JTEKT PLC

CMP-LINK provider

Version 1.1.1

User's Guide

April 14, 2023

NOTE:
This User's Guide is machine-translated.

[Revision History]

Version	Date	Content
1.0.0	2018-6-1	First edition
1.0.1	2018-11-5	Memory leakage correction during AddController
	2020-5-7	5. Addition of environment setup for application development Added the module used when checking the operation of Ethernet connected.
1.1.0	2021-8-27	Corresponds to extended addressing added by PC10 When Ethnet is connected and Bit type device is specified, specifying other than PrgNo1 is prohibited. Fixed a bug where data could not be read during packet splitting. Added range checking for addresses. Fixed a bug where a single-digit StationNumber cannot be specified during serial communication.
1.1.1	2021-12-21	Correcting error detection errors
	2023-04-14	Manual modification

[Compatible models]

Model	Version	Notes
PC10G series-		
PC3J series-		PC10 extension area cannot be accessed.
TOYOPUC-Nano		

[Operation check model]

Model	Version	Notes
PC10G-CPU (Model Code: TUC-6353)		
TOYOPUC-Nano (Model Code: TUC-6941)		

Table of contents

1. Introduction 5
2. Provider Overview 6
2.1. Overview 6
2.2. Method Properties 6
2.2.1. CaoWorkspace::AddController method 6
2.2.1.1. Conn Optional 7
2.2.1.2. StationNumber Optional 8
2.2.2. CaoController::GetVariableNames Properties 8
2.2.3. CaoController::AddVariable method 8
2.2.4. CaoController::Execute method 8
2.2.5. CaoVariable::get_Value property 9
2.2.6. CaoVariable::put_Value property 9
3. Variable List 10
3.1. CaoController class- 10
3.1.1. @MAKER_NAME 10
3.1.2. @VERSION 11
3.1.3. @PLC_MODE 11
3.1.1. <??> (PLC Device) 13
3.1.1.1. Serial connection 17
3.1.1.1.1. PrgNo Optional 17
3.1.1.1.2. Address Optional 17
3.1.1.1.3. VT Optional 25
3.1.1.1.4. Interval Optional 26
3.1.1.2. Ethernet connection 27
3.1.1.2.1. PrgNo Optional 27
3.1.1.2.2. Address Optional 27
3.1.1.2.3. VT Optional 34
4. Command Reference 36
4.1. CaoController class- 36
4.1.1. CaoController::Execute ("GetStatus") Command 36
4.1.2. CaoController::Execute ("StopScan") Command 36
4.1.3. CaoController::Execute ("AwakeStopScan ") Commands 37
4.1.4. CaoController::Execute ("StartScan") Command 37
4.1.5. CaoController::Execute ("Reset") Command 38
5. Sample program 39
6. Error code 41

7. Environment setup for application development 42
7.1. Preparation 42
7.2. Serial Communication Settings 43
7.3. Ethernet communication settings 46

1. Introduction

This document is a user's guide for CAO providers that write/read data to/from PLC made by JTEKT. CAO providers (CaoProvJTEKTCMP-LINK.dll) covered in this document are called CMP-LINK providers.

Chapter2 provides an introduction to CMP-LINK providers, as well as descriptions of variables/commands.2

The correspondence status and data string of communication commands implemented by CMP-LINK providers depends on the communication destination PLC. Refer to "Computer Link Function" in the Operation Manual of each PLC for further information on communication.

Figure 1-1. Configuration Diagram

Figure 1-2 shows the correspondence between this provider and devices.

(* This is an example. It does not mean all.)

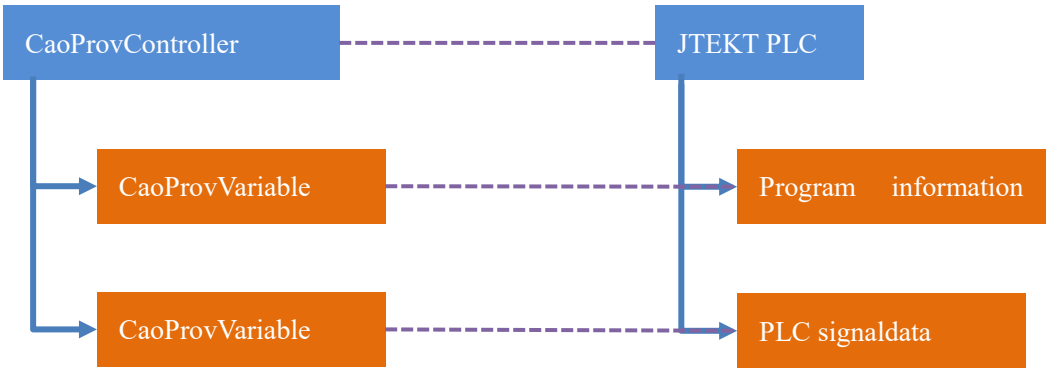


Figure 1-2: Correspondence between provider configuration and device information

2. Provider Overview

2.1. Overview

CMP-LINK provider is a CAO provider that connects TCP/IP or serial to PLC made by JTEKT and writes/reads data using the computer link function.

The file format is DLL(Dynamic Link Library) and is dynamically loaded from CAO engine. To use.CMP-LINK providers, you must install ORiN2SDK or manually register the registry as described in the table below.

Tabular 2-1 CMP-LINK Providers

File name	CaoProvJTEKTCMP-LINK.dll
ProgID	CaoProv.JTEKT.CMP-LINK
Registry registration[1]	Regsvr32 CaoProvJTEKTCMP-LINK.dll
Deleting registry entries	Regsvr32 /u CaoProvJTEKTCMP-LINK.dll

2.2. Method Properties

2.2.1. CaoWorkspace::AddController method

CMP-LINK provider refers to the connection parameters for communication during AddController and connects the communication.

Format AddController (<bstrCtrlName:BSTR>,<bstrProvName:BSTR>,
<bstrPcName:BSTR>, [<bstrOption:BSTR>])

< bstrCtrlName >	:	[in] Controller name
< bstrProvName >	:	[in] Provider name. Fixed value = "CaoProv.JTEKT.CMP-LINK"
<bstrPcName>	:	[in] Provider execution machine name (not used)
<bstrOption>	:	[in] Option string

Examples of use

```
Using(CCaoEngine engine = new CCaoEngine())
{
    CCaoController ctrl = engine.Workspaces[0].AddController(
        "PLC-Link",
        "CaoProv.JTEKT.CMP-LINK",
        Null,
        "Conn=TCP:192.168.0.1:5001,Timeout=3000,ConnTimeout=3000");
}
```

The following lists the option strings.

Tabular 2-2 CaoWorkspace::AddController Option-String

Option	Meaning
Conn = <connect parameter>	Required. Set the communication mode and its connection parameters. (see 2.2.1.1)
SumCheck[=<TRUE/FALSE>]	Enables or disables sum checks for serial connection. (Default: TRUE)
StationNumber [=<station number>]	Station number specification for serial connection. 2 octal characters Specify 00 to 37. (see 2.2.1.2)
ConnTimeout[=<timeout>]	Specifies the timeout (in milliseconds) for connection. (default: 3000)
Timeout[=<timeout>]	Specifies the timeout (in milliseconds) for TCP communication. (default: 3000)

2.2.1.1. Conn Optional

Conn optional connect parameter strings are shown below. Where square brackets ("[]") denote optional characters. The underlined part in the explanation of each parameter indicates the default value when the option is not specified.

[Serial Device]

"Conn=com:<COM Port>[:<BaudRate>[:<Parity>:<DataBits>:<StopBits>]]"

<COM Port> : COM port number . '1'-COM1,'2'-COM2, ...

<BaudRate> : Communication speed
300,600,1200,2400,4800,9600,19200,38400,57600,115200

<Parity> : Parity. 'E'-EVEN

<DataBits> : Data bits. '7'-7bit,'8'-8bit

<StopBits> : Stop Bits. '1'-1bit,'2'-2bit

[Ethernet Device]

TCP/IP

"Conn=TCP:<Dest IP Address>:<Dest Port No>[:<Src IP Address>:<Src Port No>]"

< Dest IP Address > : IP to connect to.

< Dest Port No >	:	Number of the specific port
< Src IP Address >	:	Own IP address (Used when the communication setting of PLC is TCP destination specified passive.)
< Src Port No >	:	Own Port Number (Used when the communication setting of PLC is TCP destination specified passive.)

2.2.1.2. StationNumber Optional

The station number can be specified when a serial port is connected by specifying StationNumber option. If not specified, the value is 00 (not used when TCP/IP is connected).

The specified station number is assigned to the communication command during serial connection. For more information about communication commands, refer to the User's Manual of each PLC.

2.2.2. CaoController::GetVariableNames Properties

Table 3-1 CaoController Class-Variable List31

2.2.3. CaoController::AddVariable method

Table 3-1 CaoController Class-Variable List31

Format AddVariable (<bstrVariableName:BSTR>, [<bstrOption:BSTR>])

< bstrVariableName >	:	[in] Variable Name
<bstrOption>	:	[in] Option string

Examples of use

```
CCaoVariable variable = ctrl.AddVariable("VAR1", "DeviceType=D,PrgNo=1,Address=0,VT=UI2");
```

2.2.4. CaoController::Execute method

Execute method of CaoController is the method for executing the command. For details on each command, see the command reference.Command Reference

Format Execute (<bstrCommandName: VT_BSTR, [<vntParam : VT_VARIANT>])

BstrCommandName : [in] Command name

VntParam : [in] Parameters

2.2.5. CaoVariable::get_Value property

Send the command (I/O register byte read) to read from the device to be accessed to the size specified by the option. Returns the result of reading converted to the data type specified by the option.

Examples of use

```
Object value = variable.Value;
```

2.2.6. CaoVariable::put_Value property

Converts the value passed in the argument according to the option specification, and then sends a command (I/O register byte write) to write to the device to be accessed.

If you write to an address that is used in the system, put_Value succeeds, but the value might not change.

Examples of use

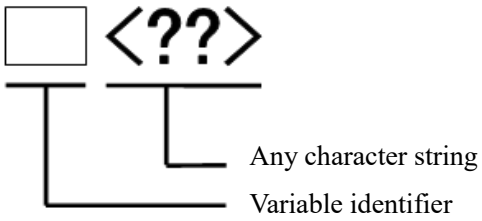
```
Variable.Value = 100;
```

3. Variable List

Defines a list of variables available for each class. A variable refers to an object of CaoVariable class. Any character string can be added to register multiple variables (useful when changing only options, etc.).

The format for assigning an arbitrary string to a variable name is shown below.

Multiple Variable Common Specification Format



3.1. CaoController class-

Table 3-1 CaoController Class-Variable List31

Variable Name	Description	Attribute		Reference
		Get	Put	
@MAKER_NAME	Obtain the manufacturer's name ("JTEKT").	✓	-	P.10
@VERSION	Get the provider version.	✓	-	P.11
@PLC_MODE	Gets CPU status.	✓	-	P.11
<??>	Accesses the devices in PLC.	✓	✓	P.13

3.1.1. @MAKER_NAME

Obtain the manufacturer name.

Option

Not found.

Data type

Data type	Description
VT_BSTR	"JTEKT" is stored.

3.1.2. @VERSION

Get the version of the provider.

Option

Not found.

Data type

Data type	Description
VT_BSTR	Contains the provider version.

3.1.3. @PLC_MODE

Gets CPU status.

Option

Not found.

Data type

Data type		Description
VT_I2 | VT_ARRAY		Stores the retrieval result as an array of 0/1 (0: OFF, 1: ON). 0 to 11 are common for serial connection and Ethernet connection. 12 and later exist only when Ethernet is connected.
0		PC3 mode
1		I/O monitor user mode
2		In debug mode
3		Pseudo-stop in progress
4		STOP REQ IN CONTINUING
5		Stopping
6		RUN
7		With memory card
8		I/O assignment parameter changed
9		Alarm
10		Minor failure
11		Serious failure
Thereafter, it exists only when Ethernet is connected.		
12		Prohibition of programming and supplementary information writing
13		Memory card operation
14		Write I/O prohibited
15		Disable reading of I/O

16		System memory write disabled
17		System memory read disabled
18		1 instruction step
19		1 Prox step
20		1 scan step
21		Trigger detection
22		Enable detection
23		Periodic sampling trace
24		Scan sampling trace
25		Trace
26		STATUS LATCH SETTING
27		Remote RUN setting
28		I/O offline
29		Writing during RUN
30		Write error during RUN
31		Program and ancillary information write permission
32		Program 1 in operation
33		Program 2 in operation
34		Program 3 in operation

3.1.1. <??> (PLC Device)

Accesses the devices in PLC. The data type depends on the specification. Define a list of addresses in the provider-accessible PLC as follows:

List of addresses in 3-2 PLC

Name	Domain	Size	Address range	Program-number [1]	Device-type [2]	Addresses (bits) [3]	Address-byte [4]	Addresses (words) [5]
Edge detection	Basic bit area	Bit	P0000~P01FF	**1**, 2, 3	P	0~1FF	0~3F	0~1F
Edge-detection for PC10	Basic bit area for PC10	Bit	P1000~P17FF	**1**, 2, 3	P1	-	0~FF	0~7F
Keep relay	Basic bit area	Bit	K0000~K02FF	**1**, 2, 3	K	0~2FF	0~5F	0~2F
Special relay	Basic bit area	Bit	V0000~V00FF	**1**, 2, 3	V	0~0FF	0~1F	0~F
Special relays for PC10	Basic bit area for PC10	Bit	V1000~V17FF	**1**, 2, 3	V1	-	0~FF	0~7F
Timer/counter	Basic bit area	Bit	T/C0000~T/C01FF	**1**, 2, 3	T / C	0~1FF	0~3F	0~1F
PC10 timers/counters	Basic bit area for PC10	Bit	T/C1000~T/C17FF	**1**, 2, 3	T1 / C1	-	0~FF	0~7F
Link relay	Basic bit area	Bit	L0000~L07FF	**1**, 2, 3	L	0~7FF	0~FF	0~7F
Link-relay for PC10	Basic bit area for PC10	Bit	L1000~L1FFF	**1**, 2, 3	L1	-	0~1FF	0~FF
Link-relay for PC10	Basic bit area for PC10	Bit	L2000~L2FFF	**1**, 2, 3	L2	-	0~1FF	0~FF
Input/Output	Basic bit area	Bit	X/Y0000~X/Y07FF	**1**, 2, 3	X / Y	0~7FF	0~FF	0~7F
Internal relay	Basic bit area	Bit	M0000~M07FF	**1**, 2, 3	M	0~7FF	0~FF	0~7F

Name	Domain	Size	Address range	Program-number [1]	Device-type [2]	Addresses (bits) [3]	Address-byte [4]	Addresses (words) [5]
PC10 internal relays	Basic bit area for PC10	Bit	M1000~M17FF	**1**, 2, 3	M1	-	0~FF	0~7F
Special register	Basic word area	Word	S0000~S03FF	**1**, 2, 3	S	-	-	0~3FF
Special register for PC10	Basic word area for PC10	Word	S1000~S13FF	**1**, 2, 3	S1	-	-	0~3FF
Current value register	Basic word area	Word	N0000~N01FF	**1**, 2, 3	N	-	-	0~1FF
Current-value register for PC10	Basic word area for PC10	Word	N1000~N17FF	**1**, 2, 3	N1	-	-	0~7FF
Link register	Basic word area	Word	R0000~R07FF	**1**, 2, 3	R	-	-	0~7FF
Data register	Basic word area	Word	D0000~D2FFF	**1**, 2, 3	D	-	-	0~FFF
Data register	Basic word area	Word	D1000~D1FFF	**1**, 2, 3	D1	-	-	0~FFF
Data register	Basic word area	Word	D2000~D2FFF	**1**, 2, 3	D2	-	-	0~FFF
Extended edge	Expansion bit area 1	Bit	EP0000~EP0FFF	**0**	EP	-	0~1FF	0~FF
Extended keep-relay	Expansion bit	Bit	EK0000~EK0FFF	**0**	EK	-	0~1FF	0~FF

Name	Domain	Size	Address range	Program-number [1]	Device-type [2]	Addresses (bits) [3]	Address-byte [4]	Addresses (words) [5]
	area 1							
Extended special relay	Expansion bit area 1	Bit	EV0000~EV0FFF	**0**	EV	-	0~1FF	0~FF
Extended Timer/Counter	Expansion bit area 1	Bit	ET/EC0000~ET/EC07FF	**0**	ET/EC	-	0~FF	0~7F
Extended link relay	Expansion bit area 1	Bit	EL0000~EL1FFF	**0**	EL	-	0~3FF	0~1FF
Extended input	Expansion bit area 1	Bit	EX/EY0000~EX/EY07FF	**0**	EX/EY	-	0~FF	0~7F
Extended internal relay	Expansion bit area 1	Bit	EM0000~EM1FFF	**0**	EM	-	0~3FF	0~1FF
Extended special register	Expansion word area 1	Word	ES0000~ES07FF	**0**	ES	-	-	0~7FF
Expansion current value	Expansion word area 1	Word	EN0000~EN07FF	**0**	EN	-	-	0~7FF
Expansion set value register	Expansion word area 1	Word	H0000~H07FF	**0**	H	-	-	0~7FF
Extended input/output	Expansion bit area 2	Bit	GX/GY0000~GX/GYFFFF	**7**	GX / GY	-	0~1FFF	0~FFF
Extended internal relay	Expansion bit area 2	Bit	GM0000~GMFFFF	**7**	GM	-	0~1FFF	0~FFF

Name	Domain	Size	Address range	Program-number [1]	Device-type [2]	Addresses (bits) [3]	Address-byte [4]	Addresses (words) [5]
Extended data register	Expansion word area 2	Word	U000000~U007FFF	**8**	U	-	-	0~7FFF
			U000000~U01FFFF	*	U	-	-	0~1FFFF
Expand file register	Expansion word area 3	Word	EB00000~EB07FFF	**9**	EB	-	-	0~7FFF
			EB08000~EB0FFFF	A	EB	-	-	0~7FFF
			EB10000~EB17FFF	B	EB	-	-	0~7FFF
			EB18000~EB1FFFF	C	EB	-	-	0~7FFF
			EB00000~EB3FFFF	*	EB	-	-	0~3FFFF
Flash register	Expansion word area 4	Word	FR000000~FR1FFFFF	*	FR	-	-	0~1FFFFF

[1] Corresponds to PrgNo option.

[2] Corresponds to DeviceType option.

[3] Corresponds to Address option that is specified when accessing bit address.

[4] Corresponds to Address option that is specified when accessing the bit address in bytes when connecting to Ethernet.

[5] Corresponds to Address option that is specified when accessing a word address or when byteaccessing a bit address when connecting serially.

3.1.1.1. Serial connection

Option

The following is a list of options to be specified for the serial connection.

Option	Required	Meaning
DeviceType	✓	Specify the device type in code. See Table 3-2 for the device types that can be specified.
PrgNo	-	Specifies the program number as a 1-digit hexadecimal character string. (See3.1.1.1.1)
Address	-	Specifies the device No. as a hexadecimal string. (see 3.1.1.1.2)
VT	-	Specifies the datatype to Put/Get. (see 3.1.1.1.3)
Elem	-	Specify the number of elements to Put/Get in decimal. (Default: 1)
Interval	-	Specifies the length of time between when PLC receives a command and when the response is returned as a one-digit hexadecimal number. Used only for serial connection. (see 3.1.1.1.4)
Array	-	Specifies whether to use an array type when reading one element. (Default: FALSE)

3.1.1.1.1. PrgNo Optional

Specifies which program area device is to be accessed. For the program numbers that can be specified, see the pull-gram number column in Table 3-2. If.PrgNo is omitted, the defaults corresponding to the device are used. The default value is indicated by bold and underbar in the pull gram number column in Table 3-2. "1, 2, 3" means the default value is "1".

3.1.1.1.2. Address Optional

Specify the leading address to read/write as a hexadecimal string. If it is omitted, it is treated as 0. There are two types of address values: bit and word. The specified address is determined automatically by the provider based on the specified device type (DeviceType), program number (PrgNo), number of elements (Elem), and data type (VT). Table 3-2 shows the range that can be specified for each address type.

When word addressing is used, a 'L', 'H' can be added to the end of the address. 'L' indicates that the low-order byte and 'H' indicates that the high-order byte is accessed. When this option is omitted, the lower byte of the specified address is accessed as the first byte. See the figure below for how addresses are represented.

	Bit address	Word address (Serial connection)	
Bit address area	X000	(LSB)	Lower byte
	X001	↑	
	X002	|	
	X003	X00L	
	X004	|	
	X005	|	
	X006	↓	
	X007	(MSB)	
	X008	(LSB)	High byte
	X009	↑	
	X00A	|	
	X00B	X00H	
	X00C	|	
	X00D	|	
	X00E	↓	
	X00F	(MSB)	

	Bit address	Word address (Serial connection)	
Word address area	D0000-0	(LSB)	Lower byte
	D0000-1	↑	
	D0000-2	|	
	D0000-3	D0000L	
	D0000-4	|	
	D0000-5	|	
	D0000-6	↓	
	D0000-7	(MSB)	
	D0000-8	(LSB)	High byte
	D0000-9	↑	
	D0000-A	|	
	D0000-B	D0000H	
	D0000-C	|	
	D0000-D	|	
	D0000-E	↓	
	D0000-F	(MSB)	

<Addressing of basic bit area (P,K,V,T,C,L,X,Y,M)>

.X/Y to be judged as shown in the figure below are the same area according to the combination of program No., number of elements and data type. If it is determined to be a word address, access is performed in byte units, and the bit position cannot be specified.

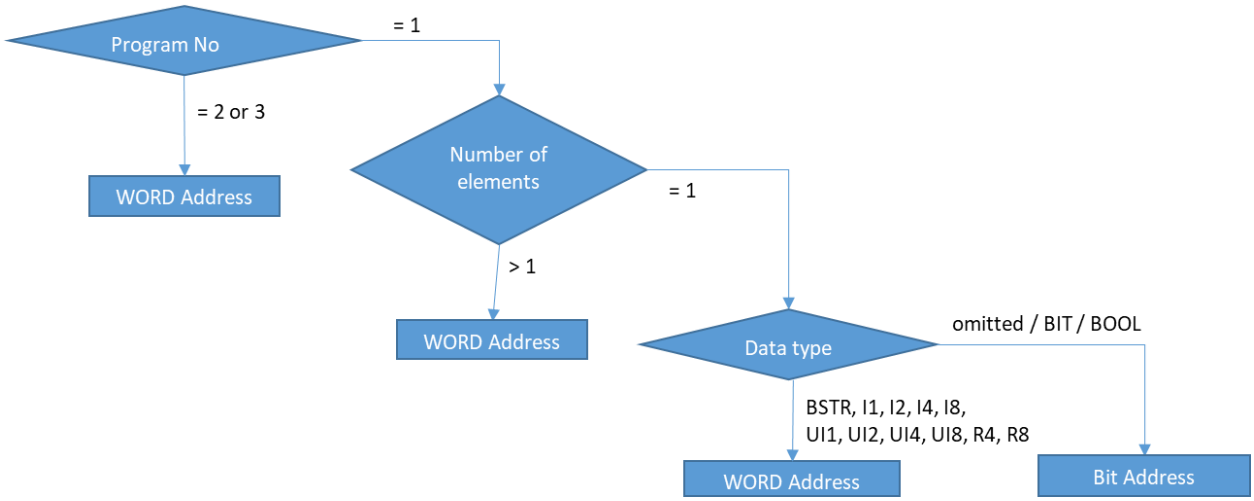


Figure 3-1 Addressing with basic bit devices

(Ex) DeviceType=X,Address=0001 (single bit access to bit device)

1 bit is read/written to the bit address X001, and when the address increases by 1, it moves by 1 bit.

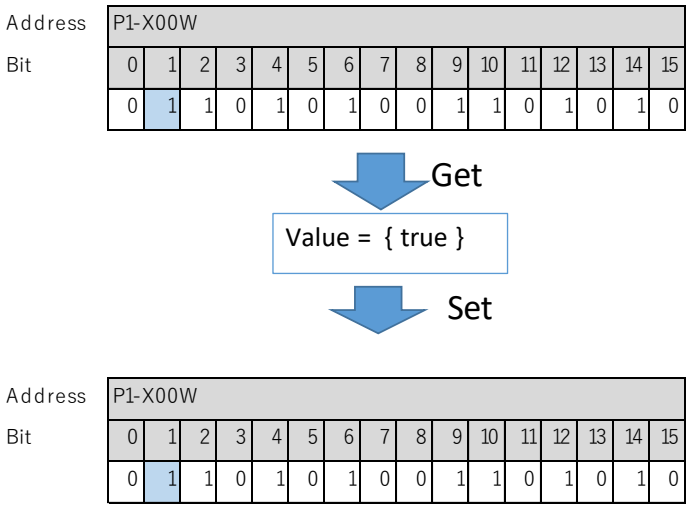


Figure 3-2 Bit-based access to basic bit devices

(e.g.) DeviceType=X,Address=0001, Elem=2 (multi-bit access to basic bit device)

When reading, 2 bits are read starting from the word-address X01W. When writing, write in 1-byte units starting from the word address X01W. Bits 0 to 1 write the specified data and bits 2 to 7 fill with 0.

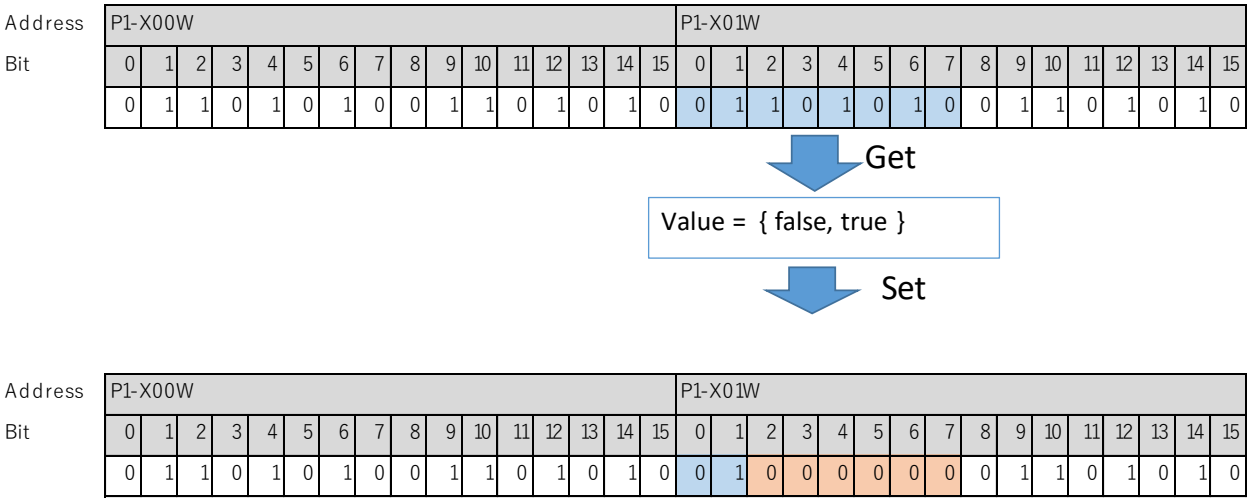


Figure 3-3 Multiple-bit access to basic bit device

(Example) DeviceType=X,Address=0001H,VT=UI2,Elem=2 (basic bit device accessed in byte units)

Two bytes are read and written starting from the upper byte address of word address X01W, and when the address increases by 1, one word is moved.

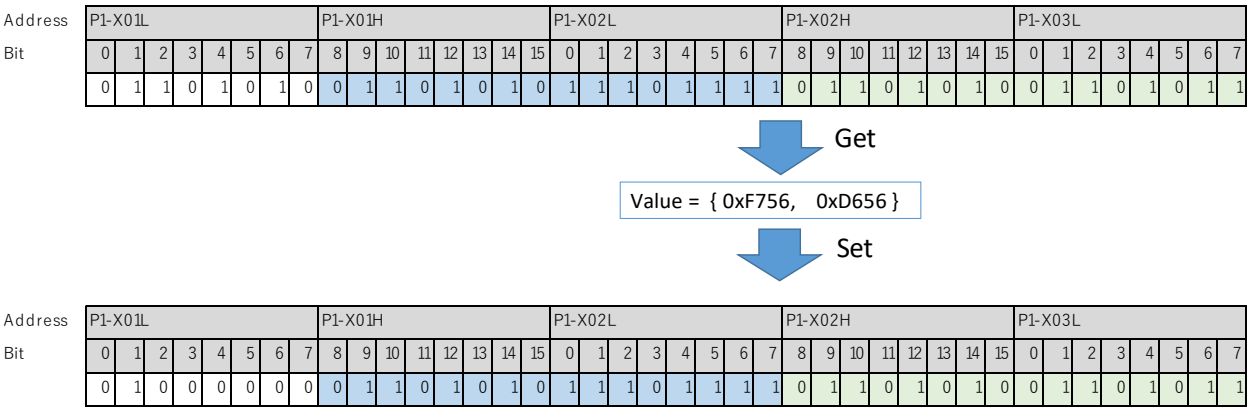


Figure 3-4 Byte-based access to basic bit devices

<Addressing of basic bit area for PC10 (P1, V1, T1, C1, L1, L2, M1)>

It is always judged as a word address and accessed in byte units. Specify the most significant digit of the address in DeviceType, and specify the word address calculated based on the bit address of the last three digits in Address.

(Example) DeviceType=P1, Address=1F, Elem=1 (single-byte access to basic bit device for PC10)

The operation differs from the basic bit area. One byte is read/written starting from the word-address P11FW.

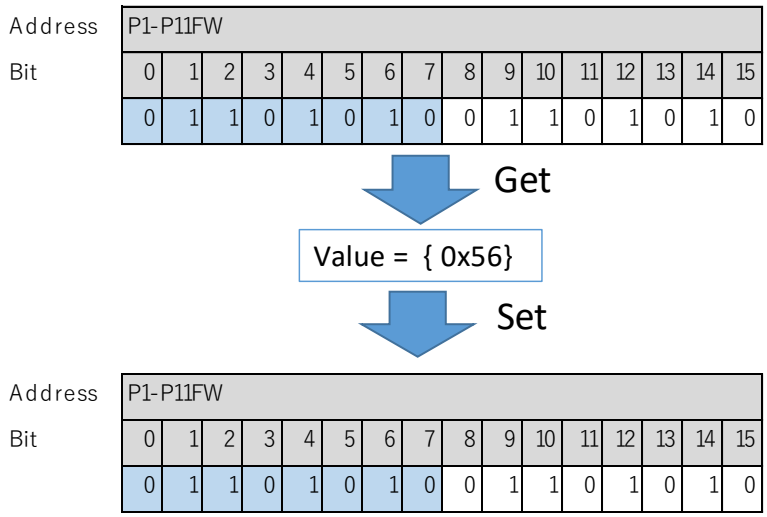


Figure 3-5 Single-Byte Access to Basic Bit Devices for PC1031

(Ex) DeviceType=P1, Address=1F, Elem=2 (multi-byte access to basic bit device for PC10)

The operation differs from the basic bit area. 2 bytes are read/written starting from the word address P11FW.

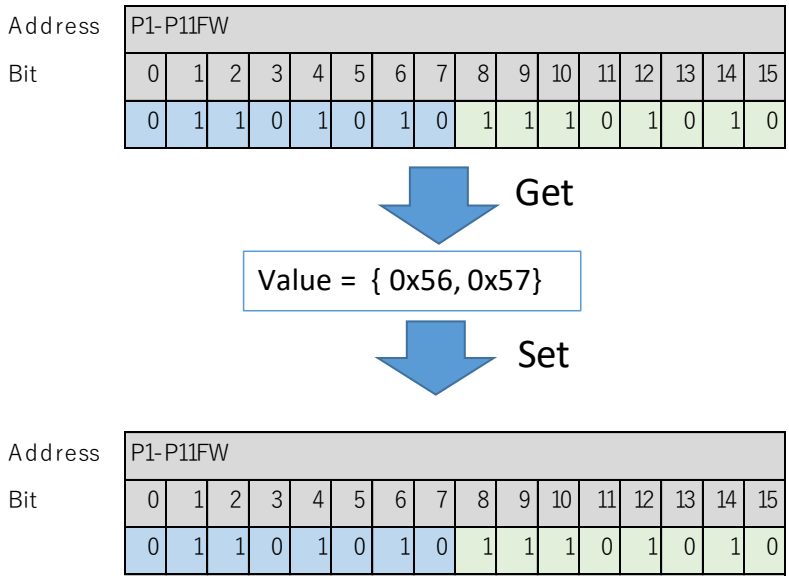


Figure 3-6 Basic bit device for PC10 is accessed by multiple bytes.

<Addressing of the base word area (S, N, R, D)>

It is always judged as a word address.

(Example) DeviceType=D,Address=0001, Elem=2 (word address is accessed in byte units)

Two bytes are read and written starting from the word address D0001, and when the address is increased

by 1, one word is moved.

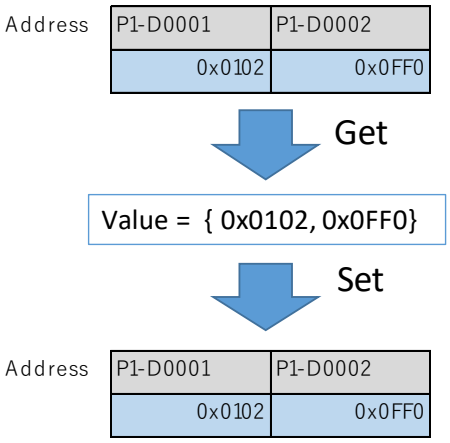


Figure 3-7. Multiple-Point Access to Base Word Device

<Addressing of the basic word area for PC10 (S1, N1, D1, D2)>

It is always judged as a word address. Specify the most significant digit of the address in DeviceType, and specify the word address of the last three digits in Address.

(Ex.) DeviceType=D1,Address=0001,VT=I2 (word address accessed in byte units)

Two bytes are read and written starting from the word address D1001, and when the address is increased by 1, one word is moved.

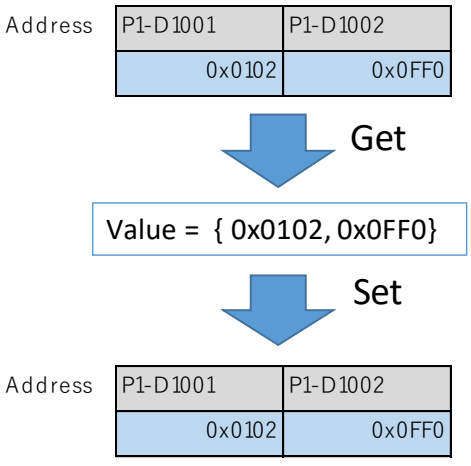


Figure 3-8. Multiple-point Access to Basic Word Device for PC10 Diagram

<Address of extended bit area 1, 2(EP, EK, EV, ET, EC, EL, EX, EY, EM, GX, GY, GM)>

It is always judged as a word address and accessed in byte units. EX/EY,GX/GY is the same area.

(Example) DeviceType=EK, Address=01F, Elem=2 (bit device accessed in byte units)

The operation differs from the basic bit area. When reading, 2 bytes are read and written starting from

the word-address EK01FW.

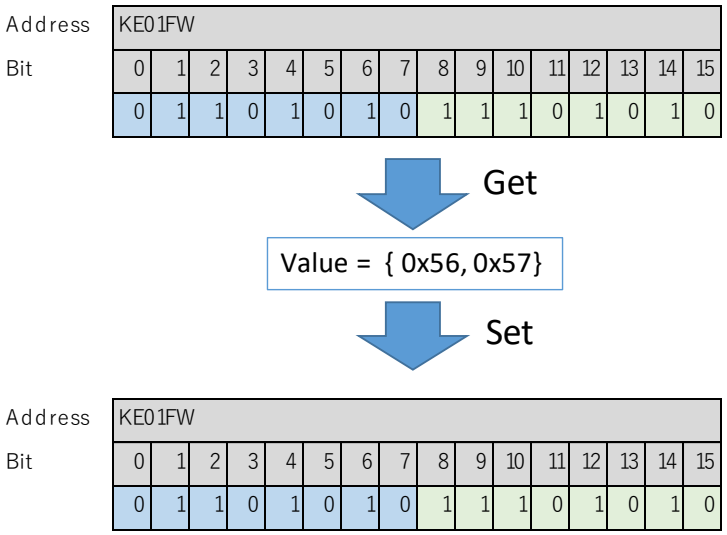


Figure 3-9: Multiple byte access to extended bit devices

<Address of extended word address area 1, 2 (ES, EN, H, U)>

It is always judged as a word address. When accessing an address extended after PC10 (U008000 or larger address), specify "*" in the program number.

(Example) DeviceType=U, Address=0F10, Elem=2 (extended word device accessed in byte units)

Two bytes are read and written starting from the word address U0F10, and when the address is increased by 1, one word is moved.

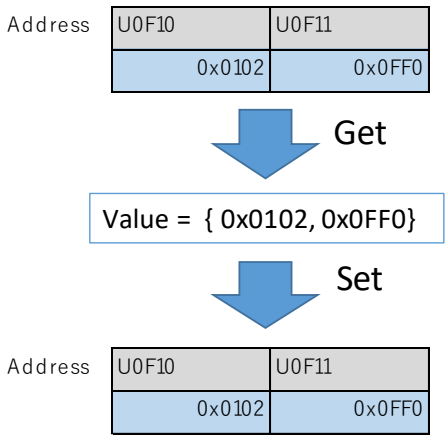


Figure 3-10: Multiple-point access to extended word devices

<Address of extended word address area 3 (EB)>

Always specify by word address. However, if the specified value of the program number is "9", "A", "B", or "C", the relative address is specified; if the specified value is "*", the absolute address is specified.

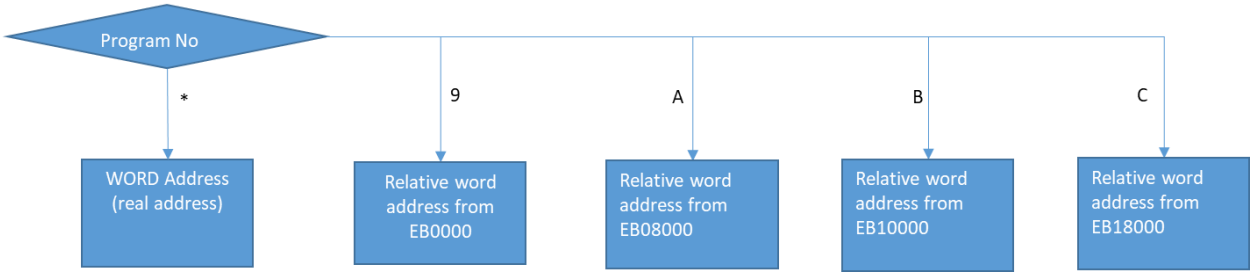


Figure 3-11: Addressing of extended word device

(Ex.) DeviceType=EB, PrgNo=C, Address=001F, Elem=2 (accessing EB1801F by specifying a relative address.)

The specifying method supported by all models. When.PrgNo = C, Address should be addressed relative to EB18000. For the correspondence between other program numbers and reference addresses, see Table 3-2. Two bytes are read and written starting from the word address EB1801F, and when the address is incremented by 1, one word is moved.

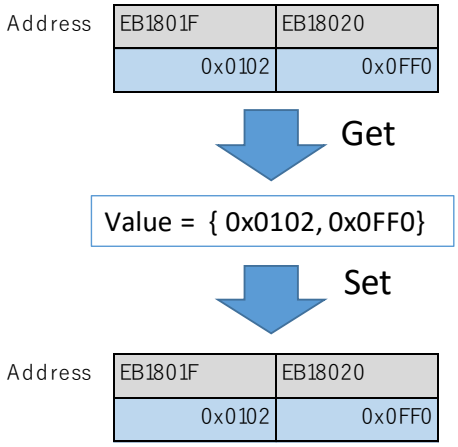


Figure 3-12 Access to extended word device 3 with relative address

(Ex.) DeviceType=EB, PrgNo=*, Address=1801F, Elem=2 (accessing EB1801F by specifying an absolute address.)

This specification is supported by PC10 or later models. Two bytes are read and written starting from the word address EB1801F, and when the address is incremented by 1, one word is moved.

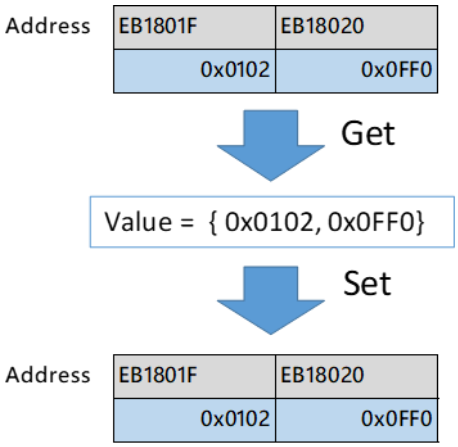


Figure 3-13 Access to extended word device 3 by absolute address

Ex. DeviceType=EB, PrgNo=*, Address=2FFFF (Accessing EB2FFFF by Specifying an Absolute Address)

Addresses extended by PC10 cannot be accessed with a relative address. Two bytes are read and written starting from the word address EB2FFFF, and when the address is incremented by 1, one word is moved.

<Address of extended word address area 4 (FR)>

It is always judged as a word address.

3.1.1.1.3. VT Optional

Specify the data type to read or write. The specification can be either a VT string or the corresponding VARTYPE (decimal value).

When VT option is omitted, the datatype that matches the address type of the device is used.

The following is a list of available data types:

If DeviceType option is specified other than the basic bit area (P,K,V,T,C,L,X,Y,M), BIT and BOOLs cannot be specified in VT.

Table 3-3: List of data types

VT	Data type	Description
BIT	VT_UI1	Data is converted to 0/1 binary value and written/read.
BOOL	VT_BOOL	Data is converted to 0/1 binary value and written/read.
BSTR	VT_BSTR	Write/read as Elem byte ASCII
I1	VT_I1	Write/read as 1-byte data (signed)
I2	VT_I2	Write/read as 2-byte data (signed)
I4	VT_I4	Write/read as 4-byte data (signed)

I8	VT_I8	Write/read as 8-byte data (signed)
UI1	VT_UI1	Write/read as 1-byte data (unsigned)
UI2	VT_UI2	Write/read as 2-byte data (unsigned)
UI4	VT_UI4	Write/read as 4-byte data (unsigned)
UI8	VT_UI8	Write/read as 8-byte data (unsigned)
R4	VT_R4	Write/read as 4-byte data (floating point)
R8	VT_R8	Write/read as 8-byte data (double precision floating point)

Table 3-4: Data types when unspecified

Address type	VT	Data type
Basic bit area	BIT	VT_UI1
Basic bit area for PC10	UI1	VT_UI1
Expansion bit area1, 2	UI1	VT_UI1
Basic word area	UI2	VT_UI2
Basic word area for PC10	UI2	VT_UI2
Expansion word area1, 2, 3, 4	UI2	VT_UI2

3.1.1.1.4. Interval Optional

Specifies the number of hexadecimal characters (0 to F) from when PLC receives a command until sending a response.

The correspondence between the specified character string and the transmission time is as follows. The default is 0.

Table 3-5: Response transmission time

Interval	Response time (ms)	Interval	Response time (ms)
0	0	8	80
1	10	9	90
2	20	A	100
3	30	B	200
4	40	C	300
5	50	D	400
6	60	E	500
7	70	F	600

(e.g.) When specifying the response time in 600ms

DeviceType=D,PrgNo=1,Address=FFF,Interval=F

3.1.1.2. Ethernet connection

The following shows the list to be specified in the option string when Ethernet is connected.

Option

The following is a list of options to be specified for the serial connection.

Option	Required	Meaning
DeviceType	✓	Specify the device type in code. See Table 3-2 for the device types that can be specified.List of addresses in 3-2 PLC
PrgNo	-	Specifies the program number as a 1-digit hexadecimal character string. (see 3.1.1.2.1)
Address	-	Specifies the device No. as a hexadecimal string. (see 3.1.1.2.2)
VT	-	Specifies the datatype to Put/Get. (see 3.1.1.2.3)
Elem	-	Specify the number of elements to Put/Get in decimal. (Default: 1)
Array	-	Specifies whether to use an array type when reading one element. (Default: FALSE)

3.1.1.2.1. PrgNo Optional

Specifies which program area device is to be accessed. For the program numbers that can be specified, see the pull-gram number column in Table 3-2. If.PrgNo is omitted, the defaults corresponding to the device are used. The default value is indicated by bold and underbar in the pull gram number column in Table 3-2. "1, 2, 3" means the default value is "1".

3.1.1.2.2. Address Optional

Specify the leading address to read/write as a hexadecimal string. If it is omitted, it is treated as 0. There are three types of address values: bit, byte, and word. The specified address is determined automatically by the provider based on the specified device type (DeviceType), program number (PrgNo), number of elements (Elem), and data type (VT). Table 3-2 shows the range that can be specified for each address type.

<Addressing of basic bit area (P,K,V,T,C,L,X,Y,M)>

.X/Y to be judged as shown in the figure below are the same area according to the combination of program No.,

number of elements and data type. If it is determined to be a byte address, access is performed in byte units, and the bit position cannot be specified. Due to the restrictions on the commands used for communication, if a value other than 1 is specified for the program number when accessing by bit addressing, an error will occur.

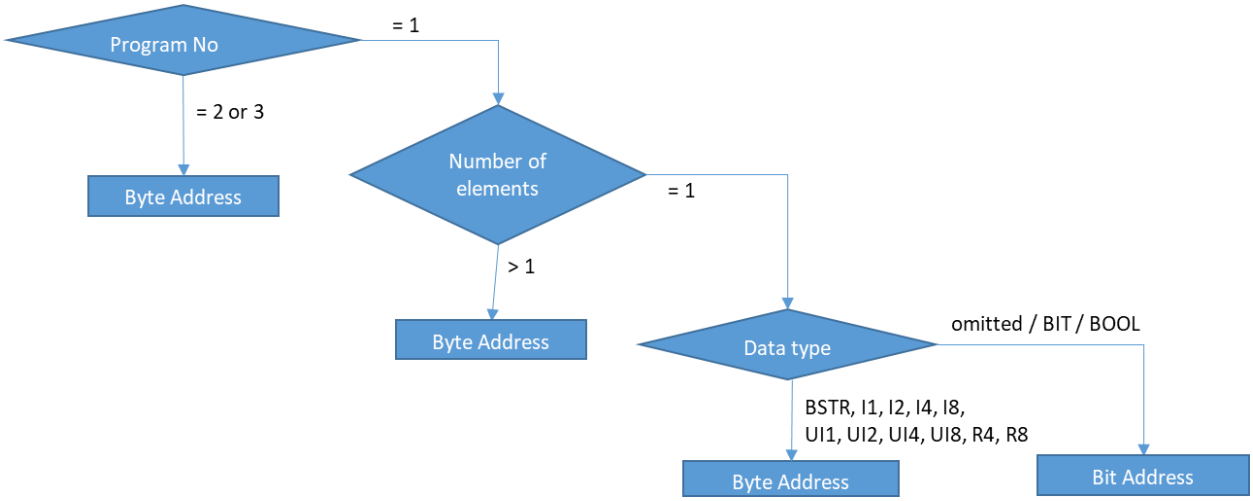


Figure 3-14: Addressing in basic bit devices

(Ex) DeviceType=X,Address=0011 (single bit access to bit device)

1 bit is read/written to the bit address X0011, and when the address increases by 1, it moves by 1 bit.

Address	P1-X01W															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0

Get

Value = { true }

Set

Address	P1-X01W															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0

Figure 3-15: Accessing basic bit devices in bit units

(e.g.) DeviceType=X,Address=0011, Elem=2 (multi-bit access to basic bit device)

Byte address is specified. When reading, 2 bits are read starting from the upper byte of word-address X08W. When writing, write in 1-byte units starting from the upper byte of word address X08W. Bits 0 to 1 write the specified data and bits 2 to 7 fill with 0.

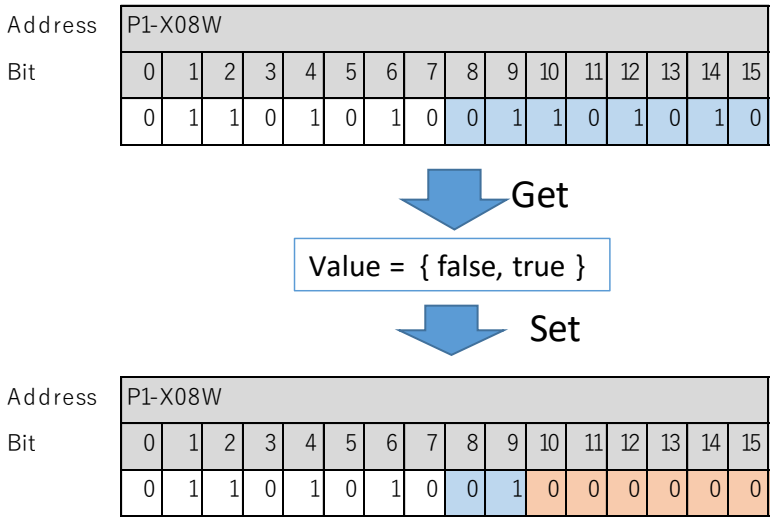


Figure 3-16 Multiple-bit access to basic bit device

(Example) DeviceType=X,Address=0001,VT=UI1,Elem=2 (basic bit device accessed in byte units)

Two bytes are read and written starting from the upper byte of word address X00W, and when the address increases by 1, one byte is moved.

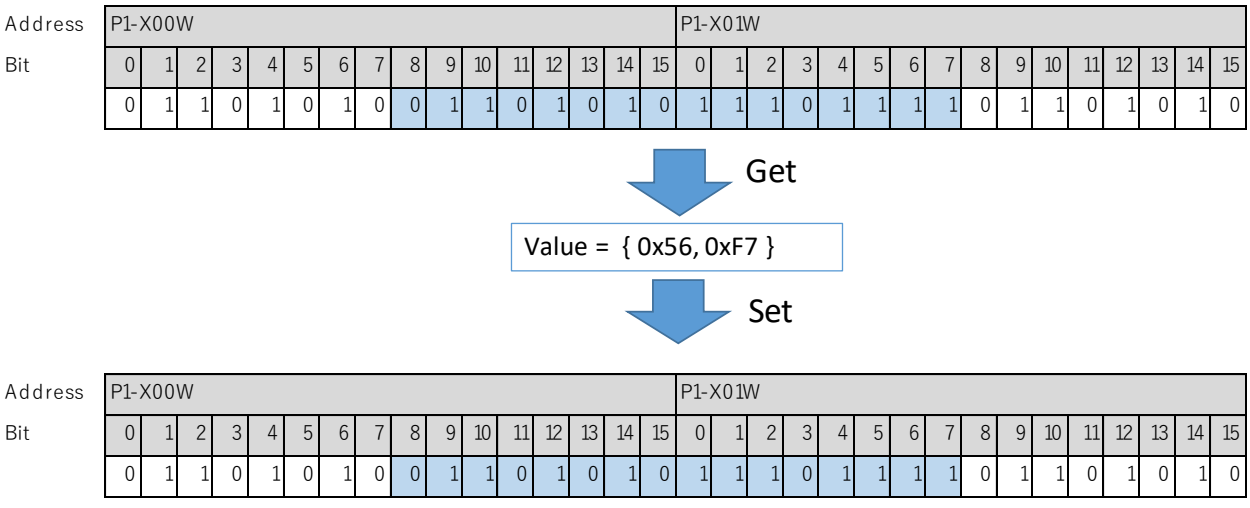


Figure 3-17 Byte-based access to basic bit devices

<Addressing of basic bit area for PC10 (P1, V1, T1, C1, L1, L2, M1)>

Byte address is always judged, and access is performed in byte units. Specify the most significant digit of the address in DeviceType, and specify the byte address calculated based on the bit address of the last three digits in Address.

(Example) DeviceType=P1, Address=3E, Elem=1 (single-byte access to basic bit device for PC10)

The operation differs from the basic bit area. Read/write 1 byte from/to the lower byte of word address P11FW.

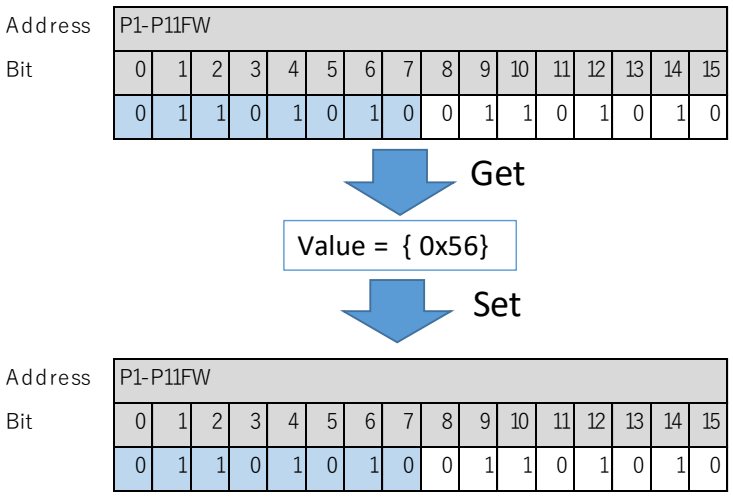


Figure 3-18 Single-Byte Access to Basic Bit Devices for PC10

(Ex) DeviceType=P1, Address=3E, Elem=2 (multi-byte access to basic bit device for PC10)

The operation differs from the basic bit area. Two bytes are read and written starting from the lower byte of the word address P11FW.

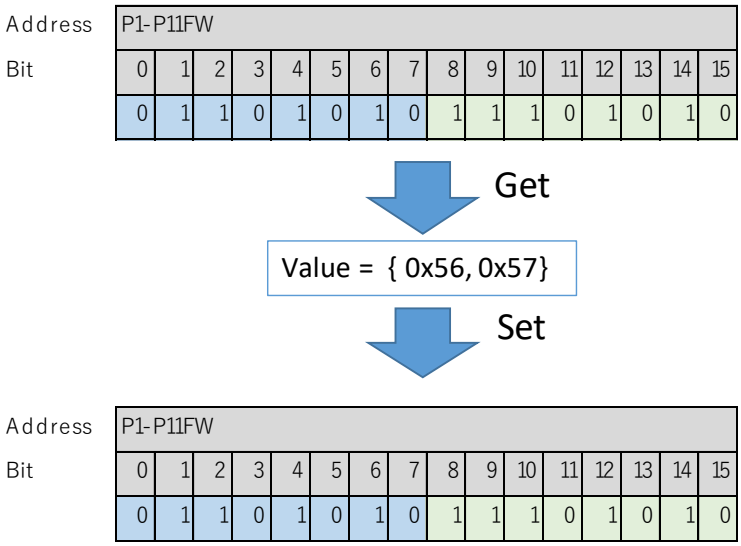


Figure 3-19 Basic bit device for PC10 is accessed by multiple bytes.

<Addressing of the base word area (S, N, R, D)>

It is always judged as a word address.

(Example) DeviceType=D,Address=0001, Elem=2 (word address is accessed in byte units)

Two bytes are read and written starting from the word address D0001, and when the address is increased by 1, one word is moved.

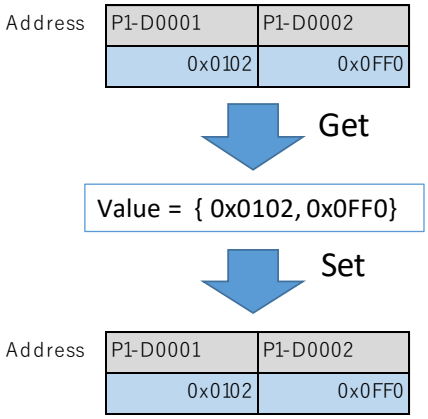


Figure 3-20: Multiple-point access to base word devices

<Addressing of the basic word area for PC10 (S1, N1, D1, D2)>

It is always judged as a word address. Specify the most significant digit of the address in DeviceType, and specify the word address of the last three digits in Address.

(Ex.) DeviceType=D1,Address=0001,VT=I2 (word address accessed in byte units)

Two bytes are read and written starting from the word address D1001, and when the address is increased by 1, one word is moved.

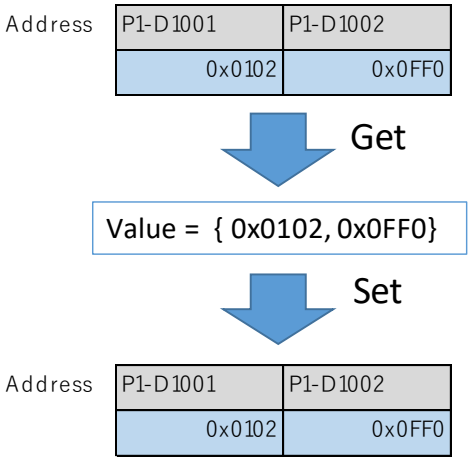


Figure 3-21 Multiple-point Access to Basic Word Device for PC10 Diagram

<Address of extended bit area 1, 2(EP, EK, EV, ET, EC, EL, EX, EY, EM, GX, GY, GM)>

Byte address is always judged, and access is performed in byte units. EX/EY,GX/GY is the same area.

(Example) DeviceType=EK, Address=01F, Elem=2 (bit device accessed in byte units)

The operation differs from the basic bit area. When reading, two bytes are read and written starting from the upper byte of the word-address EK0FW.

Address	KE00FW																KE01FW															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	1	1	0	1	1	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1	1	1	1	1	1	1	0	1	0	1	0

Get

Value = { 0x56, 0xF6}

Set

Address	KE00FW																KE01FW															
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	1	1	0	1	1	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1	1	1	1	1	1	1	0	1	0	1	0

Figure 3-22: Multiple byte access to extended bit devices

<Address of extended word address area 1, 2 (ES, EN, H, U)>

It is always judged as a word address. When accessing an address extended after PC10 (U008000 or larger address), specify "*" in the program number.

(Example) DeviceType=U, Address=0F10, Elem=2 (extended word device accessed in byte units)

Two bytes are read and written starting from the word address U0F10, and when the address is increased by 1, one word is moved.

Address	U0F10	U0F11
	0x0102	0x0FF0

Get

Value = { 0x0102, 0x0FF0}

Set

Address	U0F10	U0F11
	0x0102	0x0FF0

Figure 3-23: Multiple-point access to extended word devices

<Address of extended word address area 3 (EB)>

Always specify by word address. However, if the specified value of the program number is "9", "A", "B", or

"C", the relative address is specified; if the specified value is "*", the absolute address is specified.

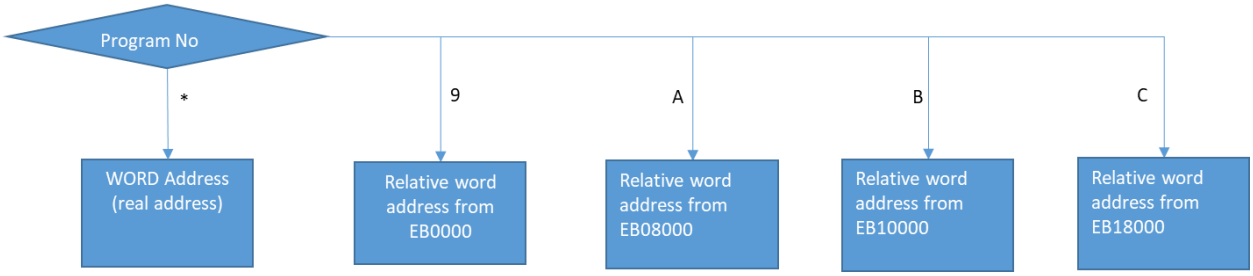


Figure 3-24: Addressing of extended word device

(Ex.) DeviceType=EB, PrgNo=C, Address=001F, Elem=2 (accessing EB1801F by specifying a relative address.)

The specifying method supported by all models. When.PrgNo = C, Address should be addressed relative to EB18000. For the correspondence between other program numbers and reference addresses, see Table 3-2. Two bytes are read and written starting from the word address EB1801F, and when the address is incremented by 1, one word is moved.

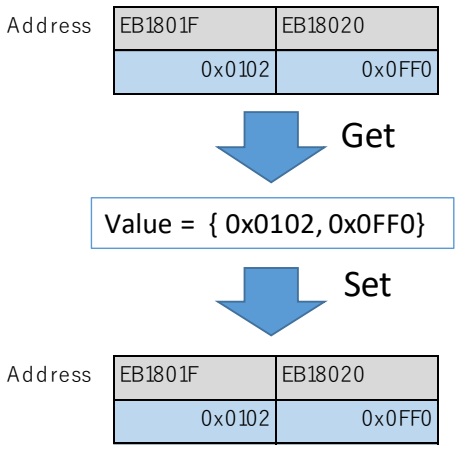


Figure 3-25 Access to extended word device 3 with relative address

(Ex.) DeviceType=EB, PrgNo=*, Address=1801F, Elem=2 (accessing EB1801F by specifying an absolute address.)

This specification is supported by PC10 or later models. Two bytes are read and written starting from the word address EB1801F, and when the address is incremented by 1, one word is moved.

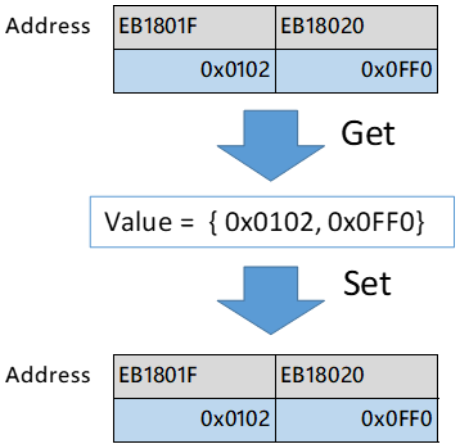


Figure 3-26 Access to extended word device 3 by absolute address

Ex. DeviceType=EB, PrgNo=*, Address=2FFFF (Accessing EB2FFFF by Specifying an Absolute Address)

Addresses extended by PC10 cannot be accessed with a relative address. Two bytes are read and written starting from the word address EB2FFFF, and when the address is incremented by 1, one word is moved.

<Address of extended word address area 4 (FR)>

It is always judged as a word address.

3.1.1.2.3. VT Optional

Specify the data type to read or write. The specification can be either a VT string or the corresponding VARTYPE (decimal value).

When VT option is omitted, the datatype that matches the address type of the device is used.

The following is a list of available data types:

If you specify a DeviceType option other than P,K,V,T,C,L,X,Y,M, you cannot specify BIT and BOOLs in VT.

Table 3-6: List of data types

VT	Data type	Description
BIT	VT_UI1	Data is converted to 0/1 binary value and written/read.
BOOL	VT_BOOL	Data is converted to 0/1 binary value and written/read.
BSTR	VT_BSTR	Write/read as Elem byte ASCII
I1	VT_I1	Write/read as 1-byte data (signed)
I2	VT_I2	Write/read as 2-byte data (signed)
I4	VT_I4	Write/read as 4-byte data (signed)
I8	VT_I8	Write/read as 8-byte data (signed)

UI1	VT_UI1	Write/read as 1-byte data (unsigned)
UI2	VT_UI2	Write/read as 2-byte data (unsigned)
UI4	VT_UI4	Write/read as 4-byte data (unsigned)
UI8	VT_UI8	Write/read as 8-byte data (unsigned)
R4	VT_R4	Write/read as 4-byte data (floating point)
R8	VT_R8	Write/read as 8-byte data (double precision floating point)

Table 3-7: Data types when unspecified

Address type	VT	Data type
Basic bit area	BIT	VT_UI1
Basic bit area for PC10	UI1	VT_UI1
Expansion bit area1, 2	UI1	VT_UI1
Basic word area	UI2	VT_UI2
Basic word area for PC10	UI2	VT_UI2
Expansion word area1, 2, 3, 4	UI2	VT_UI2

4. Command Reference

4.1. CaoController class-

List of 4-1 CaoController Class Commands

Command	Function	Page
Get status		
GetStatus	Send CPU status read command..	36
CPU Manipulation		
StopScan	Sends a scan stop command.	36
AwakeStopScan	Send the scan stop release command.	37
StartScan	Sends a scan restart command.[2]	37
Reset	Sends a reset command.	38

4.1.1. CaoController::Execute ("GetStatus") Command

Send CPU status read command to PLC. Returns the same result as get_Value for the system variable "@PLC_MODE".

Format GetStatus(<Interval>)

Interval : [in] The time from when the command is received until the response is sent.
Specify in hexadecimal character string (0 to F). Used only for serial connection. (VT_BSTR)

Return value : [List of out] CPU statuses (VT_ARRAY|VT_I2)
See "@PLC_MODE" in Table 3-1 for each data in the array.Table 3-1 CaoController Class-Variable List31

Examples of use

```
Object status = ctrl.Execute("GetStatus", "0");
```

4.1.2. CaoController::Execute ("StopScan") Command

Send the Stop Scan command to PLC.

Format StopScan(<PrgNo>, <Interval>)

PrgNo : [in] Program number to stop scanning. Used only for serial connection. (VT_BSTR)

[2] To resume scanning, you must be running Cancel from Scan. For more information, refer to the User's Manual of PLC.

0:Entire program, 2: program 2, 3: program 3

Interval [in] The time from when the command is received until the response is sent.
Specify in hexadecimal character string (0 to F). Used only for serial connection. (VT_BSTR)

Return value : None

Examples of use

```
Ctrl.Execute("StopScan", new object[] {"0", "0"});
```

4.1.3. CaoController::Execute ("AwakeStopScan ") Commands

Send the cancel scanning stop command to PLC.

Format AwakeStopScan (<PrgNo>, <Interval>)

PrgNo : [in] The program number to cancel the scan stop.
Used only for serial connection. (VT_BSTR)
0:Entire program, 2: program 2, 3: program 3

Interval [in] The time from when the command is received until the response is sent.
Specify in hexadecimal character string (0 to F). Used only for serial connection. (VT_BSTR)

Return value : None

Examples of use

```
Ctrl.Execute("AwakeStopScan", new object[] {"0", "0"});
```

4.1.4. CaoController::Execute ("StartScan") Command

Sends a scan restart command to PLC.

Format StartScan (<PrgNo >, <Interval>)

PrgNo : [in] Program number to resume scanning. Used only for serial connection. (VT_BSTR)
0:Entire program, 2: program 2, 3: program 3

Interval [in] The time from when the command is received until the response is sent.
Specify in hexadecimal character string (0 to F). Used only for serial

connection. (VT_BSTR)

Return value : None

Examples of use

```
Ctrl.Execute("StartScan", new object[] {"0", "0"});
```

4.1.5. CaoController::Execute ("Reset") Command

Sends a reset command to PLC.

Format Reset(<Interval>)

Interval [in] The time from when the command is received until the response is sent.
Specify in hexadecimal character string (0 to F). Used only for serial connection. (VT_BSTR)

Return value : None

Examples of use

```
Ctrl.Execute("Reset", "0");
```

5. Sample program

The following shows an example of C# for setting and acquiring I2 from the address "D0000".

List 5-151

```
... (omitted)...
Using ORiN2.ManagedCAO;

Namespace CMP_LINK_Sample
{
    Public partial class Sample : Form
    {
        Private CCaoEngine engine;
        Private CCaoWorkspaces wss;
        Private CCaoWorkspace ws;
        Private CCaoControllers ctrls;
        Private CCaoController ctrl;
        Private CCaoVariable value;

        Public Sample()
        {
            InitializeComponent();
        }

        Private void Sample_Load(object sender, EventArgs e)
        {
            Generating // Cao Engines
            This.engine = new CCaoEngine();

            This.wss = this.engine.Workspaces;
            This.ws = this.wss[0];
            This.ctrls = this.ws.Controllers;

            // Connect Optional (TCP)
            String option = "Conn=TCP:192.168.0.100:6000";

            // For a serial connection, do the following:
            // string option = "Conn=COM:1:115200:E:8:1,StationNumber=37";

            Connecting to a // CMP-LINK
            This.ctrl = this.ws.AddController("Sample",
                                              "CaoProv.JTEKT.CMP-LINK",
                                              "",
                                              Option);

            // Variables for access added
            This.value = ctrl.AddVariable("Item", "DeviceType=D,Address=0000,VT=I2");
        }

        Private void btnPut_Click(object sender, EventArgs e)
        {
            // Setting Values
            This.value.Value = Int16.Parse(txtItemValue.Text);
        }

        Private void btnGet_Click(object sender, EventArgs e)
        {
            // Get value
            Int16 val = (Int16)this.value.Value;
            TxtItemValue.Text = val.ToString();
        }
```

```
        Private void Sample_FormClosing(object sender, FormClosingEventArgs e)
        {
            Releasing // Cao Related Objects
            If (this.engine != null)
            {
                This.engine.Dispose();
                This.engine = null;
            }
        }
    }
}
```

6. Error code

CMP-LINK providers return the following unique error codes:

Table 6-1 Unique error codes

Error Name	Error Number	Description
No response	0x80100000	Returns when PLC cannot receive data. Check that the communication setting of.PLC matches the connectionoption.
Receive data error	0x80100001	It is returned when the checksum does not match or unexpected data is received.
Received data missing	0x80100002	It is returned when missing data (less than the minimum data size, etc.) is found in the received data.
Data conversion error	0x80100003	Returns the error response code or received data type conversion failure.
Write mode change error	0x80100004	Returns when switching to write mode failed due to data writing during serial connection.
Error response	0x801001XX	This message is returned when an error is received as a result of a command response. The hexadecimal error code () received from PLC is inserted into XX.[3] Example) 21 → 0x80100121 3F → 0x8010013F

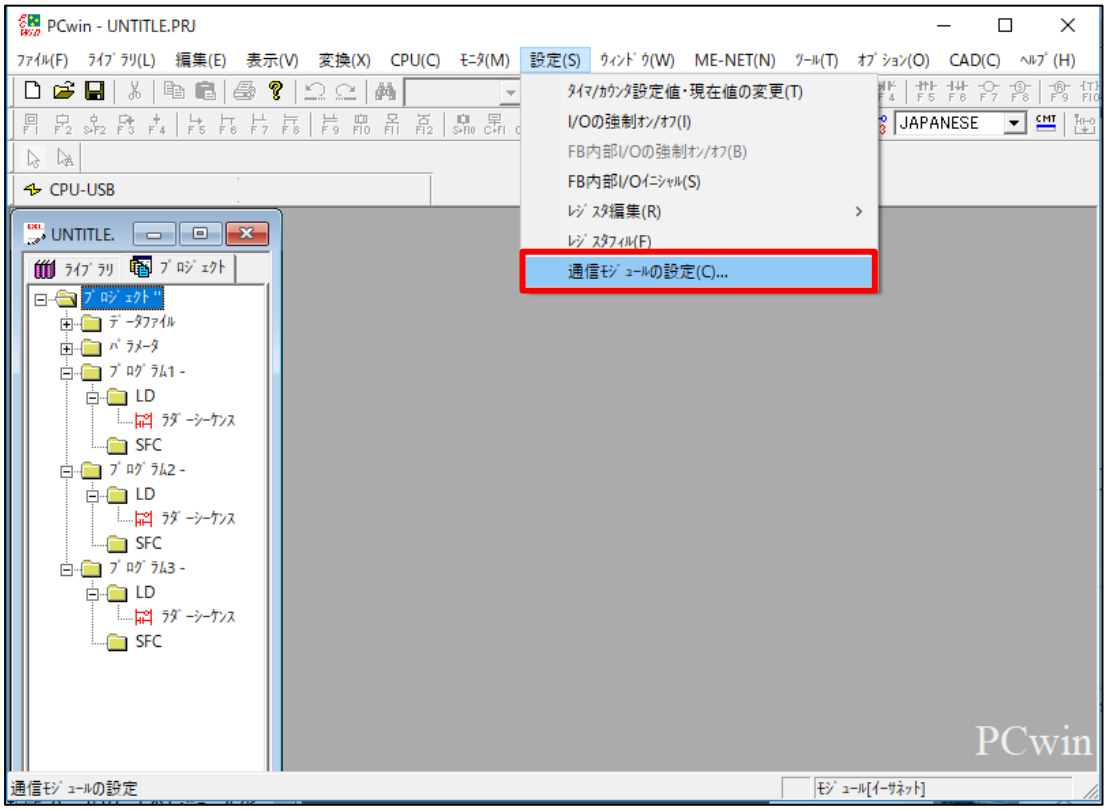
[3] Refer to the instruction manual of PLC for detailed error codes.

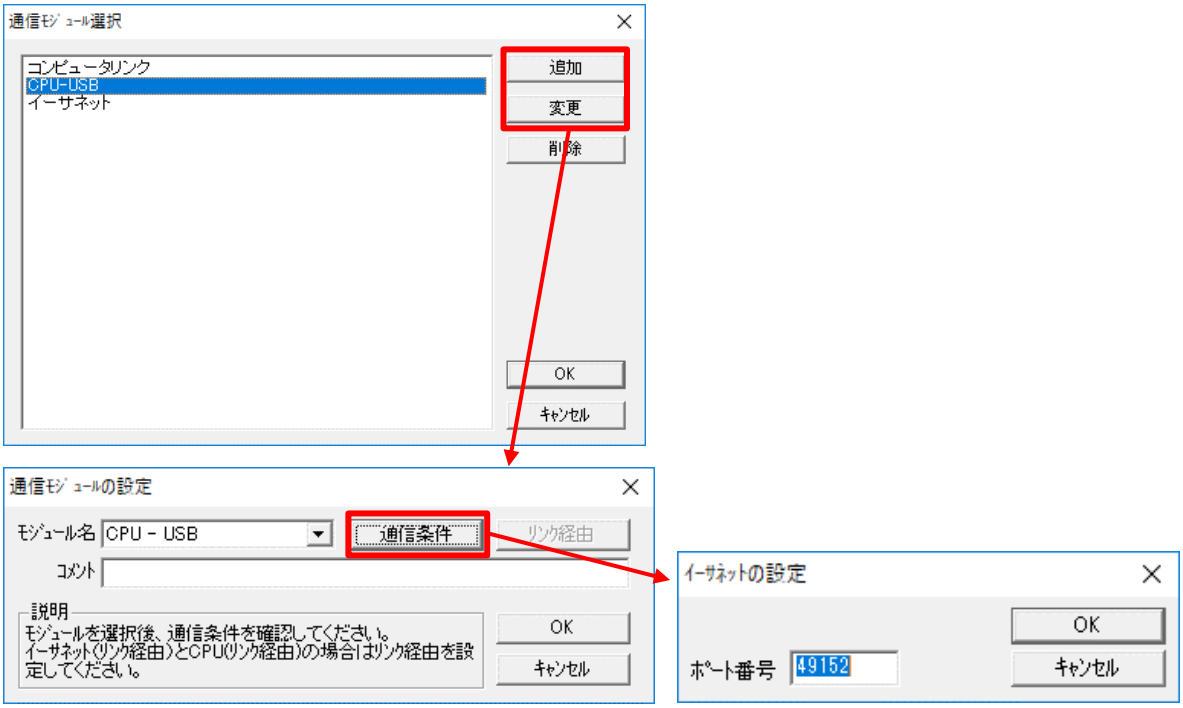
7. Environment setup for application development

This chapter describes PC10G communication settings required to connect to PC10G series-using CMP-LINK providers-from a client PC.

7.1. Preparation

The communication settings are made using the configuration PCwin provided with PC10G. Install PCwin and read the present settings prior to configuring communication settings. See the documentation that came with PCwin for more information about.PCwin.

1	Install PCwin on the client PC and connect the client PC and PC10G's CPU with USB cabling.
2	Start PCwin and configure the settings for communicating with CPU via USB. Select [Settings]-[Communication Module Settings] from the menu. 
3	Press [Add] on the communication module selection screen. (If CPU-USB is already displayed in the list, press [Change].)



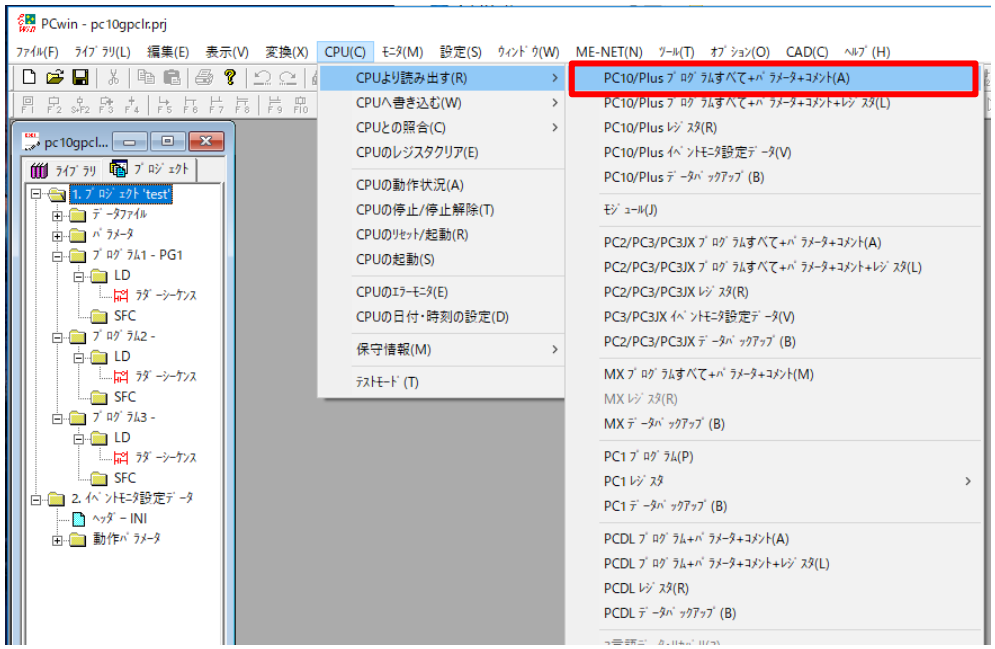
Set the module name to [CPU-USB] in the communication module setting window.

To specify a port, touch [Communication Conditions], and then enter the port number.

4 Return to the communication module selection screen. With the [CPU-USB] line selected, press [OK] to finish the setting.

5 Reads the present setting from CPU.

Select [Read from CPU]-[CPU]-[All PC10/Plus Programs + Parameters + Comments] from the menu.



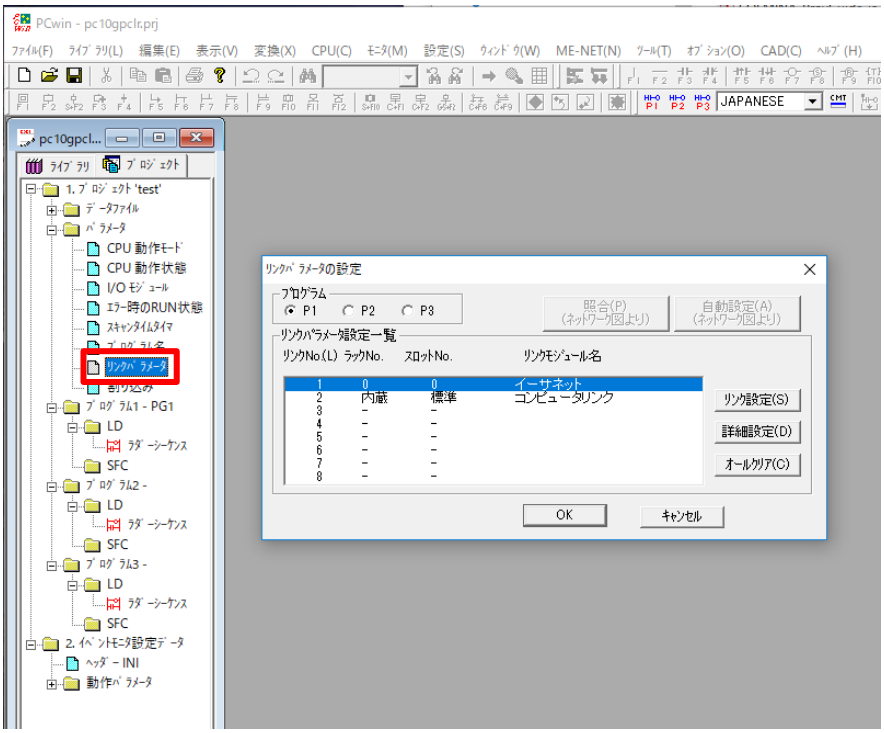
7.2. Serial Communication Settings

The following describes the settings for using serial communication (computer link). Refer to 7.1 for

preparations in advance.

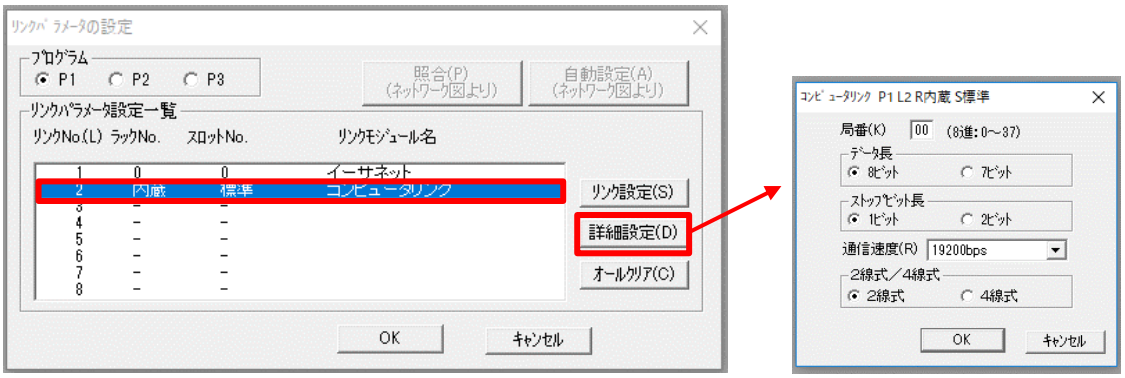
1 Select [Link Parameter] from PCwin project menu.

Set the computer link for the program (P1□P3) to be connected in the link parameter setting window.



2 Set [Built-in] for the rack No, [Normal] for the slot No, and [Computer Link] for the link module name.

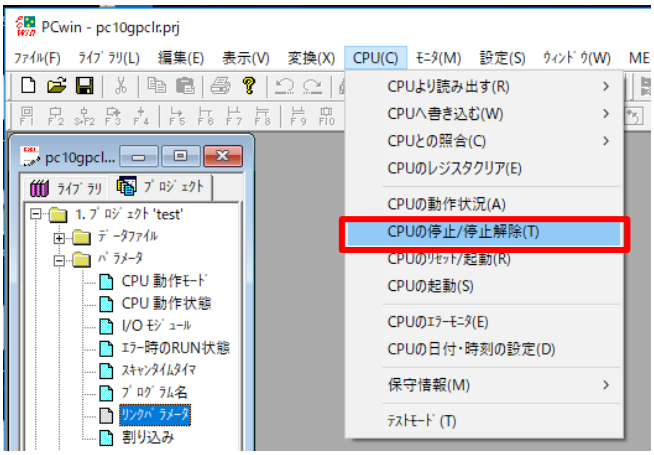
Press the Advanced button to set the serial communication parameters.



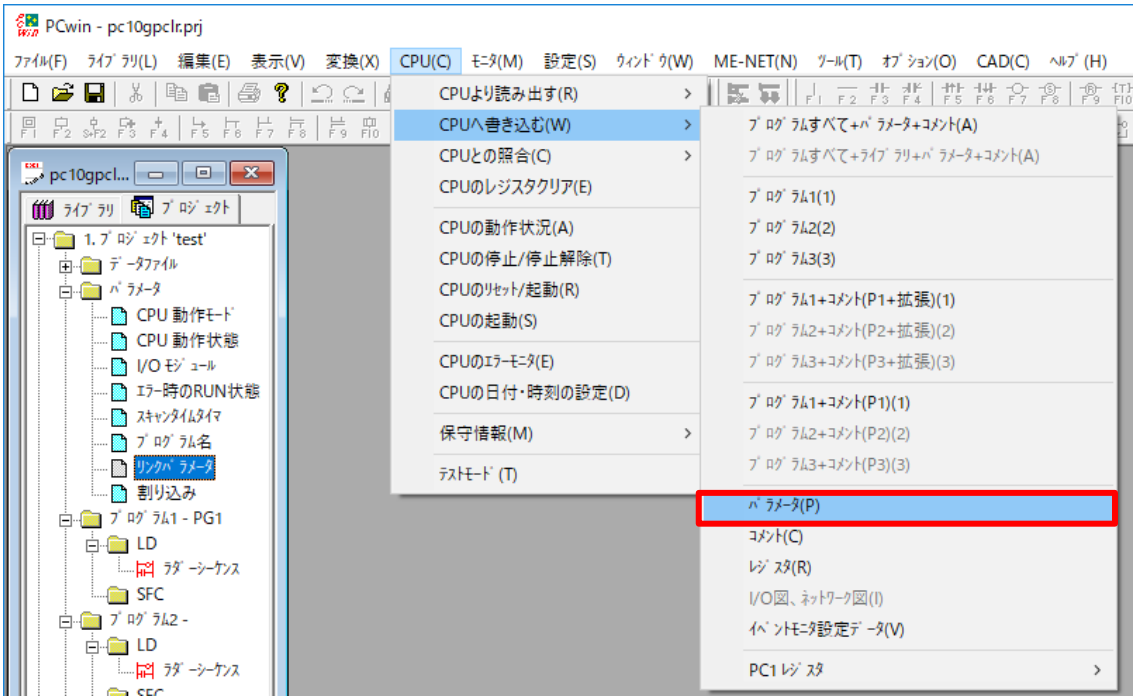
※2-wire/4-wire type can be selected in the window, but only 2-wire type can be selected for PC10G.

3 CPU must be stopped to write parameters to CPU.

Select [Stop/Cancel CPU]-[CPU] from the menu. Then, stop CPU.



4 Select [Write to CPU]-[CPU] from the menu. The set parameters are reflected in CPU.



5 After writing is complete, select [Stop/Cancel CPU]-[CPU] from the menu. CPU will be stopped.

Then, start CPU by pressing Reset + Start or by selecting [CPU]-[CPU Reset/Start] from PCwin menu.

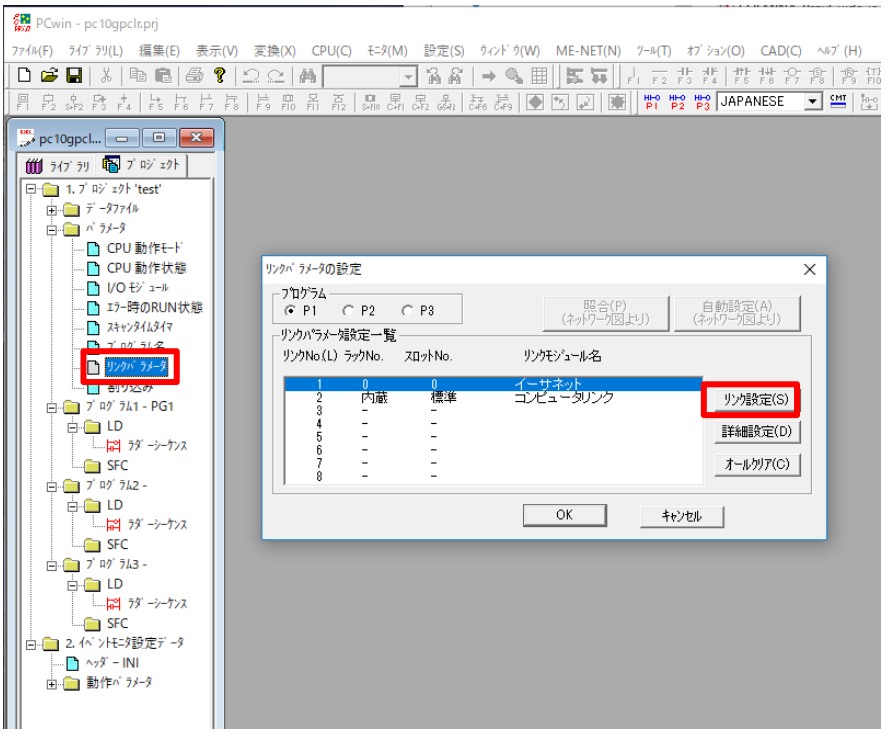
This completes the serial communication settings.

7.3. Ethernet communication settings

The following describes the setting when Ethernet communication is used. Refer to 7.1 and prepare in advance. To communicate with.PC10G's CPU via Ethernet, use the built-in CPU computer link port (L1) or add an Ethernet unit. For communication using the Ethernet unit, refer to the manual of the Ethernet unit.[4]

1 Select [Link Parameter] from PCwin project menu.

Set Ethernet for the program (P1□P3) to be connected in the link parameter setting window.



2 Set the rack No and slot No, and link module names.

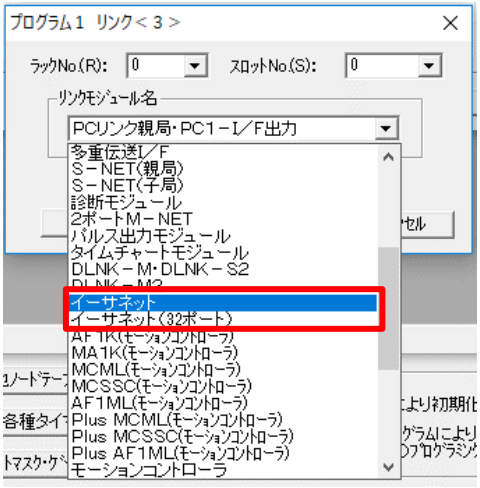
- When connecting with the built-in CPU port (L1),

Rack No	[Built in]
Slot No	[L1]

- When connecting using an Ethernet unit,

Rack No	Rack-mounted No with Ethernet unit
Slot No	Slot No with Ethernet unit

In both cases, set [Ethernet] or [Ethernet (32-port)] for the link module name.



3 Return to the link parameter setting screen, and press [Advanced].

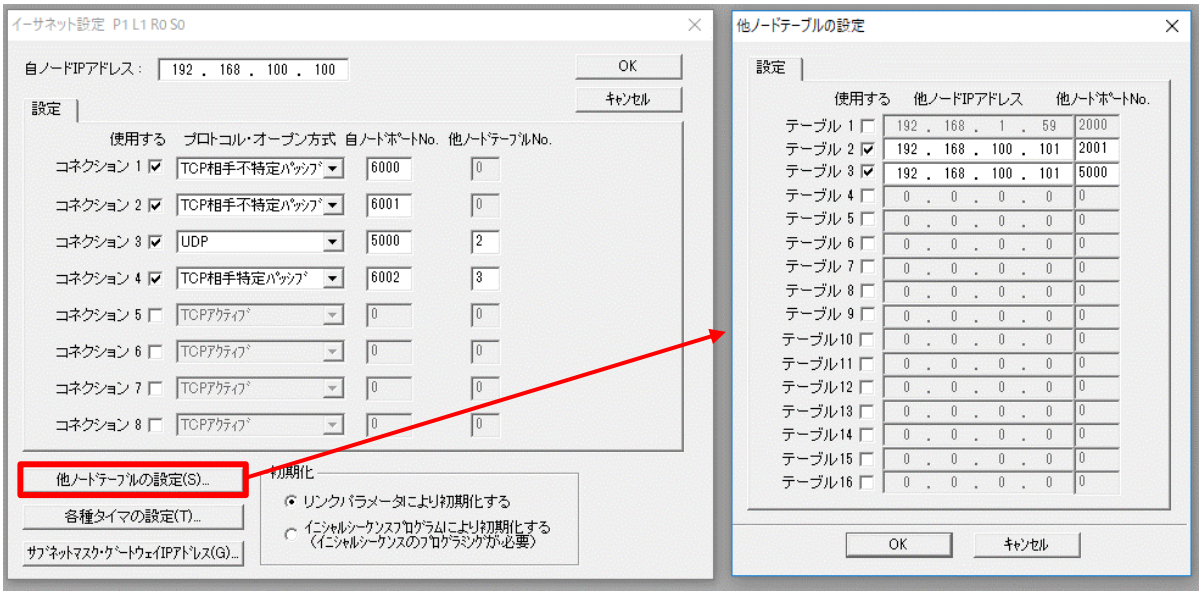
In the Ethernet Settings window, configure IP address/port settings.

[4] When developing this provider, the operation is checked using FL/ET-T-V2 which is a Ethernet unit in addition to the built-in port of CPU.

For the protocol/open method, usually select [TCP Destination Unspecified Passive].

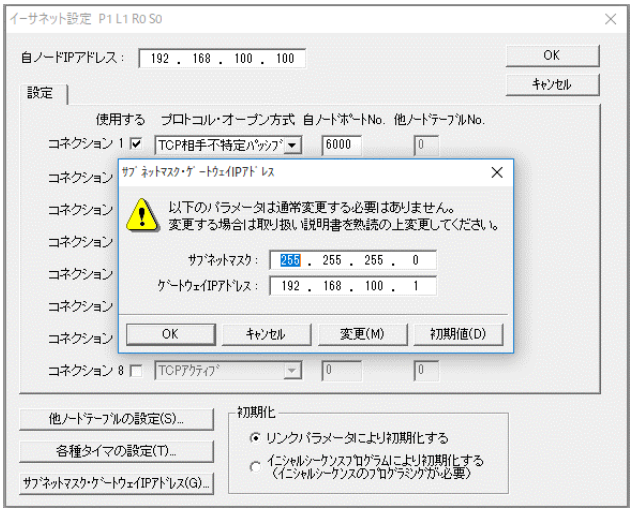
Select [TCP Destination-Specific Passive] if you want to allow connections only from certain IP addresses.

To use Destination Specific Passive, press [Set Other Node Table] and also set IP address of the connecting source.



4 If required, press [Subnet Mask/Gateway IP Address] in the Ethernet Settings window.

Sets the subnet mask gateway.



5 Follow steps 3 to 5 in 7.2.

This completes the setting of Ethernet communication.