | Check the value of the first I/O number specified in the method.<br>Using the GPP function, check the PLC CPU parameters (I/O assignment).<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x01802021 | First address error<br>The buffer address specified in the method is an unauthorised value. | Check the value of the buffer address specified in the method.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x01802038 | Clock data read/write error<br>The clock data read/write method was executed for the PLC CPU which does not have the clock devices. | Do not execute clock data read/write. |
| 0x01808001 | Duplex open error | Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808002 | Channel number specifying error<br>The port number set to the property and the port number set on the communication settings utility are unauthorised values. | Set the correct value to the port number of the property.<br>Make communication settings again on the communication settings utility. |
| 0x01808003 | Driver not yet started<br>The network board driver is not started. | Start the driver. |
| 0x01808005 | MUTEX generation error<br>Creation of MUTEX to exercise exclusive control failed. | Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0x01808007 | Socket object generation error<br>Creation of the Socket object failed. | Check for a running application which uses the same port number.<br>Retry after changing the port number value of the property.<br>Retry after changing the port number value on the communication settings utility.<br>Make Ethernet board and protocol settings on the control panel of the OS.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808008 | Port connection error<br>Establishment of connection failed.<br>The other end does not respond. | Review the IP address and port number values of the properties.<br>Review the port number value on the communication settings utility.<br>Review the system, e.g. PLC CPU, module setting and cable status.<br>Exit the program and restart the IBM-PC/AT compatible. |

| Error Code | Error Definition | Corrective Action |
|---|---|---|
| 0x01808009 | COM port handle error<br>The handle of the COM port cannot be acquired.<br>The COM port objet cannot be copied.<br>The SOCKET object cannot be copied. | Check for an application which uses the COM port.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x0180800A | Buffer size setting error<br>Setting of the COM port buffer size failed. | Check for an application which uses the COM port.<br>Make COM port setting on the control panel of the OS.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x0180800B | DCB value acquisition error<br>Acquisition of the COM port DCB value failed. | Check for an application which uses the COM port.<br>Make COM port setting on the control panel of the OS.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x0180800C | DCB setting error<br>Setting of the COM port DCB value failed. | Check for an application which uses the COM port.<br>Make COM port setting on the control panel of the OS.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x0180800D | Time-out value setting error<br>Setting of the COM port time-out value failed. | Review the time-out value of the property.<br>Review the time-out value on the communication settings utility.<br>Check for an application which uses the COM port.<br>Make COM port setting on the control panel of the OS.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x0180800E | Shared memory open error<br>Open processing of shared memory failed. | Check whether the ladder logic test function (LLT) has started.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808101 | Duplex close error | Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808102 | Handle close error<br>Closing of the COM port handle failed. | Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808103 | Driver close error<br>Closing of the driver handle failed. | Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808201 | Send error<br>Data send failed. | Review the system, e.g. PLC CPU, module setting and cable status.<br>Make COM port setting on the control panel of the OS.<br>Make Ethernet board and protocol settings on the control panel.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808202 | Send data size error<br>Data send failed. | Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808203 | Queue clear error<br>Clearing of the COM port queue failed. | Exit the program and restart the IBM-PC/AT compatible.<br>Perform Close once and execute Open again. |
| 0x01808301 | Receive error<br>Data receive failed. | Review the system, e.g. PLC CPU, module setting and cable status.<br>Review the time-out value of the property.<br>Review the time-out value on the communication settings utility.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808304 | Receive buffer size shortage<br>Receive data was larger than the receive buffer size prepared for the system. | Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808401 | Control error<br>Changing of the COM port communication control failed. | Exit the program and restart the IBM-PC/AT compatible. |

| Error Code | Error Definition | Corrective Action |
|---|---|---|
| 0x01808403 | Signal line specifying error<br>Changing of the COM port communication control failed. | Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808404 | Open not yet executed | Execute Open.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808405 | Communication parameter error<br>The data bit and stop bit combination of the properties is unauthorised. | Review the data bit and stop bit values of the properties.<br>Set them again on the communication settings utility. |
| 0x01808406 | Baudrate value specifying error<br>The baudrate of the property is unauthorised. | Review the baudrate value of the property.<br>Set it again on the communication settings utility. |
| 0x01808407 | Data length error<br>The data bit value of the property is unauthorised. | Review the data bit value of the property.<br>Set it again on the communication settings utility. |
| 0x01808408 | Parity specifying error<br>The parity value of the property is unauthorised. | Review the parity value of the property.<br>Set it again on the communication settings utility. |
| 0x01808409 | Stop bit specifying error<br>The stop bit value of the property is unauthorised. | Review the stop bit value of the property.<br>Set it again on the communication settings utility. |
| 0x0180840A | Communication control setting error<br>The control value of the property is unauthorised. | Review the control value of the property.<br>Set it again on the communication settings utility. |
| 0x0180840B | Time-out error<br>Though the time-out period had elapsed, data could not be received. | Review the time-out value of the property.<br>Set it again on the communication settings utility.<br>Review the system, e.g. PLC CPU, module setting and cable status.<br>Perform Close once and execute Open again.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x0180840C | Connect error | Exit the program and restart the IBM-PC/AT compatible. |
| 0x0180840D | Duplex connect error | Exit the program and restart the IBM-PC/AT compatible. |
| 0x0180840E | Attach failure<br>Attaching of the socket object failed. | Exit the program and restart the IBM-PC/AT compatible. |
| 0x0180840F | Signal line status acquisition failure<br>Acquisition of the COM port signal line status failed. | Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808410 | CD signal line OFF<br>The CD signal on the other end of communication is in the OFF status. | Review the system, e.g. PLC CPU, module setting and cable status.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808501 | USB driver load error<br>Loading of the USB driver failed. | Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0x01808502 | USB driver connect error<br>Connection of the USB driver failed. | Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0x01808503 | USB driver send error<br>Data send failed. | Review the system, e.g. PLC CPU, module setting and cable status.<br>Make USB setting on the control panel (device manger) of the OS.<br>Exit the program and restart the IBM-PC/AT compatible. |

| Error Code | Error Definition | Corrective Action |
|---|---|---|
| 0x01808504 | USB driver receive error<br>Data receive failed. | Review the system, e.g. PLC CPU, module setting and cable status.<br>Make USB setting on the control panel (device manger) of the OS.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808506 | USB driver initialisation error<br>Initialisation of the USB driver failed. | Make USB setting on the control panel (device manger) of the OS.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x01808507 | Other USB error<br>Error related to data send/receive occurred. | Disconnect the cable once, then reconnect.<br>Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0x04000004 | Internal server DLL load error<br>Start of the internal server failed. | Check for the deleted or moved installation file of ACT.<br>Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0x10000002 | Start of communication DLL of ACT failed. | Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0x10000003 | Open failed. (DiskDrive) | Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0x10000004 | Duplex open error | Exit the program and restart the IBM-PC/AT compatible. |
| 0x1000000C | Execution failed since another application or thread is making a request. | Execute again after some time.<br>Perform programming according to the multithread rules of COM and ActiveX.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x10000011 | Memory securing error | Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0x10000012 | Open not yet executed | Exit the program and restart the IBM-PC/AT compatible. |
| 0x10000017 | The specified size (number of devices) is unauthorised. | Check the number of points specified in the method.<br>Exit the program and restart the IBM-PC/AT compatible. |
| 0x10000018 | There is no registered device. | Exit the program and restart the IBM-PC/AT compatible. |
| 0x1000001E | Registry search failed. | Exit the program and restart the IBM-PC/AT compatible.<br>Exit other programs and secure free memory area.<br>Reinstall ACT. |
| 0x10000032 | Specified device error | Review the specified device data.<br>Exit the program and restart the IBM-PC/AT compatible.<br>Exit other programs and secure free memory area. |
| 0x10000033 | Specified device range error | Review the specified device data.<br>Exit the program and restart the IBM-PC/AT compatible.<br>Exit other programs and secure free memory area. |
| 0x10000040 | Server start failed. | Exit the program and restart the IBM-PC/AT compatible. |
| 0xF0000001 | No-license error<br>The license is not given to the IBM-PC/AT compatible. | Using the license FD, give the license to the IBM-PC/AT compatible. |
| 0xF0000002 | Set data read error<br>Reading of the set data of the logical station number failed. | Specify the correct logical station number.<br>Set the logical station number on the communication settings utility. |

| Error Code | Error Definition | Corrective Action |
|---|---|---|
| 0xF0000003 | Already open error<br>The Open method was executed in the open status. | When changing the communication target CPU, execute the Open method after performing Close. |
| 0xF0000004 | Not yet open error<br>The Open method is not yet executed. | After executing the Open method, execute the corresponding method. |
| 0xF0000005 | Initialisation error<br>Initialisation of the object possessed internally in ACT failed. | Exit the program and restart the IBM-PC/AT compatible.<br>Reinstall ACT. |
| 0xF0000006 | Memory securing error<br>Securing of ACT internal memory failed. | Exit the program and restart the IBM-PC/AT compatible.<br>Exit other programs and secure free memory area. |
| 0xF0000007 | Function non-support error<br>The method does not support. | The corresponding method does not support. |
| 0xF1000001 | Character code conversion error<br>Character code conversion (UNICODE→ASCII code or ASCII code→UNICODE) failed. | Check the character string specified in the method.<br>The ASCII character string acquired from the PLC CPU is abnormal.<br>Review the system, e.g. PLC CPU, module setting and cable status.<br>Exit the program and restart the IBM-PC/AT compatible.<br>Retry the GetCpuType method. |
| 0xF1000002 | First I/O number error<br>The first I/O number specified is an unauthorised value.<br>A matching first I/O number does not exist. | Check the value of the first I/O number specified in the method.<br>Using the GPP function, check the PLC CPU parameters (I/O assignment). |
| 0xF1000003 | Buffer address error<br>The buffer address specified is an unauthorised value.<br>The buffer address is outside the range. | Check the value of the buffer address specified in the method. |
| 0xF1000004 | Buffer read size error<br>As a result of buffer read, the specified size could not be acquired. | Perform reopen processing.<br>Review the system, e.g. PLC CPU, module setting and cable status.<br>Retry.<br>Exit the program. |
| 0xF1000005 | Size error<br>The size specified in the read/write method is abnormal.<br>The read/write first number plus size exceeds the device or buffer area. | Check the size specified in the method. |
| 0xF1000006 | Operation error<br>The operation specified for remote operation is an abnormal value. | Check the operation specifying value specified in the method. |
| 0xF1000007 | Clock data error<br>The clock data is abnormal. | Check the clock data specified in the method.<br>Set the correct clock data to the clock data of the PLC CPU. |

## 6.2 Error Codes Returned by the CPUs, Modules and Network Boards

This section explains the error codes returned by the CPUs, modules and network boards.

| Error Code | Error Occurrence Location |
|---|---|
| 0x01010000 to 0x0101FFFF | QCPU (A mode), ACPU, motion controller CPU |
| 0x01020000 to 0x0102FFFF | QnACPU |
| 0x01030000 to 0x0103FFFF | C24 |
| 0x01040000 to 0x0104FFFF | QC24(N) |
| 0x01050000 to 0x0105FFFF | E71＊1 |
| 0x01060000 to 0x0106FFFF | QE71＊1 |
| 0x01070000 to 0x0107FFFF | MELSECNET/10 board, MELSECNET(II) board, CC-Link board, CPU board, AF board |
| 0x01090000 to 0x0109FFFF | FXCPU |
| 0x010A0000 to 0x010AFFFF | QCPU (Q mode) |
| 0x010B0000 to 0x010BFFFF | Q series-compatible C24 |
| 0x010C0000 to 0x010CFFFF | Q series-compatible E71 |

＊1 : If the 4 lower digits of the error code which occurred during E71 or QE71 communication is not indicated in the E71 or QE71 manual, check whether the DIP switch (SW2) on the front of of the E71 or QE71 module is set as indicated below.
If the DIP switch is not set correctly, a difference has occurred in the packet format (ASCII/binary) and therefore the error code returned from the module cannot be recognized correctly.

| Communication | | SW2 Switch Setting |
|---|---|---|
| E71 | TCP/IP | ON (ASCII mode) |
| | UDP/IP | OFF (binary mode) |
| QE71(TCP/IP) | | ON (ASCII mode) |

**POINT**

The error codes returned by the CPUs, modules and network boards enter the 4 lower digits of the above error codes.
For details of the above error codes, check the error code in the 4 lower digits and refer to the manual of the corresponding CPU, module or network board.

## 6.3 HRESULT Type Error Codes

Normally, the ActiveX control returns the HRESULT type returned value. So does the ACT control.
When the custom interface is used, the returned value is equivalent to the returned value of method API.
When the dispatch interface is used, the HRESULT type returned value can be acquired by performing exception processing.
The following table indicates the HRESULT type returned values of the ACT controls.

| Returned Value | Termination Status | Description |
|---|---|---|
| S_OK | Normal termination | Function processing terminated normally. |
| S_FALSE | Normal termination | Function processing (as ActiveX control) terminated normally, but operation (access to PLC) failed. |
| E_POINTER | Abnormal termination | The pointer passed to the function is abnormal. |
| E_OUTOFMEMORY | Abnormal termination | Memory securing or object creation failed. |

**POINT**

If exception processing for acquiring the HRESULT type returned value has not been performed, the dispatch interface shows the error dialog box on the OS level when E_POINTER (E_XXXXX defined returned value) or the like is returned from the ACT control.

# MEMO

# Type SW0D5C-ACT-E ActiveX Communication Support Tool Programming Manual

| | |
|---|---|
| MODEL | SW0D5C-ACT-E-P-E |
| MODEL CODE | 13JF62 |
| SH(NA)-080078-A(0004)MEE | |

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE : MITSUBISHI DENKI BLDG MARUNOUCHI TOKYO 100-8310 TELEX : J24532 CABLE MELCO TOKYO
NAGOYA WORKS : 1-14 , YADA-MINAMI 5 , HIGASHI-KU, NAGOYA , JAPAN